FUJITSU

NetCOBOL for .NET V5.0

COBOL文法書

B1WD-3174-01Z0(00)
2012年12月

まえがき

NetCOBOLシリーズについて

NetCOBOLシリーズの最新情報については、富士通のサイトをご覧ください。

http://software.fujitsu.com/jp/cobol/

商標について

- HP、HP-UXは、米国Hewlett-Packard Companyの商標です。
- Micro Focus、Micro Focus COBOLおよびMicro Focus COBOL/2は、Micro Focus International Limitedの登録商標または商標です。
- OracleとJavaは、Oracle Corporationおよびその子会社、関連会社の米国およびその他の国における登録商標です。文中の社名、商品名等は各社の商標または登録商標である場合があります。
- Microsoft、Windows、Windows Server、Windows Vistaおよび.NETは、米国Microsoft Corporationの米国およびその他の国における登録商標または商標です。
- Linuxは、Linus Torvalds氏の米国およびその他の国における登録商標あるいは商標です。
- UNIXは、米国およびその他の国におけるオープン・グループの登録商標です。
- Intel、Itaniumは、Intel Corporationの登録商標です。
- その他の会社名または製品名は、それぞれ各社の登録商標または商標です。

製品の呼び名について

本書では、各製品を次のように略記しています。あらかじめご了承ください。

正式名称	略称
Microsoft(R) Windows(R) operating system Version 3.1	Windows 3.1
Microsoft(R) Windows(R) 95 operating system	Windows 95
Microsoft(R) Windows(R) XP Home Edition operating system Microsoft(R) Windows(R) XP Professional operating system	Windows XP
Microsoft(R) Windows Vista(R) Home Basic Microsoft(R) Windows Vista(R) Home Premium Microsoft(R) Windows Vista(R) Business Microsoft(R) Windows Vista(R) Enterprise Microsoft(R) Windows Vista(R) Ultimate	Windows Vista
Windows(R) 7 Home Premium Windows(R) 7 Professional Windows(R) 7 Enterprise Windows(R) 7 Ultimate	Windows 7 または Windows 7(x64)
Windows(R) 8 Windows(R) 8 Pro Windows(R) 8 Enterprise	Windows 8 または Windows 8(x64)
Microsoft(R) Windows Server(R) 2003, Standard Edition Microsoft(R) Windows Server(R) 2003, Enterprise Edition Microsoft(R) Windows Server(R) 2003 R2, Standard Edition	Windows Server 2003

正式名称	略称
Microsoft(R) Windows Server(R) 2003 R2, Enterprise Edition	
Microsoft(R) Windows Server(R) 2003, Standard x64 Edition Microsoft(R) Windows Server(R) 2003, Enterprise x64 Edition Microsoft(R) Windows Server(R) 2003 R2, Standard x64 Edition Microsoft(R) Windows Server(R) 2003 R2, Enterprise x64 Edition	Windows Server 2003 または Windows Server 2003(x64)
Microsoft(R) Windows Server(R) 2003, Enterprise Edition for Itanium-based Systems Microsoft(R) Windows Server(R) 2003, Datacenter Edition for Itanium-based Systems	Windows Server 2003 または Windows Server 2003 (Itanium)
Microsoft(R) Windows Server(R) 2008 Foundation Microsoft(R) Windows Server(R) 2008 Standard Microsoft(R) Windows Server(R) 2008 Standard without Hyper-V(TM) Microsoft(R) Windows Server(R) 2008 Enterprise Microsoft(R) Windows Server(R) 2008 Enterprise without Hyper-V(TM) Microsoft(R) Windows Server(R) 2008 Datacenter Microsoft(R) Windows Server(R) 2008 Datacenter without Hyper-V(TM)	Windows Server 2008 または Windows Server 2008(x64)
Microsoft(R) Windows Server(R) 2008 R2 Foundation Microsoft(R) Windows Server(R) 2008 R2 Standard Microsoft(R) Windows Server(R) 2008 R2 Enterprise Microsoft(R) Windows Server(R) 2008 R2 Datacenter	Windows Server 2008 または Windows Server 2008(x64) または Windows Server 2008 R2
Microsoft(R) Windows Server(R) 2008 for Itanium-Based Systems	Windows Server 2008 または Windows Server 2008 (Itanium)
Microsoft(R) Windows Server(R) 2012 Datacenter Microsoft(R) Windows Server(R) 2012 Standard Microsoft(R) Windows Server(R) 2012 Essentials Microsoft(R) Windows Server(R) 2012 Foundation	Windows Server 2012

本書の目的

本書では、FUJITSU NetCOBOL、FUJITSU COBOL97およびFUJITSU COBOL85のCOBOL(COmmon Business Oriented Language)に従ったプログラムを書くための規則を説明します。

本書は、プログラミングの基本知識を習得された方を対象にしています。

本書の位置付け

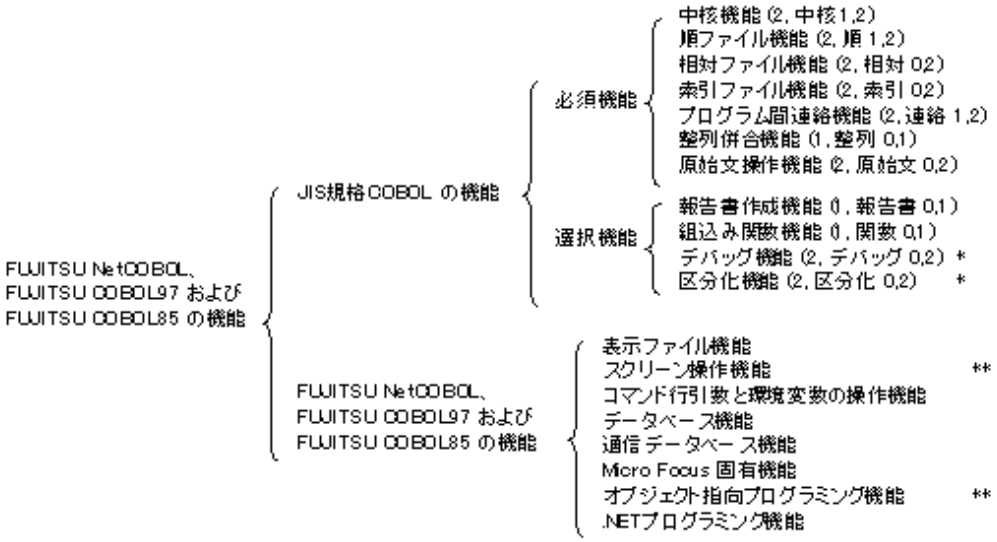
FUJITSU NetCOBOL、FUJITSU COBOL97およびFUJITSU COBOL85のマニュアルには、本書のほかに“NetCOBOL 使用手引書”があります。

プログラムの作成、翻訳、実行およびデバッグの方法については、“NetCOBOL 使用手引書”を参照してください。

その他のシステムについては、NetCOBOL 使用手引書をCOBOL97 使用手引書またはCOBOL85 使用手引書と置き換えてお読みください。

FUJITSU NetCOBOL、FUJITSU COBOL97およびFUJITSU COBOL85の機能範囲

FUJITSU NetCOBOL、FUJITSU COBOL97およびFUJITSU COBOL85の機能は、JIS規格COBOL(JIS X 3002-1992)の機能とFUJITSU NetCOBOL、FUJITSU COBOL97およびFUJITSU COBOL85拡張機能で構成されます。FUJITSU NetCOBOL、FUJITSU COBOL97およびFUJITSU COBOL85の機能を、以下に示します。



* ：デバッグ機能および区分化機能は、実行時は注釈扱いになります。
ただし、【Win×64】ではサポートしていません。
**：国際規格COBOL2002に含まれる機能です。

参考

JIS規格COBOLの機能の中で“( )”で囲んだ部分は、規格水準の簡略記号です。左から、階層における水準位置、機能単位略号、水準が属する機能単位の最小および最大の水準を示します。

なお、本書ではFUJITSU NetCOBOL、FUJITSU COBOL97およびFUJITSU COBOL85を略して“COBOL”と表記します。

本書の構成

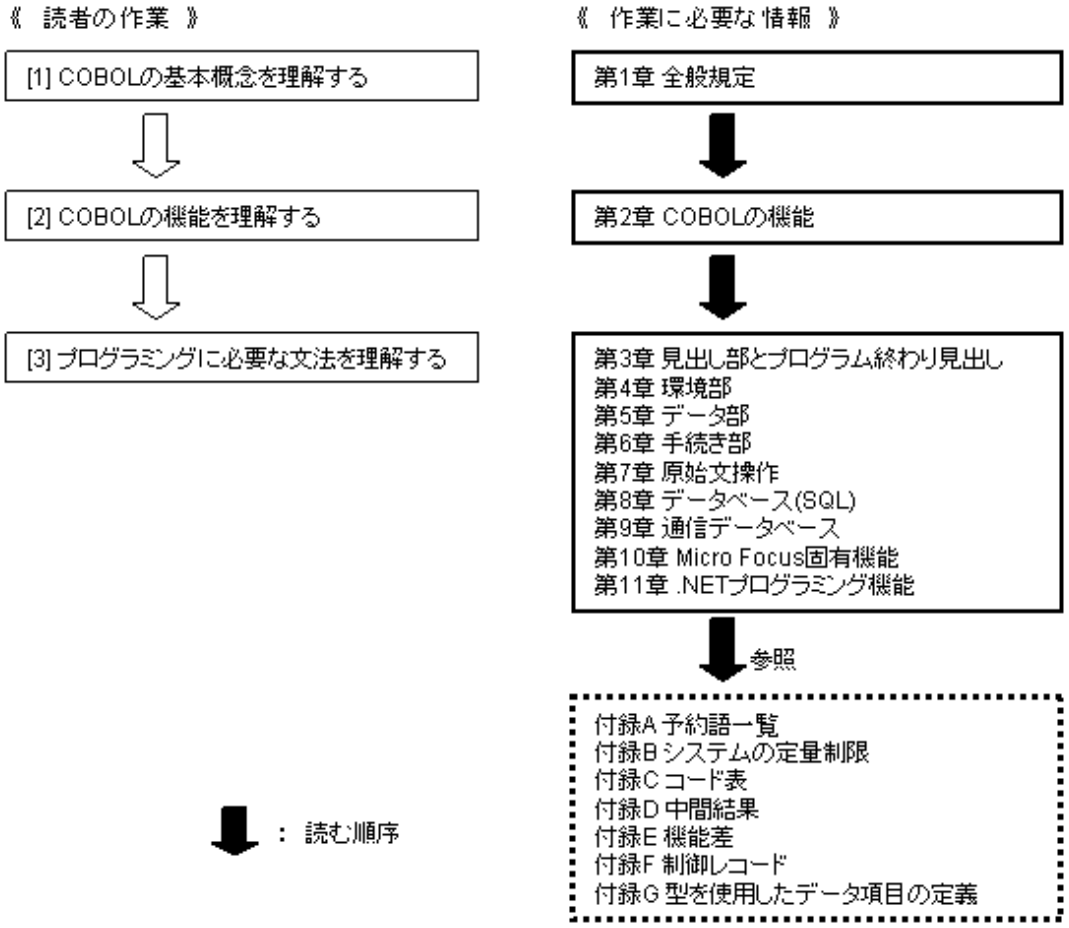
本書の構成と内容を、下表に示します。

構成	内容
第1章 全般規定	COBOL の言語要素、一意参照のしかた、定数の書き方、正書法など、全般規定を説明します。
第2章 COBOLの機能	COBOL の機能を、機能別に説明します。
第3章 見出し部とプログラム終わり見出し	見出し部とプログラム終わり見出しの文法を説明します。
第4章 環境部	環境部の文法を説明します。
第5章 データ部	データ部の文法を説明します。
第6章 手続き部	手続き部の文法を説明します。
第7章 原始文操作	原始文操作機能で使う文の文法を説明します。
第8章 データベース(SQL)	データベース機能の文法を説明します。
第9章 通信データベース	通信データベース機能の文法を説明します。
第10章 Micro Focus固有機能	Micro Focus 固有機能の文法を説明します。
第11章 .NETプログラミング機能	.NETプログラミング機能の文法を説明します。

構成	内容
付録A 予約語一覧	COBOL の予約語を示します。
付録B システムの定量制限	システムに依存する定量制限を示します。
付録C コード表	文字集合とその大小順序を示します。
付録D 中間結果	中間結果の属性と精度を説明します。
付録E 機能差	各システムでの機能差を示します。
付録F 制御レコード	制御レコードの書き方を説明します。
付録G 型を使用したデータ項目の定義	型の利用方法を説明します。

本書の読み方

COBOLの基本概念および機能を理解したい方は、最初に第1章および第2章を順にお読みください。COBOLの基本概念および機能を理解されている方は、必要に応じて第3章～第11章および付録をお読みください。本書の読み方を下図に示します。

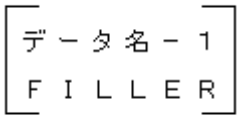
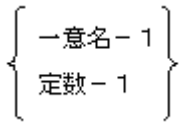


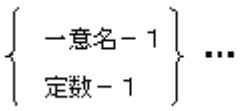
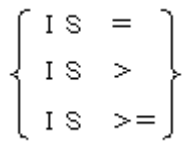
表記上の約束

【書き方】の中で使う記号について

各章の【書き方】では、文や句などCOBOLの言語要素の書き方を示します。書き方の中の各語は、書き方に示す順序で書かなければなりません。書き方に示す順序以外の順序で書いてもよい場合は、構文規則または一般規則に、その旨を記述しています。

【書き方】の中で使う記号の意味を、下表に示します。

記号	意味	用例
英大文字の文字列	COBOL の予約語です。 【書き方】に書いてあるとおりに書かなければなりません。	〔用例〕 VALUE IS 定数-1
下線付き文字列 （下線）	下線を付けた文字列が必要語であることを示します。 必要語は省略することはできません。 下線のない文字列は省略することができます。	〔意味〕 VALUE を省略することはできません。ISは省略することができます。定数-1には、構文規則に従った任意の定数を書くことができます。 例えば、以下のように書くことができます。
日本語の文字列	利用者語、定数、PICTURE 句の文字列および注記項の区別を示します。 この部分には、構文規則に従った任意の文字列を書くことができます。	VALUE IS "XXXX" VALUE 12345
[ ] （角括弧）	角括弧の中の1つを選択するか、省略することを示します。	〔用例1〕  〔用例1の意味〕 データ名-1を書くか、FILLERを書くか、または角括弧全体を省略することができ ます。 〔用例2〕 [ON SIZE ERROR 無条件文-1] 〔用例2の意味〕 角括弧全体を省略することができます。
{ } （波括弧）	波括弧の中の1つを選択することを示します。	〔用例〕 {一意名－１｜定数－１} または  〔意味〕 一意名-1および定数-1のいずれか1つを書くことができます。
{ | |} （選択指示子）	選択指示子の中の1つ以上を書くことを示します。ただし同じ文字列は1回しか書けません。	〔用例〕

記号	意味	用例
		 〔意味〕 以下のいずれか1つを書くことができます。 ・ COMMON ・ INITIAL ・ COMMON INITIAL ・ INITIAL COMMON
… (反復記号)	反復記号の直前の部分(角括弧または波括弧で囲まれた部分)を繰り返して書けることを示します。	〔用例〕  〔意味〕 一意名-1または定数-1を繰り返して書くことができます。
. (ピリオド)	書き方の中にピリオドが書いてある場合、その位置にピリオドを書かなければなりません。	〔用例〕 `WORKING-STORAGE SECTION.` 〔意味〕 SECTIONの後に、ピリオドを書かなければなりません。
＋、－、＞、＜、＝、＞＝、＜＝、－＞などの特殊文字	特殊文字は必要語です。下線部を付けていませんが、省略することはできません。	〔用例〕  〔意味〕 ＝、＞または＞＝を書かなければなりません。

網掛けについて

本文中の水色の網がけは、FUJITSU NetCOBOL、FUJITSU COBOL97およびFUJITSU COBOL85拡張機能またはFUJITSU NetCOBOL、FUJITSU COBOL97およびFUJITSU COBOL85固有の処理であることを示します。節、項などのタイトルに網掛けがある場合は、そのタイトルに含まれる記述全体が、FUJITSU NetCOBOL、FUJITSU COBOL97およびFUJITSU COBOL85拡張機能であることを示します。

構文規則および一般規則について

各章では、文や句などのCOBOLの言語要素を、書き方、構文規則および一般規則に分けて説明します。

書き方では、文や句を構成する要素の並べ方を示します。

構文規則では、書き方の中の各要素の並べ方および各要素の制限を説明します。

一般規則では、文や句を書いたときの実行結果および翻訳結果を説明します。書き方の中の各要素の意味、各要素間の相互関係も、一般規則で説明します。

書き方の中の各要素に関する規則がない場合は、構文規則または一般規則を省略しています。

廃要素について

本文中で“廃要素”と記述している要素は、1985版のANSI COBOL規格に含まれていますが、次期規格では削除される予定です。廃要素は、次期規格では動作が保証されません。新しく作成するプログラムでは、廃要素を使わないことをおすすめします。

システム固有機能について

本書で記述するNetCOBOL、COBOL97およびCOBOL85共通文法の中には、システムの機能に依存して、仕様が一部異なる部分があります。それらについては、以下に示すシステム記号を用いて本文中に記述しています。

図表内の記述	文章中の記述	対応システム
DS	[DS]	UXP/DS COBOL85 V20L11
HP	[HP]	HP-UX COBOL85 V20L11
Solaris	[Solaris]	Oracle Solaris NetCOBOL V10
Win16	[Win16]	Windows 95 COBOL85 V20L11 Windows 3.1 COBOL85 V20L11
Win32	[Win32]	Windows XP NetCOBOL V10 Windows Vista NetCOBOL V10 Windows 7 NetCOBOL V10 Windows 8 NetCOBOL V10 Windows Server 2003 NetCOBOL V10 Windows Server 2008 NetCOBOL V10 Windows Server 2012 NetCOBOL V10
WinIPF	[WinIPF]	Windows Server 2003(Itanium) NetCOBOL V10 Windows Server 2008(Itanium) NetCOBOL V10
Winx64	[Winx64]	Windows Server 2003(x64) NetCOBOL V10 Windows Server 2008(x64) NetCOBOL V10 Windows Server 2012 NetCOBOL V10 Windows 7(x64) NetCOBOL V10 Windows 8(x64) NetCOBOL V10
Linux	[Linux]	Red Hat(R) Enterprise Linux(R) 5(for x86) NetCOBOL V7.3L10 Red Hat(R) Enterprise Linux(R) 5(for Intel64) 32bit互換 NetCOBOL V7.3L10
LinuxIPF	[LinuxIPF]	Red Hat(R) Enterprise Linux(R) 5(for Intel Itanium) NetCOBOL V10
Linux64	[Linux64]	Red Hat(R) Enterprise Linux(R) 5(for Intel64) 64bit NetCOBOL V10 Red Hat(R) Enterprise Linux(R) 6(for Intel64) 64bit NetCOBOL V10
.NET	[.NET]	Windows 7 NetCOBOL for .NET V5.0 Windows 8 NetCOBOL for .NET V5.0 Windows Server 2008 R2 NetCOBOL for .NET V5.0 Windows Server 2012 NetCOBOL for .NET V5.0

なお、機能差については“付録E 機能差”を参照してください。

本書で説明されているCOBOL85および本書は、次に示す許諾に基づき富士通株式会社においてBULLグループが開発しているlibXpmを利用しています。COBOL85および本書の二次著作物としての著作権は、富士通株式会社が保有します。

Copyright (c) 1989-94 GROUPE BULL
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions :
The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.
THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,* FITNESS FOR A PARTICULAR PURPOSE AND NONINFRIGEMENT. IN NO EVENT SHALL* GROUPE BULL BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN * AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
Except as contained in this notice, the name of GROUPE BULL shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Software without prior written authorization from GROUPE BULL.

輸出管理規制について

本ドキュメントを輸出または提供する場合は、外国為替および外国貿易法および米国輸出管理関連法規等の規制をご確認の上、必要な手続きをおとりください。

2012年12月

Copyright FUJITSU LIMITED 1992-2012

謝辞

COBOLの言語仕様は、データシステムズ言語協議会(COnference on DAta SYstems Languages)の作業によって開発された原仕様に基づくものであり、本書で記述される仕様もまたそれに由来する。データシステムズ言語協議会の要求によって、以下の文章を掲げる。

COBOLは産業界の言語であって、いかなる会社、組織、団体等の占有物でもない。COBOLの委員会は、このプログラミング方式および言語の正確さと機能について、いかなる保証を与えるものでもなく、またそれに関連して、いかなる責任を負うものでもない。

次に示す著作権者は、原仕様書の作成に当たって、それぞれの著作物の一部分を利用することを承認した。この承認は、原仕様書をほかのCOBOLの仕様書で利用する場合にまで拡張されるものである。

- FLOW-MATIC(スペリランド社の商標),Programming for the Univac I and II,Data Automation Systems,スペリランド社 1958年,1959年,著作権.
- IBM Commercial Translator,図書番号 F28-8013, IBM社 1959年,著作権.
- FACT,図書番号 27A5260-2760,ミネアポリスハニウェル社 1960年,著作権.

目　次

第1章 全般規定 1
1.1 文字と文字集合 1
1.2 言語の基本要素 2
1.2.1 分離符 3
1.2.2 COBOLの語 5
1.2.2.1 利用者語 5
1.2.2.2 システム名 9
1.2.2.3 予約語 9
1.2.2.4 関数名 11
1.2.3 定数 11
1.2.3.1 数字定数 11
1.2.3.2 文字定数 12
1.2.3.3 16進文字定数 13
1.2.3.4 日本語定数 14
1.2.3.5 ブール定数 15
1.2.4 表意定数 15
1.2.5 連結式 16
1.2.6 特殊な用途の定数 17
1.2.7 PICTURE句の文字列 18
1.2.8 注記項 18
1.3 データ記述の概念 18
1.3.1 レベルの概念 18
1.3.2 字類の概念 18
1.3.3 型の概念 19
1.3.3.1 弱く型付けされた項目 19
1.3.3.2 強く型付けされた項目 19
1.3.4 標準桁よせ規則 20
1.3.5 データの境界調整 20
1.3.5.1 遊びバイト 20
1.3.5.2 遊びビット 23
1.4 一意参照 24
1.4.1 修飾 24
1.4.2 添字付け 27
1.4.3 部分参照 28
1.4.4 ポインタ付け 30
1.4.5 一意名 31
1.4.6 条件名の一意参照 31
1.4.7 関数一意名 32
1.5 正書法 32
1.5.1 1行の構成 32
1.5.2 A領域とB領域の正書法 33
1.5.3 空白行 34
1.5.4 注記行 34
1.5.5 行のつなぎ 35
1.5.6 デバッグ行 36
1.5.7 行内注記 36
1.5.8 自由形式の正書法 36
1.5.8.1 1行の構成 36
1.5.8.2 空白行 36
1.5.8.3 注記 36
1.5.8.4 行のつなぎ 36
1.5.8.5 デバッグ行 37
1.6 プログラムの構成 37
1.6.1 プログラムの結果 38

第2章 COBOLの機能....39
2.1 中核機能....39
2.1.1 データの転記....41
2.1.2 算術演算....41
2.1.3 選択処理と分岐....42
2.1.4 繰返し処理....43
2.1.5 表操作....44
2.1.6 データ項目の初期化....47
2.1.7 文字列操作....47
2.1.8 小入出力....48
2.1.9 プログラムの実行の終了....48
2.1.10 ポインタの操作....48
2.1.11 浮動小数点数の操作....49
2.2 入出力機能....51
2.2.1 ファイルの編成....51
2.2.2 ファイル結合子....53
2.2.3 入出力文の動作....53
2.2.4 ファイル位置指示子....54
2.2.5 ボリューム指示子....54
2.2.6 ファイルの共用と排他....54
2.2.7 レコードのロック....55
2.2.8 入出力状態....56
2.2.9 レコード形式....57
2.2.10 レコード領域....57
2.2.11 特殊レジスタ....57
2.3 プログラム間連絡機能....58
2.3.1 プログラムの呼出しと復帰....58
2.3.2 大域名と局所名....59
2.3.3 外部属性と内部属性....59
2.3.4 外部名と内部名....60
2.3.5 活性状態....60
2.3.6 プログラムの共通属性....60
2.3.7 プログラムの初期状態....60
2.3.8 パラメタの受渡し....61
2.3.9 名前の範囲....62
2.3.10 プログラム名および二次入口名の名前の範囲....63
2.3.11 特殊レジスタ....64
2.4 整列併合機能....64
2.4.1 整列の方法....65
2.4.2 併合の方法....65
2.4.3 入力手続き....65
2.4.4 出力手続き....65
2.4.5 整列併合用ファイル....65
2.4.6 特殊レジスタ....65
2.4.6.1 SORT-STATUS....66
2.4.6.2 SORT-CORE-SIZE....66
2.5 原始文操作機能....66
2.6 表示ファイル機能....67
2.6.1 あて先種別....68
2.6.2 画面帳票定義体....69
2.6.3 ファイルの編成と呼出し法....69
2.6.4 入出力文の動作....69
2.6.5 入出力状態....69
2.6.6 特殊レジスタ....70
2.7 組込み関数機能....71
2.8 スクリーン操作機能....73
2.8.1 画面と画面項目....73

2.8.1.1 画面データ記述項....73
2.8.1.2 画面項目....73
2.8.2 画面の入出力操作....74
2.8.3 画面入力状態....74
2.9 コマンド行引数と環境変数の操作機能....74
2.9.1 コマンド行引数の操作....75
2.9.1.1 引数の個数を求める方法....75
2.9.1.2 引数の値を参照する方法....75
2.9.2 環境変数の操作....75
2.9.2.1 環境変数の値を参照する方法....76
2.9.2.2 環境変数の値を更新する方法....76
2.10 報告書作成機能....76
2.10.1 報告書ファイル....77
2.10.2 特殊レジスタ....77
2.10.2.1 ページカウンタ....77
2.10.2.2 行カウンタ....77

第3章 見出し部とプログラム終わり見出し....78
3.1 見出し部の構成(IDENTIFICATION DIVISION)....78
3.1.1 プログラム名段落(PROGRAM-ID)....78
3.1.2 翻訳日付段落(DATE-COMPILED)....79
3.2 プログラム終わり見出し(END PROGRAM)....80

第4章 環境部....81
4.1 環境部の構成(ENVIRONMENT DIVISION)....81
4.2 構成節(CONFIGURATION SECTION)....81
4.2.1 翻訳用計算機段落(SOURCE-COMPUTER)....81
4.2.1.1 WITH DEBUGGING MODE句....82
4.2.2 実行用計算機段落(OBJECT-COMPUTER)....82
4.2.2.1 MEMORY SIZE句....83
4.2.2.2 PROGRAM COLLATING SEQUENCE句....83
4.2.3 特殊名段落(SPECIAL-NAMES)....84
4.2.3.1 機能名-1句....84
4.2.3.2 機能名-2句....87
4.2.3.3 機能名-3句....88
4.2.3.4 ALPHABET句....88
4.2.3.5 CLASS句....90
4.2.3.6 CRT STATUS句....91
4.2.3.7 CURRENCY SIGN句....92
4.2.3.8 CURSOR句....92
4.2.3.9 DECIMAL-POINT IS COMMA句....93
4.2.3.10 POSITIONING UNIT句....93
4.2.3.11 PRINTING MODE句....94
4.2.3.12 SYMBOLIC CHARACTERS句....96
4.2.3.13 SYMBOLIC CONSTANT句....96
4.3 入出力節(INPUT-OUTPUT SECTION)....97
4.3.1 ファイル管理段落(FILE-CONTROL)....97
4.3.1.1 ACCESS MODE句(順ファイル・相対ファイル・索引ファイル・表示ファイル・報告書作成)....107
4.3.1.2 ALTERNATE RECORD KEY句(索引ファイル)....108
4.3.1.3 ASSIGN句(順ファイル・相対ファイル・索引ファイル)....109
4.3.1.4 ASSIGN句(整列併合・報告書作成)....111
4.3.1.5 ASSIGN句(表示ファイル)....111
4.3.1.6 DESTINATION句(表示ファイル)....112
4.3.1.7 END KEY句(表示ファイル)....112
4.3.1.8 FILE STATUS句(順ファイル・相対ファイル・索引ファイル・表示ファイル・報告書作成)....113
4.3.1.9 FORMAT句(順ファイル・表示ファイル)....114
4.3.1.10 GROUP句(順ファイル・表示ファイル)....114
4.3.1.11 LOCK MODE句(順ファイル・相対ファイル・索引ファイル)....115

4.3.1.12 MESSAGE CLASS句(表示ファイル)....116
4.3.1.13 MESSAGE CODE句(表示ファイル)....117
4.3.1.14 MESSAGE MODE句(表示ファイル)....117
4.3.1.15 MESSAGE OWNER句(表示ファイル)....118
4.3.1.16 MESSAGE SEQUENCE句(表示ファイル)....118
4.3.1.17 ORGANIZATION句(順ファイル)....119
4.3.1.18 ORGANIZATION句(相対ファイル)....119
4.3.1.19 ORGANIZATION句(索引ファイル)....120
4.3.1.20 ORGANIZATION句(表示ファイル・報告書作成)....120
4.3.1.21 PADDING CHARACTER句(順ファイル)....120
4.3.1.22 PROCESSING CONTROL句(表示ファイル)....121
4.3.1.23 PROCESSING MODE句(表示ファイル)....121
4.3.1.24 PROCESSING TIME句(表示ファイル)....122
4.3.1.25 RECORD DELIMITER句(順ファイル)....122
4.3.1.26 RECORD KEY句(索引ファイル)....123
4.3.1.27 RESERVE句(順ファイル・相対ファイル・索引ファイル・報告書作成)....124
4.3.1.28 SELECT句(順ファイル・相対ファイル・索引ファイル・報告書作成)....124
4.3.1.29 SELECT句(整列併合・表示ファイル)....124
4.3.1.30 SELECTED FUNCTION句(表示ファイル)....125
4.3.1.31 SESSION CONTROL句(表示ファイル)....125
4.3.1.32 SYMBOLIC DESTINATION句(表示ファイル)....126
4.3.1.33 UNIT CONTROL句(表示ファイル)....127
4.3.2 入出力管理段落(I-O-CONTROL)....127
4.3.2.1 APPLY MULTICONVERSATION-MODE句(表示ファイル)....128
4.3.2.2 APPLY SAVED-AREA句(表示ファイル)....129
4.3.2.3 MULTIPLE FILE TAPE句(順ファイル・報告書作成)....129
4.3.2.4 RERUN句(順ファイル・相対ファイル・索引ファイル)....129
4.3.2.5 SAME句(順ファイル・相対ファイル・索引ファイル・整列併合・表示ファイル・報告書作成)....130

第5章 データ部....132
5.1 データ部の構成(DATA DIVISION)....132
5.2 ファイル記述項....135
5.2.1 BLOCK CONTAINS句(順ファイル・相対ファイル・索引ファイル・報告書作成)....136
5.2.2 CODE-SET句(順ファイル・報告書作成)....137
5.2.3 CONTROL RECORDS句(順ファイル)....137
5.2.4 DATA RECORDS句(順ファイル・相対ファイル・索引ファイル)....137
5.2.5 EXTERNAL句(順ファイル・相対ファイル・索引ファイル・表示ファイル・報告書作成)....138
5.2.6 GLOBAL句(順ファイル・相対ファイル・索引ファイル・表示ファイル・報告書作成)....138
5.2.7 LABEL RECORDS句(順ファイル・相対ファイル・索引ファイル・報告書作成)....139
5.2.8 LINAGE句(順ファイル)....139
5.2.9 RECORD句(順ファイル・相対ファイル・索引ファイル)....141
5.2.10 RECORD句(表示ファイル)....143
5.2.11 RECORD句(報告書作成)....144
5.2.12 REPORT句(報告書作成)....144
5.2.13 VALUE OF句(順ファイル・相対ファイル・索引ファイル・報告書作成)....145
5.3 整列併合用ファイル記述項....145
5.4 データ記述項....146
5.4.1 BLANK WHEN ZERO句....149
5.4.2 CHARACTER TYPE句....149
5.4.3 EXTERNAL句....153
5.4.4 GLOBAL句....154
5.4.5 JUSTIFIED句....155
5.4.6 OCCURS句....156
5.4.7 PICTURE句....158
5.4.8 PRINTING POSITION句....167
5.4.9 REDEFINES句....167
5.4.10 RENAMES句....169

5.4.11 SIGN句....170
5.4.12 SYNCHRONIZED句....171
5.4.13 TYPE句....172
5.4.14 TYPEDEF句....173
5.4.15 USAGE句....174
5.4.16 VALUE句....180
5.4.17 BASED ON句....182
5.5 画面データ記述項....183
5.5.1 AUTO句....185
5.5.2 BACKGROUND-COLOR句....186
5.5.3 BELL句....187
5.5.4 BLANK LINE句....187
5.5.5 BLANK SCREEN句....187
5.5.6 BLANK WHEN ZERO句....188
5.5.7 BLINK句....188
5.5.8 COLUMN NUMBER句....189
5.5.9 ERASE句....189
5.5.10 FOREGROUND-COLOR句....190
5.5.11 FULL句....191
5.5.12 HIGHLIGHT句....191
5.5.13 JUSTIFIED句....192
5.5.14 LINE NUMBER句....192
5.5.15 LOWLIGHT句....193
5.5.16 PICTURE句....194
5.5.17 REQUIRED句....195
5.5.18 REVERSE-VIDEO句....195
5.5.19 SECURE句....196
5.5.20 SIGN句....196
5.5.21 UNDERLINE句....196
5.5.22 USAGE句....197
5.5.23 VALUE句....197
5.6 報告書記述項....198
5.6.1 CODE句....199
5.6.2 CONTROL句....200
5.6.3 PAGE句....200
5.7 報告集団記述項....203
5.7.1 COLUMN NUMBER句....206
5.7.2 GROUP INDICATE句....206
5.7.3 LINE NUMBER句....207
5.7.4 NEXT GROUP句....208
5.7.5 SIGN句....208
5.7.6 SOURCE句....209
5.7.7 SUM句....210
5.7.8 TYPE句....211
5.7.9 USAGE句....215
5.7.10 VALUE句....215
5.8 報告集団の表示規則....216
5.8.1 表示規則表の見かた....216
5.8.2 報告書頭書き報告集団の表示規則....217
5.8.3 ページ頭書き報告集団の表示規則....219
5.8.4 本体集団の表示規則....219
5.8.5 ページ脚書き報告集団の表示規則....222
5.8.6 報告書脚書き報告集団の表示規則....223

第6章 手続き部....225
6.1 手続き部の構成（PROCEDURE DIVISION）....225
6.2 手続き部の見出し....229

6.3 文に関する共通の規則....231
6.3.1 算術式....231
6.3.2 ブール式....233
6.3.3 条件式....234
6.3.3.1 比較条件....234
6.3.3.2 字類条件....236
6.3.3.3 条件名条件....237
6.3.3.4 スイッチ状態条件....238
6.3.3.5 正負条件....238
6.3.3.6 複合条件....239
6.3.3.7 組合せ比較条件の略記法....244
6.3.4 比較の規則....245
6.3.5 転記の規則....249
6.3.6 算術文....252
6.3.7 算術文における複数個の答....252
6.3.8 ROUNDED指定....252
6.3.9 ON SIZE ERROR指定....253
6.3.10 CORRESPONDING指定....254
6.3.11 作用対象の重なり....254
6.3.12 INVALID KEY指定....254
6.3.13 AT END指定....255
6.3.14 矛盾するデータ....256
6.4 文....256
6.4.1 ACCEPT文（中核）....257
6.4.2 ACCEPT文（スクリーン操作）....258
6.4.3 ACCEPT文（コマンド行引数と環境変数の操作）....259
6.4.4 ADD文（中核）....260
6.4.5 ALTER文（中核）....262
6.4.6 CALL文（プログラム間連絡）....262
6.4.7 CANCEL文（プログラム間連絡）....267
6.4.8 CLOSE文（順ファイル・相対ファイル・索引ファイル・表示ファイル・報告書作成）....268
6.4.9 COMPUTE文（中核）....271
6.4.10 CONTINUE文（中核）....272
6.4.11 DELETE文（相対ファイル・索引ファイル）....272
6.4.12 DISPLAY文（中核）....274
6.4.13 DISPLAY文（スクリーン操作）....275
6.4.14 DISPLAY文（コマンド行引数と環境変数の操作）....276
6.4.15 DIVIDE文（中核）....278
6.4.16 ENTRY文（プログラム間連絡）....280
6.4.17 EVALUATE文（中核）....281
6.4.18 EXIT文（中核）....284
6.4.19 EXIT PERFORM文（中核）....285
6.4.20 EXIT PROGRAM文（プログラム間連絡）....286
6.4.21 GENERATE文（報告書作成）....286
6.4.22 GO TO文（中核）....288
6.4.23 IF文（中核）....288
6.4.24 INITIALIZE文（中核）....290
6.4.25 INITIATE文（報告書作成）....292
6.4.26 INSPECT文....292
6.4.27 MERGE文（整列併合）....300
6.4.28 MOVE文（中核）....304
6.4.29 MULTIPLY文（中核）....305
6.4.30 OPEN文（順ファイル・相対ファイル・索引ファイル）....306
6.4.31 OPEN文（表示ファイル）....310
6.4.32 OPEN文（報告書作成）....311
6.4.33 PERFORM文（中核）....311
6.4.34 READ文（順ファイル・相対ファイル・索引ファイル）....319

6.4.35 READ文(表示ファイル)....325
6.4.36 RELEASE文(整列併合)....327
6.4.37 RETURN文(整列併合)....327
6.4.38 REWRITE文(順ファイル・相対ファイル・索引ファイル)....328
6.4.39 SEARCH文(中核)....331
6.4.40 SET文(中核)....335
6.4.41 SORT文(整列併合)....337
6.4.42 START文(相対ファイル)....341
6.4.43 START文(索引ファイル)....343
6.4.44 STOP文(中核)....347
6.4.45 STRING文(中核)....348
6.4.46 SUBTRACT文(中核)....353
6.4.47 SUPPRESS文(報告書作成)....355
6.4.48 TERMINATE文(報告書作成)....355
6.4.49 UNLOCK文(順ファイル・相対ファイル・索引ファイル)....356
6.4.50 UNSTRING文(中核)....356
6.4.51 USE文(順ファイル・相対ファイル・索引ファイル・表示ファイル・報告書作成)....365
6.4.52 USE BEFORE REPORTING文(報告書作成)....367
6.4.53 USE FOR DEAD-LOCK文....368
6.4.54 WRITE文(順ファイル)....369
6.4.55 WRITE文(相対ファイル・索引ファイル)....373
6.4.56 WRITE文(表示ファイル)....376
6.5 関数の全般規則....377
6.5.1 関数の呼出し形式....377
6.5.2 引数の型....377
6.5.3 引数に表を指定する場合の規則....378
6.5.4 関数の型....378
6.6 関数....379
6.6.1 ACOS関数....379
6.6.2 ADDR関数....380
6.6.3 ANNUITY関数....380
6.6.4 ASIN関数....381
6.6.5 ATAN関数....382
6.6.6 CAST-ALPHANUMERIC関数....382
6.6.7 CHAR関数....382
6.6.8 COS関数....383
6.6.9 CURRENT-DATE関数....383
6.6.10 DATE-OF-INTEGER関数....384
6.6.11 DAY-OF-INTEGER関数....385
6.6.12 DISPLAY-OF関数....385
6.6.13 FACTORIAL関数....386
6.6.14 INTEGER関数....386
6.6.15 INTEGER-OF-DATE関数....387
6.6.16 INTEGER-OF-DAY関数....387
6.6.17 INTEGER-PART関数....388
6.6.18 LENG関数....388
6.6.19 LENGTH関数....389
6.6.20 LOG関数....390
6.6.21 LOG10関数....390
6.6.22 LOWER-CASE関数....391
6.6.23 MAX関数....391
6.6.24 MEAN関数....392
6.6.25 MEDIAN関数....392
6.6.26 MIDRANGE関数....393
6.6.27 MIN関数....393
6.6.28 MOD関数....394
6.6.29 NATIONAL関数....394

6.6.30 NATIONAL-OF関数....395
6.6.31 NUMVAL関数....396
6.6.32 NUMVAL-C関数....396
6.6.33 ORD関数....397
6.6.34 ORD-MAX関数....398
6.6.35 ORD-MIN関数....398
6.6.36 PRESENT-VALUE関数....399
6.6.37 RANDOM関数....399
6.6.38 RANGE関数....400
6.6.39 REM関数....401
6.6.40 REVERSE関数....401
6.6.41 SIN関数....402
6.6.42 SQRT関数....402
6.6.43 STANDARD-DEVIATION関数....403
6.6.44 STORED-CHAR-LENGTH関数....403
6.6.45 SUM関数....404
6.6.46 TAN関数....404
6.6.47 UCS2-OF関数....405
6.6.48 UPPER-CASE関数....405
6.6.49 UTF8-OF関数....406
6.6.50 VARIANCE関数....406
6.6.51 WHEN-COMPILED関数....407

第7章 原始文操作....409
7.1 COPY文....409
7.2 REPLACE文....414

第8章 データベース(SQL)....416
8.1 埋込みSQLの正書法....416
8.1.1 全般規定....416
8.1.2 行のつなぎ....416
8.1.3 COBOLの注記行および行内注記....416
8.2 データ部....417
8.2.1 埋込みSQL宣言節....417
8.2.2 ホスト変数定義....417
8.2.3 ホスト変数の参照....419
8.2.4 SQLSTATE/SQLCODE....420
8.2.5 SQLMSG....420
8.2.6 SQLERRD....420
8.3 手続き部....421
8.3.1 文字....422
8.3.2 定数....422
8.3.2.1 数定数....422
8.3.2.2 文字列定数....423
8.3.2.3 各国語文字列定数....423
8.3.3 トークン....424
8.3.4 名前....425
8.3.5 値指定と相手指定....425
8.3.6 列指定....427
8.3.7 集合関数指定....427
8.3.8 値式....428
8.3.9 述語....429
8.3.9.1 比較述語....429
8.3.9.2 BETWEEN述語....430
8.3.9.3 IN述語....430
8.3.9.4 LIKE述語....431
8.3.9.5 NULL述語....431
8.3.9.6 限定述語....431

8.3.9.7 EXISTS述語....432
8.3.10 探索条件....432
8.3.11 表式....432
8.3.11.1 FROM句....433
8.3.11.2 WHERE句....433
8.3.11.3 GROUP BY句....433
8.3.11.4 HAVING句....434
8.3.12 問合せ指定....434
8.3.13 問合せ式....434
8.3.14 副問合せ....435
8.3.15 FOR句....435
8.4 埋込み例外宣言....436
8.5 非カーソル系データ操作文....436
8.5.1 SELECT文....436
8.5.2 DELETE文(探索)....437
8.5.3 INSERT文....438
8.5.4 UPDATE文(探索)....439
8.6 カーソル系データ操作文....440
8.6.1 カーソル宣言....440
8.6.2 OPEN文....441
8.6.3 CLOSE文....441
8.6.4 FETCH文....442
8.6.5 DELETE文(位置付け)....443
8.6.6 UPDATE文(位置付け)....443
8.7 動的SQL....444
8.7.1 INTO句/USING句....444
8.7.2 PREPARE文....444
8.7.3 EXECUTE文....445
8.7.4 EXECUTE IMMEDIATE文....446
8.7.5 動的SELECT文....447
8.7.6 動的カーソル宣言....447
8.7.7 動的OPEN文....447
8.7.8 動的CLOSE文....448
8.7.9 動的FETCH文....448
8.7.10 動的DELETE文(位置付け)....449
8.7.11 動的UPDATE文(位置付け)....449
8.8 セション制御文....450
8.8.1 COMMIT文....450
8.8.2 ROLLBACK文....450
8.9 コネクション制御文....450
8.9.1 CONNECT文....450
8.9.2 SET CONNECTION文....451
8.9.3 DISCONNECT文....452
8.10 ストアドプロシージャ....452
8.10.1 CALL文....452

第9章 通信データベース....454
9.1 埋込みDCSQLの基本要素....454
9.1.1 使用できる文字....455
9.1.2 引用符、キーワードおよび分離符号....455
9.1.3 通信データベース名....455
9.1.4 サービス名....455
9.1.5 テーブル名....455
9.1.6 ホスト変数名....456
9.1.7 定数....456
9.1.7.1 文字列定数....456
9.1.7.2 漢字列定数....456

9.1.7.3 真数定数....457
9.1.7.4 概数定数....457
9.2 埋込みDCSQLの正書法....458
9.2.1 記述の全般規定....458
9.2.2 行のつなぎ....458
9.2.3 COBOLの注記行および行内注記....458
9.2.4 埋込みDCSQLの注釈....458
9.3 埋込みDCSQL....458
9.3.1 データ部....458
9.3.1.1 埋込みDCSQL宣言節....458
9.3.1.2 ホスト変数定義....459
9.3.1.3 ホスト変数の参照....460
9.3.1.4 DCSQLSTATE....460
9.3.1.5 DCSQLMSG....461
9.3.2 手続き部....461

第10章 Micro Focus固有機能....462
10.1 名前付き定数....462
10.1.1 利用者語....462
10.1.2 名前の範囲....462
10.1.3 データ記述項....462
10.2 16進数字定数....463
10.3 スクリーン操作機能....463
10.3.1 環境部....464
10.3.1.1 特殊名段落....464
10.3.1.1.1 CONSOLE IS CRT句....464
10.3.2 データ部....464
10.3.2.1 画面データ記述項....464
10.3.2.1.1 COLUMN NUMBER句....467
10.3.2.1.2 GRID句....467
10.3.2.1.3 LEFTLINE句....467
10.3.2.1.4 LINE NUMBER句....468
10.3.2.1.5 OCCURS句....468
10.3.2.1.6 OVERLINE句....469
10.3.2.1.7 PROMPT句....469
10.3.2.1.8 SIZE句....469
10.3.2.1.9 ZERO-FILL句....470
10.3.3 手続き部....470
10.3.3.1 ACCEPT文（スクリーン操作）....470
10.3.3.2 DISPLAY文（スクリーン操作）....473
10.4 日本語機能....475
10.4.1 手続き部....475
10.4.1.1 字類条件....475
10.5 入出力文....476
10.5.1 手続き部....476
10.5.1.1 READ文（相対ファイル・索引ファイル）....476
10.5.1.2 START文（相対ファイル）....480
10.5.1.3 START文（索引ファイル）....481
10.6 プログラム間連絡機能....483
10.6.1 手続き部....483
10.6.1.1 CALL文（プログラム間連絡）....483

第11章 .NETプログラミング機能....485
11.1 .NETとは....485
11.2 用語の説明....485
11.2.1 オブジェクト指向プログラミング機能の用語....485
11.2.2 .NETプログラミング機能の用語....487
11.3 全般規定....490

11.3.1 言語の基本要素....490
11.3.1.1 COBOLの語....490
11.3.1.1.1 利用者語....490
11.3.1.1.2 予約語....491
11.3.1.2 表意定数....492
11.3.2 データ記述の概念....492
11.3.3 一意参照....492
11.3.3.1 一意名....492
11.3.3.2 メソッドの行内呼出し....494
11.3.3.3 オブジェクト指定子....495
11.3.3.4 定義済みオブジェクト一意名....496
11.3.3.4.1 EXCEPTION-OBJECT....496
11.3.3.4.2 NULL....497
11.3.3.4.3 SELFとSUPER....497
11.3.3.5 オブジェクトプロパティ....498
11.3.3.6 TYPE OF特殊レジスタ....498
11.3.3.7 フィールド参照....499
11.3.4 外部名と内部名....499
11.3.4.1 外部名の表記の規則....500
11.3.4.1.1 名前空間や他の型の入れ子となっている場合の修飾....500
11.3.4.1.2 型が配列型であることを示すための修飾....500
11.3.4.1.3 型やメソッドが型パラメタを持つことを示すための修飾....501
11.3.5 正書法....501
11.3.5.1 行のつなぎ....501
11.3.5.2 名前の範囲....501
11.4 プログラム構造....502
11.4.1 プログラムの構成....502
11.4.1.1 COBOL翻訳集団....502
11.4.2 継承....506
11.4.3 総称性....506
11.5 見出し部と終わり見出し....507
11.5.1 見出し部の構成....507
11.5.1.1 プログラム名段落(PROGRAM-ID)....507
11.5.1.2 クラス名段落(CLASS-ID)....509
11.5.1.3 スタティック段落(STATIC)....510
11.5.1.4 オブジェクト段落(OBJECT)....510
11.5.1.5 メソッド名段落(METHOD-ID)....510
11.5.1.6 デリゲート名段落(DELEGATE-ID)....514
11.5.1.7 インタフェース名段落(INTERFACE-ID)....514
11.5.1.8 ENUM名段落(ENUM-ID)....515
11.5.1.9 CUSTOM-ATTRIBUTE句....515
11.5.2 終わり見出し....516
11.6 環境部....517
11.6.1 構成節(CONFIGURATION SECTION)....517
11.6.1.1 特殊名段落(SPECIAL-NAMES)....518
11.6.1.2 CUSTOM-ATTRIBUTE句....518
11.6.1.3 リポジトリ段落(REPOSITORY)....520
11.6.2 入出力節(INPUT-OUTPUT SECTION)....524
11.6.2.1 ファイル管理段落(FILE-CONTROL)....524
11.6.2.1.1 ファイル管理記述項....524
11.6.2.1.2 SYMBOLIC DESTINATION句....525
11.6.2.2 入出力管理段落(I-O-CONTROL)....525
11.7 データ部....525
11.7.1 ファイル節(FILE SECTION)....525
11.7.2 連絡節(LINKAGE SECTION)....526
11.7.3 データ記述項....526
11.7.3.1 CHARACTER TYPE句....528

11.7.3.2 CUSTOM-ATTRIBUTE句....528
11.7.3.3 EXTERNAL句....529
11.7.3.4 GLOBAL句....529
11.7.3.5 OCCURS句....530
11.7.3.6 PRINTING POSITION句....530
11.7.3.7 PROPERTY句....530
11.7.3.8 REDEFINES句....532
11.7.3.9 USAGE句....532
11.7.3.10 VALUE句....535
11.7.3.11 アクセス指定句(PUBLIC/PRIVATE/PROTECTED)....536
11.8 手続き部....537
11.8.1 手続き部の見出し....537
11.8.2 文に関する共通の規則....539
11.8.2.1 呼出し演算子....539
11.8.2.2 メソッドおよびオブジェクトの状態....539
11.8.2.2.1 メソッドの状態....539
11.8.2.2.2 オブジェクトにだけ関連付けられたデータの状態....540
11.8.2.3 制御の明示移行と暗黙移行....540
11.8.2.4 条件式....540
11.8.2.4.1 字類条件....540
11.8.2.4.2 型条件....542
11.8.2.5 比較の規則....542
11.8.2.6 適合....546
11.8.2.6.1 代入時の適合....546
11.8.2.6.2 引数の適合....552
11.8.2.6.3 引数のより適切な適合....567
11.8.2.7 例外条件....568
11.8.3 文....568
11.8.3.1 条件文と無条件文....569
11.8.3.2 ALTER文....569
11.8.3.3 CALL文....569
11.8.3.4 ENTRY文....571
11.8.3.5 EVALUATE文....571
11.8.3.6 EXIT文....571
11.8.3.7 GO TO文....572
11.8.3.8 INITIALIZE文....573
11.8.3.9 INVOKE文....573
11.8.3.10 MERGE文....576
11.8.3.11 OPEN文....576
11.8.3.12 PERFORM文....577
11.8.3.13 RAISE文....578
11.8.3.14 RESUME文....578
11.8.3.15 SET文....579
11.8.3.16 SORT文....580
11.8.3.17 STOP文....580
11.8.3.18 TRY文....581
11.8.3.19 USE文....582
11.8.3.20 WRITE文(順ファイル)....583
11.9 データベース(SQL)....583
11.9.1 埋込みSQLの正書法....583
11.9.2 データ部....583
11.9.2.1 埋込みSQL宣言節....583
11.9.3 埋込み例外宣言....583
11.9.4 カーソル系データ操作文....584
11.9.4.1 カーソル宣言....584
11.9.5 動的SQL....584
11.9.5.1 EXECUTE文....584

11.9.5.2 動的カーソル宣言....584
11.10 組込み関数....585
11.10.1 組込み関数の全般規則....585
11.10.1.1 概念....585
11.10.1.2 関数の定義....585
11.10.1.3 引数の型....585
11.10.1.4 関数の型....586
11.10.2 組込み関数....587
11.10.2.1 ACP-OF関数....587
11.10.2.2 DISPLAY-OF関数....587
11.10.2.3 ENUM-AND関数....588
11.10.2.4 ENUM-NOT関数....588
11.10.2.5 ENUM-OR関数....589
11.10.2.6 NATIONAL-OF関数....589
11.10.2.7 UNICODE-OF関数....590

付録A 予約語一覧....591

付録B システムの定量制限....608
B.1 正書法....608
B.2 中核のデータ部....608
B.3 中核の手続き部....608
B.4 順ファイル....609
B.5 相対ファイル....609
B.6 索引ファイル....609
B.7 プログラム間連絡....609
B.8 整列併合....609
B.9 原始文操作....610
B.10 表示ファイル....610
B.11 オブジェクト指向プログラミング....610

付録C コード表....611
C.1 EBCDICにおける文字の内部コード....611
C.2 ASCIIにおける文字の内部コード....612
C.3 JIS8単位コードにおける文字の内部コード....613

付録D 中間結果....614
D.1 中間結果の属性と精度....614
D.2 四則演算の中間結果....614
D.2.1 固定小数点属性の四則演算の中間結果の精度....614
D.2.2 浮動小数点属性の四則演算の中間結果の精度....618
D.3 べき乗の中間結果....618
D.3.1 固定小数点属性のべき乗の中間結果の精度....618
D.3.2 浮動小数点属性のべき乗の中間結果の精度....619
D.4 関数値の属性と精度....619

付録E 機能差....621
E.1 自由形式の正書法....621
E.2 順ファイル....621
E.3 データ項目定義....621
E.4 定数....621
E.5 表示ファイル....622
E.6 プログラム間連絡....622
E.7 文....623
E.8 関数....623
E.9 データベース....623
E.10 通信データベース....624
E.11 Micro Focus固有機能....624

E.12 オブジェクト指向プログラミング機能....624
E.13 .NETプログラミング機能....624
E.14 システムの定量制限....624

付録F 制御レコード....625
F.1 I制御レコード....625
F.2 S制御レコード....629

付録G 型を使用したデータ項目の定義....631
G.1 型....631
G.1.1 基本項目の型....631
G.1.2 集団項目の型....632
G.2 強い型....633
G.3 他の型を参照する型....635
G.4 型を使用したデータ定義の有効な使い方....637

付録H サンプル集....638
H.1 連結式....638
H.2 修飾....638
H.3 添字付け....639
H.4 部分参照....641
H.5 ポインタ付け....642
H.6 条件名の一意参照....643
H.7 行のつなぎ....644
H.8 自由形式の正書法....644
H.9 PROGRAM COLLATING SEQUENCE句....645
H.10 機能名-1句....646
H.11 機能名-2句....647
H.12 機能名-3句....648
H.13 ALPHABET句....649
H.14 CLASS句....650
H.15 CURRENCY SIGN句....650
H.16 DECIMAL-POINT IS COMMA句....651
H.17 POSITIONING UNIT句....652
H.18 PRINTING MODE句....653
H.19 SYMBOLIC CHARACTERS句....654
H.20 SYMBOLIC CONSTANT句....654
H.21 RENAMES句....655
H.22 VALUE句....656
H.23 BLANK WHEN ZERO句....657
H.24 GLOBAL句....657
H.25 JUSTIFIED句....659
H.26 OCCURS句....659
H.27 PICTURE句....661
H.28 REDEFINES句....662
H.29 SIGN句....662
H.30 TYPEDEF句....663
H.31 TYPE句....665
H.32 BASED ON句....666
H.33 ブール式....667
H.34 字類条件....668
H.35 組合せ比較条件の略記法....669
H.36 COMPUTE文....670
H.37 INITIALIZE文....671
H.38 INSPECT文....672
H.39 MOVE文....673
H.40 SEARCH文....674
H.41 STRING文....676

H.42 UNSTRING文.....677
H.43 USE文.....678
H.44 SIN関数.....679
H.45 COS関数.....680
H.46 TAN関数.....681
H.47 ADDR関数.....682
H.48 LENG関数.....683
H.49 NATIONAL関数.....685
H.50 CURRENT-DATE関数.....685
H.51 SUM関数.....686
H.52 REM関数.....688
H.53 INTEGER-OF-DATE関数.....688
H.54 DATE-OF-INTEGER関数.....689
H.55 LOWER-CASE関数.....690
H.56 UPPER-CASE関数.....690
H.57 MAX関数.....691
H.58 MIN関数.....692
H.59 REVERSE関数.....692
H.60 STORED-CHAR-LENGTH関数.....693
H.61 UCS2-OF関数.....693
H.62 UTF8-OF関数.....694
H.63 WHEN-COMPILED関数.....695
H.64 COPY文.....696

索引.....698

第1章 全般規定

本章では、COBOLの言語の概念と全般規定を説明します。

1.1 文字と文字集合

COBOLプログラムは、計算機で使うことができる文字集合のうちの、特定の文字集合を使って書きます。COBOLプログラムを書くための文字集合を「COBOL文字集合」といいます。COBOL文字集合には、以下の4種類があります。

- 英字
- 数字
- 特殊文字
- 日本語文字

英字

英字を以下に示します。1文字の英字の記憶領域は1バイトです。

- 英大文字(A、B、C、…、Zの26文字)
- 英小文字(a、b、c、…、zの26文字)
- 空白

英小文字は、文字定数の中を除いて、対応する英大文字と等価とみなされます。

数字

数字は、以下の10文字です。1文字の数字の記憶領域は1バイトです。

```
0、1、2、3、4、5、6、7、8、9
```

特殊文字

特殊文字を下表に示します。1文字の特殊文字の記憶領域は1バイトです。

文字	意味
+	正号
-	負号またはハイフン
*	星印
/	斜線
=	等号
¥	通貨記号
,	コンマ
;	セミコロン
.	終止符または小数点
"	引用符
(	左括弧
)	右括弧
>	より大きい記号
<	より小さい記号
:	コロン

文字	意味
&	アンパサンド
_	アンダースコア

アンダースコアは、[Win32][WinIPF][Winx64][Solaris][Linux][LinuxIPF][Linux64][.NET]固有機能です。

日本語文字

日本語文字を下表に示します。1文字の日本語文字の記憶領域は複数バイトです。

文字	分類	備考
亜、唖、…、腕	JIS 第1水準漢字	JIS 漢字符号系(JIS X0208-1990)の漢字6349文字のうち、基本的な漢字2965文字
弌、丐、…、龠	JIS 第2水準漢字	JIS 漢字符号系の漢字6349文字のうちJIS 第1水準漢字以外の漢字3384文字
0、1、…、9	JIS 非漢字の数字	JIS 漢字符号系の非漢字の数字10文字
A、B、…、Z	JIS 非漢字の英大文字	JIS 漢字符号系の非漢字の英大文字26文字
a、b、…、z	JIS 非漢字の英小文字	JIS 漢字符号系の非漢字の英小文字26文字
ぁ、あ、ぃ、い、…、ん	JIS 非漢字のひらがな	JIS 漢字符号系の非漢字のひらがな(半濁音、濁音、拗音、促音を含む)83文字
ァ、ア、ィ、イ、…、ン	JIS 非漢字のカタカナ	JIS 漢字符号系の非漢字のカタカナ(半濁音、濁音、拗音、促音を含む)86文字
ー	JIS 非漢字の長音記号	
-	JIS 非漢字のハイフン	
―	JIS 非漢字の負号	
々	JIS 非漢字の繰返し記号	

計算機文字集合

計算機で使うことができる文字集合を、「計算機文字集合」といいます。COBOL文字集合は、計算機文字集合の部分集合です。
以下の場所では、COBOL文字集合中の文字だけでなく、計算機文字集合中の文字を使うことができます。

- 文字定数
- 日本語文字定数
- 注記項
- 注記行
- 行内注記

1.2 言語の基本要素

COBOLプログラムは、以下で構成します。

- 分離符
- COBOLの語
- 定数
- PICTURE句の文字列
- 注記項

・ 注記行

・ 行内注記

これらの要素のうち定数の値、注記項、注記行および行内注記を除くものは、COBOL文字集合の文字を使って書きます。定数の値、注記項、注記行および行内注記は、計算機文字集合中の任意の文字を使って書くことができます。
COBOLプログラムの中には、上記以外に、空白だけからなる行(空白行)を書くことができます。

1.2.1 分離符

分離符は、文字列を区切るための文字です。分離符として使える文字は、以下のものです。

・ 1つ以上の空白の並び
これを「分離符の空白」といいます。

・ 1つのコンマ“,”の後に1つ以上の空白の並びをつないだもの
これを「分離符のコンマ」といいます。

・ 1つのセミコロン“;”の後に1つ以上の空白の並びをつないだもの
これを「分離符のセミコロン」といいます。

・ 1つの終止符“.”の後に1つ以上の空白の並びをつないだもの
これを「分離符の終止符」といいます。

・ 1つの左括弧“(”

・ 1つの右括弧“)”

・ 定数値の開始を示す以下の引用符

　― “””

　― “X””、“NC””、“N””、“NX””および“B””

・ 定数値の終了を示す引用符“””

・ 1つの仮原文区切り記号“＝＝”

・ 1つのコロン“:”

・ 1つのポインタ修飾記号“－＞”

・ 1つの連結演算子“＆”

分離符の空白

分離符の空白は、COBOLの語、定数またはPICTURE句の文字列を区切るために使います。

分離符のコンマおよび分離符のセミコロン

分離符のコンマおよび分離符のセミコロンは、プログラムを読みやすくするために使います。これらの分離符が書ける場所は、分離符の空白が書ける場所と同じです。
これらの分離符の前後には、分離符の空白を書くこともできます。

分離符の終止符

分離符の終止符は、部、節および段落の見出しの終わり、または部、節、段落、記述項、完結文などの終わりを示すために使います。分離符の終止符は、【書き方】で“.”と示しているところにだけ書くことができます。
分離符の終止符の前後には、分離符の空白を書くこともできます。

左括弧と右括弧

左括弧と右括弧は、以下を囲むために使います。

・ 添字

・ 部分参照子

- 算術式
- ブール式
- 条件式
- 引数の並び

左括弧と右括弧は、これらを対にして使います。
左括弧の前後および右括弧の前後には、分離符の空白を書くこともできます。

引用符

引用符は、以下の値を示す文字列を囲むために使います。

- 文字定数
- 16進文字定数
- 日本語文字定数
- 日本語16進文字定数
- ブール定数

引用符は、定数値の開始を示す引用符と定数値の終了を示す引用符を対にして使います。引用符の記述規則を、以下に示します。

- 文字定数を書く場合、“”と“”を対にして使います。
- 16進文字定数を書く場合、“X”と“”を対にして使います。
- 日本語文字定数を書く場合、“NC”と“”を対にするか、または“N”と“”を対にして使います。
- 日本語16進文字定数を書く場合、“NX”と“”を対にして使います。
- ブール定数を書く場合、“B”と“”を対にして使います。

上記のいずれの場合も、定数値の開始を示す引用符の直前は、分離符の空白または左括弧でなければなりません。また、定数値の終了を示す引用符の直後は、分離符の空白、分離符のコンマ、分離符のセミコロン、分離符の終止符、または右括弧でなければなりません。

仮原文区切り記号

仮原文区切り記号は、仮原文を囲むために使います。仮原文区切り記号は、2つを対にして使います。左側の仮原文区切り記号の直前は、分離符の空白でなければなりません。右側の仮原文区切り記号の直後は、分離符の空白、分離符のコンマ、分離符のセミコロン、または分離符の終止符でなければなりません。

コロン

コロンは、部分参照子を書くために使います。
コロンの前後には、分離符の空白を書くこともできます。

ポインタ修飾記号

ポインタ修飾記号は、ポインタ修飾子を書くために使います。
ポインタ修飾記号の前後には、分離符の空白を書くこともできます。

連結演算子

連結演算子は、定数を連結するために使います。
連結演算子の前後には、分離符の空白を書かなければなりません。

分離符とみなされない文字列

以下の場所に書いた文字列には、分離符の規則は適用されません。すなわち、以下の場所に書いた文字列は、分離符と同じ形式であっても分離符とはみなされません。

- 数字定数

- 文字定数の中の、定数値の開始を示す分離符と定数値の終了を示す分離符で囲まれた部分
- 16進文字定数、日本語定数およびブール定数の中の、定数値の開始を示す分離符と定数値の終了を示す分離符で囲まれた部分
- PICTURE句の文字列
- 注記項
- 注記行
- 行内注記

1.2.2 COBOLの語

COBOLの語は、COBOL文字集合中の文字からなる、ひとつながりの文字列です。COBOLの語には、以下の4種類があります。

- 利用者語
- システム名
- 予約語
- 関数名

COBOLの語は、以下の規則を満足する文字列です。以下の規則のほかに、COBOLの語ごとに制限があります。

1. 長さは30文字以内です。
2. 以下で構成します。
   - 空白を除く英字(A～Zおよびa～z)
   - 数字(0～9)
   - ハイフン(－)
   - アンダースコア(_)

   英小文字は、対応する英大文字と等価であるとみなされます。利用者語の場合は、日本語文字で構成することができます。
3. 最初の文字と最後の文字は、以下であってはいけません。
   - ハイフン(－)
   - アンダースコア(_)

注意事項

名前の先頭の1文字がアンダースコアで始まる名前は、COBOLが内部的に予約しています。このような利用者語を使用するプログラムは、将来のバージョンで正しく動作しない場合があります。名前を変更するか、そのような名前を定義/参照するすべての資産を再翻訳して、翻訳エラーの発生しないことを確認する必要があります。

1.2.2.1 利用者語

利用者語は、利用者が任意に命名する語です。利用者語は、以下の20個です。

- 位置決め単位名
- 印字モード名
- 記号定数
- 記号文字
- 原文名
- 指標名

- 字類名
- 条件名
- 節名
- 段落名
- データ名
- 登録集名
- ファイル名
- 符号系名
- プログラム名
- 報告書名
- 呼び名
- レコード名
- レベル番号
- 型名

レベル番号以外の利用者語には、すべて異なる名前を付けなければなりません。ただし、条件名、データ名、レコード名および指標名は、修飾によって一意参照できるならば、同じ名前を付けることもできます。レベル番号については、1つのプログラムに、同じレベル番号を2つ以上書くことができます。

利用者語の記述規則

利用者語は、以下の組み合わせ、または日本語文字で構成します。

- 英字
- 数字
- ハイフン(ー)
- アンダースコア

レベル番号、登録集名、プログラム名および原文名以外の利用者語は、日本語文字で構成することもできます。ただし、[Win32][WinIPF][Winx64][Solaris][Linux][LinuxIPF][Linux64][.NET]では、プログラム名を日本語文字で構成することができます。日本語文字からなる利用者語を、特に「日本語利用者語」といいます。

利用者語を英字、数字、ハイフンおよびアンダースコアの組合せで構成する場合、以下の規則に従わなければなりません。

1. 30文字以内でなければなりません。
2. 空白を除く英字(A～Zおよびa～z)、数字(0～9)、ハイフン(ー)およびアンダースコア(_)で構成しなければなりません。英小文字は、対応する英大文字と等価であるとみなされます。
3. 最初の文字と最後の文字は、ハイフンおよびアンダースコアであってはいけません。
4. 利用者語を構成する文字列は、予約語と同じ文字列であってはいけません。しかし、関数名またはシステム名と同じ文字列を、利用者語として使うことができます。
5. 段落名、節名およびレベル番号以外のすべての利用者語は、少なくとも1文字の英字を含まなければなりません。
6. 利用者語に2つ以上の連続したアンダースコアを含んではなりません。

利用者語を日本語文字で構成する場合、以下の規則に従わなければなりません。

1. 30文字以内でなければなりません。
2. 空白を除く日本語文字で構成しなければなりません。JIS非漢字の英小文字は、対応するJIS非漢字の英大文字とは異なる文字であるとみなされます。

3. 最初の文字と最後の文字は、JIS非漢字のハイフン(-)またはJIS非漢字の負号(－)であってはいけません。
4. 少なくとも1文字のJIS第1水準漢字、JIS第2水準漢字、JIS非漢字のひらがな、JIS非漢字のカタカナまたはJIS非漢字の長音記号(ー)を含まなければなりません。
5. レベル番号、登録集名、プログラム名および原文名を、日本語文字で構成することはできません。ただし、[Win32][WinIPF][Winx64][Solaris][Linux][LinuxIPF][Linux64][.NET]では、プログラム名を日本語文字で構成することができます。

利用者語の用途

利用者語の用途および利用者語を書くことができる場所は、以下のとおりです。

位置決め単位名

位置決め単位名は、印字位置を示す値の単位に付ける名前です。位置決め単位名は、環境部の特殊名段落で定義します。位置決め単位名は、【書き方】で“位置決め単位名-n”と示しているところに書くことができます。

印字モード名

印字モード名は、印字する文字の表示形式(文字サイズ、文字ピッチ、文字書体、文字回転量、文字形態など)に付ける名前です。印字モード名は、環境部の特殊名段落で定義します。印字モード名は、【書き方】で“印字モード名-n”と示しているところに書くことができます。

記号定数

記号定数は、定数に付ける名前です。定数を名前で参照するために使います。記号定数は、環境部の特殊名段落で定義します。記号定数は、【書き方】で“定数-n”および“整数-n”と示しているところに書くことができます。

記号文字

記号文字は、文字に付ける名前です。特定の文字を使って表意定数を表すために使います。記号文字は、環境部の特殊名段落で定義します。記号文字は、【書き方】で“定数-n”と示しているところに書くことができます。

原文名

原文名は、登録集原文に付ける名前です。COBOL登録集の中の原文を参照するために使います。原文名は、【書き方】で“原文名-n”と示しているところに書くことができます。

指標名

指標名は、表の指標に付ける名前です。指標名は、データ部のデータ記述項で定義します。指標名は、【書き方】で“指標名-n”と示しているところに書くことができます。

字類名

字類名は、利用者が定めた文字集合に付ける名前です。データ項目の内容の字類を検査するために使います。字類名は、環境部の特殊名段落で定義します。字類名は、【書き方】で“字類名-n”と示しているところに書くことができます。

条件名

条件名には、以下の2つがあります。

1. データ項目の取ることができる値に付ける名前
2. 外部スイッチのオン状態またはオフ状態に付ける名前

1.の条件名は、データ部の条件名記述項で定義します。条件名を関係付けたデータ項目を、「条件変数」といいます。1.の条件名は、条件名条件およびSET文で使うことができます。条件名条件は、比較条件の省略した書き方として使います。条件名条件は、条件変数の値が条件名の値と等しいかどうかという条件を表します。1.の条件名は、条件変数に値を設定するために、SET文でも使うことができます。

2.の条件名は、環境部の特殊名段落で定義します。2.の条件名は、スイッチ状態条件で使うことができます。スイッチ状態条件は、外部スイッチの状態が、特殊名段落で指定した状態に設定されているかという条件を表します。外部スイッチの状態を設定するためには、SET文を使います。SET文では、条件名に対応付けた呼び名を指定します。

条件名は、【書き方】で“条件名-n”と示しているところに書くことができます。

節名

節名は、手続き部の節に付ける名前です。節名は、【書き方】で“節名-n”または“手続き名-n”と示しているところに書くことができます。

段落名

段落名は、手続き部の段落に付ける名前です。段落名は、【書き方】で“段落名-n”または“手続き名-n”と示しているところに書くことができます。

データ名

データ名は、データ項目に付ける名前です。データ名は、データ部のデータ記述項で定義します。データ名は、【書き方】で“データ名-n”または“一意名-n”と示しているところに書くことができます。
“データ名-n”と示している場合、特に許されない限り、以下をしてはいけません。

- 部分参照
- 添字付け
- 修飾
- 明のポインタ付け

“一意名-n”と示している場合、データ名を一意に参照するために、必要ならば、以下をしなければなりません。

- 部分参照
- 添字付け
- 修飾
- 明のポインタ付け

登録集名

登録集名は、COBOL登録集に付ける名前です。COBOL登録集を参照するために使います。

ファイル名

ファイル名は、ファイル結合子に付ける名前です。ファイル名は、環境部のファイル管理記述項で定義します。ファイル名は、【書き方】で“ファイル名-n”と示しているところに書くことができます。

符号系名

符号系名は、特定の文字の組および文字の大小順序に付ける名前です。符号系名は、環境部の特殊名段落で定義します。符号系名は、【書き方】で“符号系名-n”と示しているところに書くことができます。

プログラム名

プログラム名は、COBOL原始プログラムに付ける名前です。プログラム名は、見出し部のプログラム名段落で定義します。プログラム名は、【書き方】で“プログラム名-n”と示しているところに書くことができます。

報告書名

報告書名は、報告書に付ける名前です。報告書名は、データ部の報告書記述項で定義します。報告書名は、【書き方】で“報告書名-n”と示しているところに書くことができます。

呼び名

呼び名は、プログラムの外部で定義した機能を使うために、機能名(機能を識別するための名前)に関係付ける名前です。呼び名は、環境部の特殊名段落で定義します。呼び名は、【書き方】で“呼び名-n”と示しているところに書くことができます。

レコード名

レコード名は、レコードに付ける名前です。レコード名は、データ部のレコード記述項で定義します。レコード名は、【書き方】で“レコード名-n”と示しているところに書くことができます。

レベル番号

レベル番号は、データ項目、報告集団項目および画面項目の階層を表すために付ける番号です。レベル番号01～49は、レコードの階層構造における項目の位置付けを表すために使います。レベル番号66、77および88は、特別な

性質を持つデータ記述項を書くために使います。レベル番号01～09の上位のゼロは、省略することができます。レベル番号は、データ部の【書き方】で“レベル番号”と示しているところに書くことができます。

型名

型名は型に付ける名前です。型は、データ部で型宣言の旨の指定のあるデータ記述項で定義します。
なお、型名は、[Win32][WinIPF][Winx64][Solaris][Linux][LinuxIPF][Linux64][.NET]固有の機能です。

1.2.2.2 システム名

システム名は、オペレーティングシステムとの連絡に使うための語です。システム名には、利用者が任意に命名する語と特定の語があります。システム名は、以下の4つです。

- 機能名
- 計算機名
- 言語名
- ファイル識別名

機能名

機能名は、システムの論理的装置、印字ページの表現、外部スイッチなど、プログラムの外部で定義した機能を識別するための名前です。外部で定義した機能を使う場合、環境部の特殊名段落で、機能名を呼び名に対応付けなければなりません。
機能名は、使用する機能ごとに定められています。
機能名の記述規則は、“4.2.3 特殊名段落（SPECIAL-NAMES）”で説明します。

計算機名

計算機名は、プログラムを翻訳または実行する計算機を識別するための名前です。
計算機名は、利用者が命名します。計算機名は、COBOLの語の規則に従う任意の文字列で書くことができます。

言語名

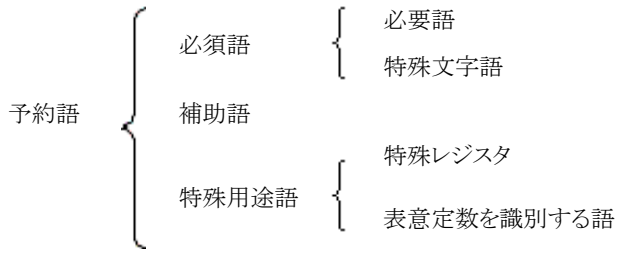
言語名は、特定のプログラム言語を識別するための名前です。

ファイル識別名

ファイル識別名は、記憶媒体上のファイルを識別するための名前です。ファイルを使う場合、環境部のファイル管理記述項で、ファイル識別名をファイル名に関係付けなければなりません。
ファイル識別名は、利用者が命名します。
ファイル識別名の記述規則は、“4.3.1.3 ASSIGN句（順ファイル・相対ファイル・索引ファイル）”～“4.3.1.5 ASSIGN句（表示ファイル）”で説明します。

1.2.2.3 予約語

予約語は、プログラムの処理を書くための特定の語です。予約語には、以下の種類があります。



予約語と同じ文字列を、利用者語またはシステム名として使うことはできません。
予約語の一覧については、“付録A 予約語一覧”を参照してください。

必須語

必須語は、必ず書かなければならない語です。必須語には、必要語と特殊文字語があります。

「必要語」は、【書き方】で、英大文字に下線を付けている語です。

「特殊文字語」は、以下の12個です。【書き方】では、他の記号との混同を避けるため、下線を付けていません。

+, -, *, /, **, <, >, =, >=, <=, &, ->

補助語

補助語は、書いても書かなくてもよい語です。【書き方】で、下線のない英大文字が補助語です。

特殊用途語

特殊用途語は、次の2種類とします。

- 特殊レジスタ
- 表意定数

特殊レジスタ

特殊レジスタは、コンパイラが自動的に作成する記憶領域です。特殊レジスタには、特定の機能を使ったときに発生する情報が格納されます。また、特定の機能を実行するプログラム(システム)に対して情報を通知するために使うこともできます。

特殊レジスタには、以下のものがあります。

- 順ファイル機能で使うことができる特殊レジスタ
  - LINAGE-COUNTER
- プログラム間連絡機能で使うことができる特殊レジスタ
  - PROGRAM-STATUS
  - RETURN-CODE (PROGRAM-STATUSと同義語)
- 整列併合機能で使うことができる特殊レジスタ
  - SORT-STATUS
  - SORT-CORE-SIZE
- 表示ファイル機能で使うことができる特殊レジスタ
  - EDIT-MODE
  - EDIT-OPTION
  - EDIT-OPTION2
  - EDIT-OPTION3
  - EDIT-COLOR
  - EDIT-STATUS
  - EDIT-CURSOR
- 報告書作成機能で使うことができる特殊レジスタ
  - LINE-COUNTER
  - PAGE-COUNTER

特殊レジスタは、特定の項類を持つデータ項目として扱われます。特殊レジスタは、【書き方】で“データ名-n”または“一意名-n”で示しているところで、特殊レジスタが持つ項類が許されるところに書くことができます。

なお、EDIT-OPTION2、EDIT-OPTION3は、[Win32][WinIPF][Winx64][Solaris][Linux][LinuxIPF][Linux64][.NET]固有の機能です。

表意定数を識別する語

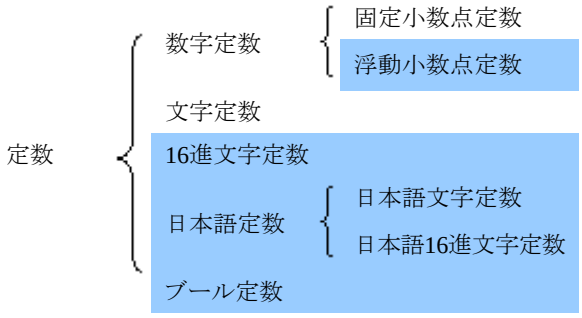
表意定数は、予約語を使って表現します。例えば、表意定数ALL SPACEの、ALLおよびSPACEは予約語です。

1.2.2.4 関数名

関数名は、関数の名前を示す語です。関数名は、関数一意名の中に書くことができます。関数一意名が書ける場所は、関数の型によって決められています。
関数一意名が書ける場所は、“6.5 関数の全般規則”で説明します。

1.2.3 定数

定数は、プログラムに書いた値を持つデータです。定数には、以下の種類があります。



1.2.3.1 数字定数

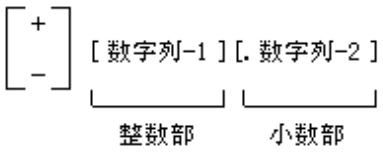
数字定数は、数値を値として持つ定数です。数字定数には、以下の2つがあります。

- 固定小数点定数
- 浮動小数点定数

固定小数点定数

固定小数点定数は、符号、数字および小数点を組み合わせて表現します。

【書き方】



---

1. 固定小数点定数は、符号、整数部、小数点および小数部で構成します。
2. “＋”は正号、“－”は負号、“.”は小数点を表します。
3. 数字列-1および数字列-2は、0～9の数字でなければなりません。
4. 整数部と小数部の桁数の合計は、以下でなければなりません。
   - [DS][HP][Win16]の場合、1～18桁

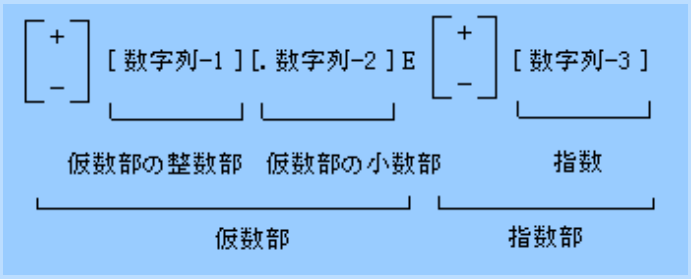
－ [Win32][WinIPF][Winx64][Solaris][Linux][LinuxIPF][Linux64][.NET]の場合、1～19桁

5. 固定小数点定数の値は、代数的な値です。
6. 固定小数点定数の項類は、数字です。
7. 固定小数点定数は、以下のところに書くことができます。
   a. 小数点を持つ固定小数点定数は、【書き方】で“定数-n”で示しているところで、数字定数が許されるところに書くことができます。
   b. 小数点を持たない固定小数点定数は、【書き方】で“定数-n”で示しているところで数字定数が許されるところ、および“整数-n”で示しているところに書くことができます。“整数-n”で示しているところには、構文規則で許されない限り、符号のない1以上の整数だけを書くことができます。

浮動小数点定数

浮動小数点定数は、“仮数＊(10＊＊指数)”の形式で表現します。

【書き方】



---

1. 浮動小数点定数は、仮数部、“E”および指数部で構成します。仮数部は、仮数部の符号、仮数部の整数部、小数点および仮数部の小数部で構成します。指数部は、指数部の符号および指数で構成します。
2. “＋”は正号、“－”は負号、“.”は小数点を表します。
3. 数字列-1～数字列-3は、0～9の数字でなければなりません。
4. 仮数部の整数部と小数部の桁数の合計は、1～16桁でなければなりません。
5. 指数の桁数は、1または2桁でなければなりません。
6. 浮動小数点定数の値は、以下の式で表される代数的な値です。

   仮数部 ＊ (10 ＊＊ 指数部)

7. 浮動小数点定数の項類は、数字です。
8. 浮動小数点定数は、【書き方】で“定数-n”で示しているところで、数字定数が許されるところに書くことができます。

1.2.3.2 文字定数

文字定数は、計算機文字集合の文字を値として持つ定数です。

【書き方】

"{文字-1} … "

---

1. 文字-1の並びは、左端と右端を分離符の引用符で区切らなければなりません。

2. 文字-1には、計算機文字集合中の任意の文字を書くことができます。
3. 文字-1に引用符を書く場合、1つの引用符を、連続する2つの引用符で表現しなければなりません。
4. 文字-1の個数は1～160個でなければなりません。文字-1の並びの長さは、1～160バイトでなければなりません。
5. 文字定数の値は、文字-1の並びに書いた文字です。文字定数を囲む分離符の引用符は、文字定数の値の一部ではありません。
6. 文字定数の項類は、英数字です。
7. 文字定数は、【書き方】で“定数-n”と示しているところで、文字定数が許されるところに書くことができます。

1.2.3.3 16進文字定数

16進文字定数は、計算機文字集合の文字を値として持つ定数です。16進文字定数は、定数の値を16進文字で表現します。

【書き方】

X"{16進文字-1} … "

---

1. 16進文字-1の並びは、左端を分離符の“X””で区切り、右端を分離符の引用符で区切らなければなりません。
2. 16進文字-1は、“0”～“9”または“A”～“F”でなければなりません。16進文字-1の連続する2文字で、計算機文字集合中の1つの文字コードまたは文字コードの一部を表します。
3. 16進文字-1の個数は2～320個でなければなりません。
4. 16進文字定数の値は、16進文字-1の並びで表される値(計算機文字集合の文字)です。16進文字定数は、文字定数と等価とみなされます。
5. 16進文字定数の項類は、英数字です。
6. 16進文字定数は、【書き方】で“定数-n”と示しているところで、文字定数が許されるところに書くことができます。本書で“文字定数”と示しているところは、16進文字定数も含みます。
7. 16進文字の内部ビット構成を、下表に示します。

16進文字	内部ビット構成
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101

16進文字	内部ビット構成
E	1110
F	1111

1.2.3.4 日本語定数

日本語定数は、日本語文字を値として持つ定数です。日本語定数には、日本語文字定数と日本語16進文字定数があります。

日本語文字定数

日本語文字定数は、定数の値を計算機文字集合の日本語文字で表現します。

【書き方1】

NC"{日本語文字-1} … "

【書き方2】

N"{日本語文字-1} … "

---

1. 書き方1と書き方2は等価です。
2. 日本語文字-1の並びは、左端を分離符の“NC””または分離符の“N””で区切り、右端を分離符の引用符で区切らなければなりません。
3. 日本語文字-1には、計算機文字集合中の任意の日本語文字を書くことができます。
   ただし、[Win32][WinIPF][Winx64][Solaris][Linux][LinuxIPF][Linux64][.NET]では、動作モードがUnicode(ISO/IEC 10646-1)の場合、計算機文字集合中の任意の文字を書くことができます。
4. 日本語文字-1の個数は1～80個でなければなりません。
5. 日本語文字定数の値は、日本語文字-1の並びに書いた日本語文字です。
6. 日本語文字定数の項類は、日本語です。
7. 日本語文字定数は、【書き方】で“定数-n”と示しているところで、日本語定数が許されるところに書くことができます。

日本語16進文字定数

日本語16進文字定数は、定数の値を16進文字で表現します。

【書き方】

NX"{16進文字-1} … "

---

1. 16進文字-1の並びは、左端を分離符の“NX””で区切り、右端を分離符の引用符で区切らなければなりません。
2. 16進文字-1は、“0”～“9”または“A”～“F”でなければなりません。16進文字-1の連続する2文字で、計算機文字集合中の1つの日本語文字コードの一部を表します。
3. 16進文字-1の個数は4～320個でなければなりません。

4. 日本語16進文字定数の値は、16進文字-1の並びで表される値(計算機文字集合の日本語文字)です。日本語16進文字定数は、日本語文字定数と等価とみなされます。
5. 日本語16進文字定数の項類は、日本語です。
6. 日本語16進文字定数は、【書き方】で“定数-n”と示しているところで、日本語定数が許されるところに書くことができます。

1.2.3.5 ブール定数

ブール定数は、ブール文字を値として持つ定数です。

【書き方】

B"{ブール文字-1} … "

---

1. ブール文字-1の並びは、左端を分離符の“B””で区切り、右端を分離符の引用符で区切らなければなりません。
2. ブール文字-1は、“0”または“1”でなければなりません。
3. ブール文字-1の個数は1～160個でなければなりません。
4. ブール定数の項類は、ブールです。
5. ブール定数の値は、ブール文字-1の並びに書いた値です。
6. ブール定数は、【書き方】で“定数-n”と示しているところで、ブール定数が許されるところに書くことができます。

1.2.4 表意定数

表意定数は、特定の値を持つ定数または定数の繰返しを書くときに使います。表意定数には、以下の7種類があります。

- ZERO
- SPACE
- HIGH-VALUE
- LOW-VALUE
- QUOTE
- ALL定数
- 記号文字

表意定数の書き方と値を、下表に示します。

名称	書き方	値
ZERO	［ALL］ZERO ［ALL］ZEROS ［ALL］ZEROES	文脈によって、以下のいずれかの値を持ちます。 ・ 数値ゼロ ・ 計算機文字集合中の文字“0”の繰返し ・ ブール文字“0”の繰返し
SPACE	［ALL］SPACE ［ALL］SPACES	文脈によって、以下のいずれかの値を持ちます。 ・ 計算機文字集合中の英字の空白の繰返 ・ 計算機文字集合中の日本語文字の空白の繰返し
HIGH-VALUE	［ALL］HIGH-VALUE	プログラムの文字の大小順序における最大の文字コードの繰返し

名称	書き方	値
	［ALL］HIGH-VALUES	
LOW-VALUE	［ALL］LOW-VALUE ［ALL］LOW-VALUES	プログラムの文字の大小順序における最小の文字コードの繰返し
QUOTE	［ALL］QUOTE ［ALL］QUOTES	引用符(")の繰返し
ALL 定数	ALL 定数	ALL の後に書いた定数の繰返し
記号文字	［ALL］記号文字	記号文字の繰返し

1. ALLを書く場合、ALLの後に分離符の空白を書かなければなりません。
2. SPACEとSPACESなど、単数形と複数形は同義語です。
3. 以下の値を囲むための引用符を、表意定数QUOTEで書くことはできません。
   - 文字定数
   - 16進文字定数
   - 日本語定数
   - ブール定数

   例えば、“ABD”をQUOTE ABD QUOTEと書くことはできません。
4. ALL定数に書く定数は、以下のいずれかでなければなりません。
   - 文字定数
   - 16進文字定数
   - 日本語定数
   - ブール定数
   - これらに対応付けた記号定数

   ALL定数の中に表意定数を書くことはできません。
5. 記号文字は、特殊名段落のSYMBOLIC CHARACTERS句で定義します。
6. 表意定数は、特に制限がなければ、【書き方】で“定数-n”と示しているところに書くことができます。
7. ALLの後に2文字以上の定数を書いたALL定数と、数字項目または数字編集項目との間で転記または比較を行うことは、廃要素です。新しく作成するプログラムでは、この要素を使わないことをおすすめします。
8. 何桁かの文字列を表す表意定数をVALUE句に書いた場合、またはそのような表意定数とデータ項目との間で転記または比較を行う場合、表意定数はデータ項目と同じ長さを持つとみなされます。すなわち、表意定数で指定した文字列が、データ項目の長さより大きいか等しくなるまで繰り返され、データ項目の長さを超えた部分が、データ項目の長さに合わせて右から切り捨てられます。表意定数の値は、データ項目のJUSTIFIED句を処理する前に決まります。

1.2.5 連結式

連結式は、2つ以上の定数を連結演算子でつないで、1つの定数の値を表現する方法です。

使い方の例については、“付録H サンプル集”の“H.1 連結式”を参照してください。

【書き方】

[ALL] {定数-1} {& 定数-2} …

---

1. “&”を「連結演算子」といいます。連結演算子の前後には、分離符の空白を書かなければなりません。
2. 定数-1および定数-2は、以下のいずれかの組合せでなければなりません。
   - 文字定数、16進文字定数、表意定数SPACE、表意定数HIGH-VALUE、表意定数LOW-VALUE、表意定数QUOTE、表意定数ZEROおよび記号文字の中の1つ以上の組合せ。この場合、連結式で表される定数の項類は、英数字です。
   - 日本語定数の組合せ、または日本語定数と表意定数SPACEの組合せ。この場合、連結式で表される定数の項類は、日本語です。
3. 定数-1または定数-2に表意定数を書く場合、表意定数にALLを付けてはいけません。
4. 定数-1または定数-2に表意定数を書いた場合、表意定数の値として以下の値が使われます。
   - 連結式の中の定数に、少なくとも1つの文字定数または16進文字定数を書いた場合、または連結式の中の定数に表意定数だけを書いた場合、表意定数の値として、表意定数で指定した1文字の英数字が使われます。
   - 連結式の中の定数に、少なくとも1つの日本語定数を書いた場合、表意定数の値として、表意定数で指定した1文字の日本語文字が使われます。
5. 連結式の値は、連結式の中の各定数の値を、左から順に連結した値です。
6. ALLは、連結式で表した定数を表意定数として使うときに書きます。ALLを書いた連結式は、ALL定数と同じように扱われます。
7. 連結式は、【書き方】で“定数-n”と示しているところで、連結式の項類を持つ定数が書けるところに書くことができます。
8. 連結式の中の連結演算子と定数の間に、注記行または空白行を書くこともできます。

1.2.6 特殊な用途の定数

特殊な用途の定数として、プログラム名定数、ファイル識別名定数および原文名定数があります。

プログラム名定数

プログラム名定数は、プログラム名を値として持つ定数です。
プログラム名定数の値は、システムで定められたプログラム名の規則に従うものでなければなりません。プログラム名定数の規則の詳細は、“NetCOBOL使用手引書”を参照してください。
プログラム名定数は、【書き方】で“プログラム名定数-n”と示しているところに書くことができます。

ファイル識別名定数

ファイル識別名定数は、順ファイル、相対ファイルまたは索引ファイルのファイル識別名を値として持つ定数です。
ファイル識別名定数の形式を以下に示します。

```
″名前″
```

名前には、記憶媒体上のファイルの名前を指定します。
ファイル識別名定数は、文字定数の規則に従って書かなければなりません。ファイル識別名定数の値および長さは、システムで定められた規則に従うものでなければなりません。ファイル識別名定数の規則の詳細は、“NetCOBOL使用手引書”を参照してください。
ファイル識別名定数は、【書き方】で“ファイル識別名定数-n”と示しているところに書くことができます。

原文名定数

原文名定数は、原文名を値として持つ定数です。
原文名定数は、文字定数の規則に従って書かなければなりません。原文名定数の値および長さは、システムで定められた規則に従うものでなければなりません。原文名定数の規則の詳細については、“NetCOBOL使用手引書”を参照してください。
原文名定数は、【書き方】で“原文名定数-n”と示しているところに書くことができます。

1.2.7 PICTURE句の文字列

PICTURE句の文字列は、PICTURE句の中に書く文字列です。PICTURE句の文字列は、COBOL文字集合中の英字、数字および特殊文字で構成します。
PICTURE句の文字列の記述規則は、“5.4.7 PICTURE句”で説明します。

1.2.8 注記項

注記項は、見出し部の記述項です。注記項は、計算機文字集合中の任意の文字で構成します。

注記項は、廃要素です。新しく作成するプログラムでは、この要素を使わないことをおすすめします。

1.3 データ記述の概念

ここでは、データ記述の概念を説明します。

1.3.1 レベルの概念

1つ以上のデータで構成する、最も包括的な集まりを、「レコード」といいます。
レコードは、レベルの概念に従って構成します。この概念は、データを参照するためにレコードを細分する必要性からきています。一度細分したものをさらに細分して、データを参照することができます。
レコードで最も基本的な細分、つまりそれ以上分割できない部分を「基本項目」といいます。いくつかの基本項目は、それらをまとめて参照するために、集団にすることができます。さらに、いくつかの集団をまとめて、集団にすることもできます。したがって、基本項目は、いくつかの集団の下位に従属することができます。基本項目を従属する集団を、「集団項目」といいます。

1.3.2 字類の概念

指標データ項目、ポインタデータ項目および内部浮動小数点項目以外の基本項目は、いずれかの項類および字類に属します。集団項目の字類は、それに従属する基本項目が何であっても、実行時には英数字として扱われます。
データ項目の字類と項類の関係を、下表に示します。

項目のレベル	種類	項類	字類
基本項目	英字項目	英字	英字
	数字項目	数字	数字
	数字編集項目	数字編集	英数字
	英数字項目	英数字	
	英数字編集項目	英数字編集	
	日本語項目	日本語	日本語
	日本語編集項目	日本語編集	
	ブール項目	ブール	ブール
	外部浮動小数点項目	外部浮動	英数字
集団項目	―	英字 数字 数字編集 英数字 英数字編集	英数字
		日本語 日本語編集 ブール 外部浮動	

1.3.3 型の概念

- 型はデータ項目とその従属項目のすべての属性を含むテンプレートで、TYPEDEF句を使用して宣言されます。型名によって識別される型に不可欠な属性は、型宣言の中で定義される基本項目それぞれの長さと相対的な文字位置、各項目のPICTURE句、USAGE句、SIGN句、SYNCHRONIZED句、JUSTIFIED句、BLANK WHEN ZERO句の指定です。
- 型は、データ記述項にTYPE句を指定することにより参照されます。この指定によって型付けされた項目は、参照する型のすべての属性を持ちます。
- 型付けには強い型付けと弱い型付けがあります。
  基本項目は弱い型付けしかできませんが、集団項目は強い型付けおよび弱い型付けができます。

型は[Win32][WinIPF][Winx64][Solaris][Linux][LinuxIPF][Linux64][.NET]固有の機能です。ただし、[.NET]では強い型付けは使用できません。

1.3.3.1 弱く型付けされた項目

弱く型付けされた項目は、参照する型の属性を持ちます。それらの属性は、型付けられた項目の上位集団項目によって上書きできないことを除けば、型付けされていない項目と同様に使用できます。したがって、弱い型付けの宣言は一連のデータ記述項の略記とみなすことができます。

1.3.3.2 強く型付けされた項目

強く型付けされた項目は、参照する型の属性を持ち、以下の制限によってデータが完全に保護される項目です。

1. 強く型付けされた集団項目とは、STRONGが指定されている型宣言を参照するレベル番号01の集団項目、または、それに従属する集団項目のことをいいます。
2. 強く型付けされた集団項目の中にVALUE句を含むことはできません。
3. 強く型付けされた集団項目とその集団項目に従属する基本項目は、以下の規則に従わなければなりません。
   a. 明示的または暗示的に再定義してはなりません。
   b. 全部または一部を再命名してはなりません。
   c. 項類が英数字または日本語でない基本項目を部分参照してはなりません。
   d. 集団項目を部分参照してはなりません。
4. 強く型付けされた項目は、以下で参照することができます。
   a. メソッド呼出しのパラメタ、仮パラメタ、復帰項目
   b. 比較条件
   c. DISPLAY文
   d. INITIALIZE文
   e. MOVE文(ただし、CORRESPONDING指定は除く)
   f. READ文
   g. RELEASE文
   h. RETURN文
   i. REWRITE文
   j. WRITE文
   k. LENGTH関数
   l. LENG関数
5. 強く型付けされた項目の転記、比較および受け渡しをする場合、以下の規則に従わなければなりません。

   ー 一方がTYPE句で型を参照する項目である場合、もう一方は同等な型を参照する項目でなければなりません。

ー 一方が参照する型の従属項目である場合、もう一方は同等な型に従属する、同じ相対位置から始まり、同じ長さを持つ項目でなければなりません。

ここで、同等な型とは、2つの型宣言が以下の条件を満足していることをいいます。

a. 型名は、同じ名前で宣言されていなければなりません。

b. 型宣言に従属する集団項目がTYPE句を持つ場合、TYPE句で参照する型名は同じでなければなりません。

c. 型宣言に従属する基本項目は、同じ相対位置から始まり、同じ長さを持たなければなりません。

d. 型宣言に従属する基本項目に、BLANK WHEN ZERO句、JUSTIFIED句、PICTURE句、SIGN句、SYNCHRONIZED句、USAGE句の指定がある場合、これらの句の情報は一致していなければなりません。

1.3.4 標準桁よせ規則

データを基本項目に格納するときの桁よせの規則は、受取り側の項類によって決まります。送出し側のデータは、以下の規則に従って、桁よせされます。

1. 受取り側が数字項目の場合、以下の規則に従って、桁よせされます。

   a. 受取り側に想定小数点を明示している場合、送出し側は、受取り側の小数点の位置に合わせて格納されます。このとき、必要に応じて、端にゼロが補われたり、端が切り捨てられたりします。

      ただし、このコンパイラでは、翻訳オプションTRUNCを指定する必要があります。

   b. 受取り側に想定小数点を明示していない場合、受取り側の右端に想定小数点があるものとみなされ、a.の規則に従って、桁よせされます。

   ただし、受取り側がint型2進整数データ項目であり、かつ送出し側の整数部の値が、“表5.1 int型2進整数データ項目の大きさと値の範囲”に示されている値の範囲に収まらない場合、転記の結果は規定されません。

2. 受取り側が数字編集項目の場合、送出し側は、受取り側の小数点の位置に合わせて編集転記されます。このとき、必要に応じて、端にゼロが補われたり、端が切り捨てられたりします。ただし、数字位置の先行ゼロ列を空白などで置き換える場合、その桁位置にはゼロは補われません。

3. 受取り側が英数字項目、英数字編集項目、英字項目、日本語項目または日本語編集項目の場合、送出し側は、受取り側の左端に合わせて転記されます。このとき、必要に応じて、右端に空白が補われたり、右端が切り捨てられたりします。ただし、受取り側にJUSTIFIED句を指定した場合は、JUSTIFIED句の規則に従って、桁よせされます。

4. 受取り側がブール項目の場合、送出し側の最左端のブール位置を受取り側の最左端のブール位置に合わせて格納されます。このとき、必要ならば、右端にブール文字の0が補われたり、右端が切り捨てられたりします。

1.3.5 データの境界調整

データ記述項にSYNCHRONIZED句を書いた場合、データ項目を固有の境界に割り付けるために、以下がコンパイラによって挿入されることがあります。

- 使用されない文字位置
- ブール位置

固有の境界に合わせるためにコンパイラが挿入する文字位置を、「遊びバイト」といいます。

固有の境界に合わせるためにコンパイラが挿入するブール位置を、「遊びビット」といいます。

1.3.5.1 遊びバイト

2進項目または内部浮動小数点項目にSYNCHRONIZED句を指定した場合、そのデータ項目は、SYNCHRONIZED句の規則に従って、固有の境界に割り付けられます。このとき、レコードの中に、遊びバイトが挿入されることがあります。

遊びバイトを挿入する規則

コンパイラは、レコードの中の各データ項目に相対番地を割り当てます。レコードの先頭を0番地とし、データ項目の相対番地にデータ項目自身の大きさを加えた値を、次のデータ項目の相対番地とします。

2進項目または内部浮動小数点項目にSYNCHRONIZED句を指定した場合、桁づめが必要になります。レコードの中に桁づめを必要とするデータ項目があり、その相対番地がそのデータ項目の固有の境界のバイト数の倍数でない場合、遊びバイトが挿入されます。遊びバイトは、桁づめに必要な最小の長さを持つ、暗黙のFILLER項目です。

遊びバイトが挿入される位置は、以下のとおりです。

1. 桁づめを必要とするデータ項目の直前のデータ項目が基本項目の場合、桁づめを必要とするデータ項目のレベル番号と同じレベル番号を持つFILLER項目が、桁づめを必要とするデータ項目の直前に挿入されます。
2. 桁づめを必要とするデータ項目の直前のデータ項目が集団項目の場合、桁づめを必要とするデータ項目を従属する一連の集団項目のうちで、直前に基本項目がありかつレベル番号が最も大きいものを求めます。そして、その集団項目のレベル番号と同じレベル番号を持つFILLER項目が、その集団項目の直前に挿入されます。
3. OCCURS句を指定した集団項目が桁づめを必要とするデータ項目を従属する場合、その集団項目の1つの反復の大きさが、その集団項目に従属するデータ項目の固有の境界のバイト数の最大値の倍数でないとき、遊びバイトが挿入されます。遊びバイトは、その集団項目の各反復の最後に挿入されます。

遊びバイトが挿入された場合、集団項目の大きさは、集団項目の中に挿入された暗黙のFILLER項目の大きさだけ拡張されます。

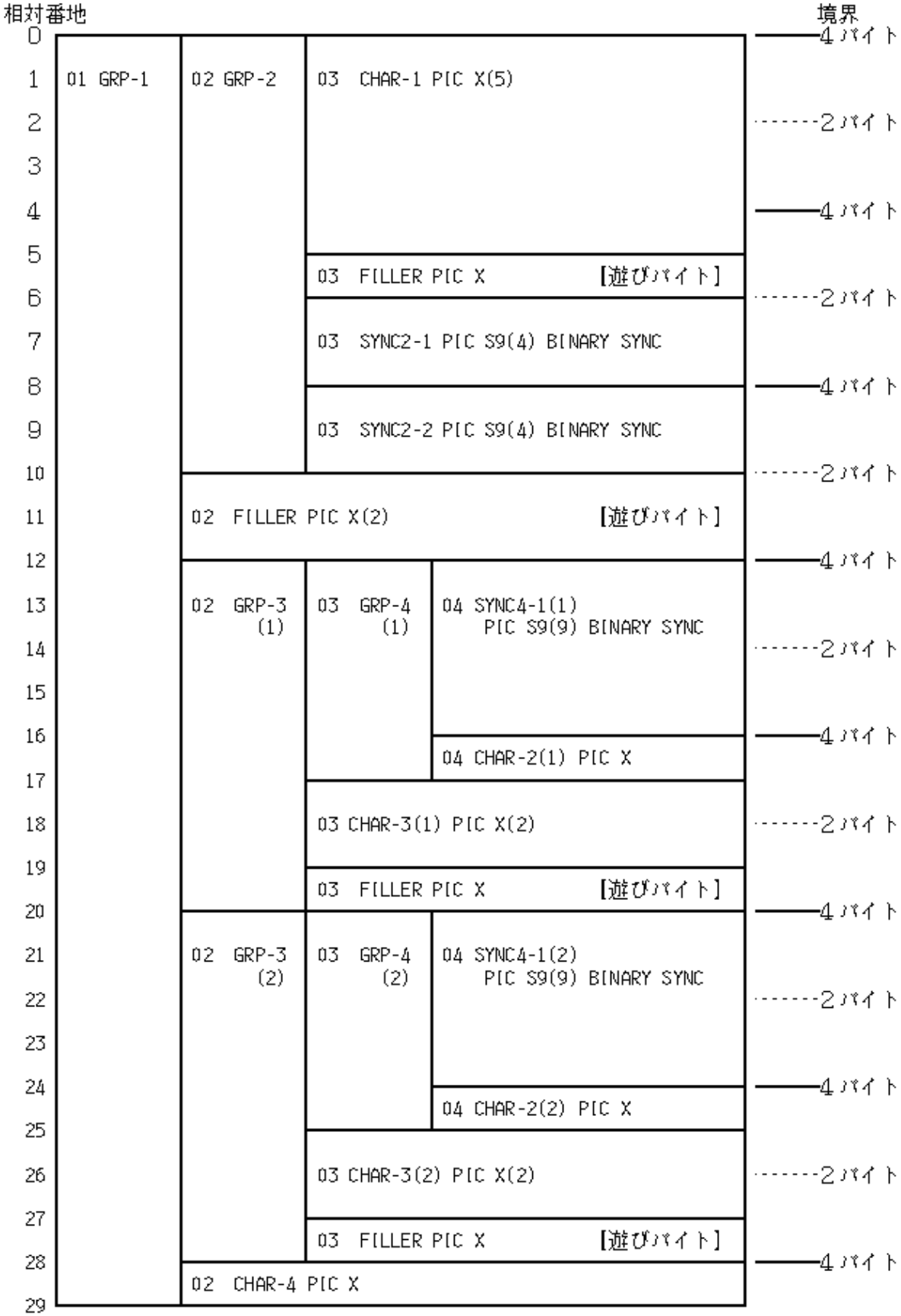
例

遊びバイトの例

遊びバイトが挿入される例を、以下のレコード記述項に従って説明します。

```
01  GRP-1.
    02  GRP-2.
        03  CHAR-1   PIC X(5).
        03  SYNC2-1  PIC S9(4)  BINARY SYNC.
        03  SYNC2-2  PIC S9(4)  BINARY SYNC.
    02  GRP-3    OCCURS 2.
        03  GRP-4.
            04  SYNC4-1  PIC S9(9)  BINARY SYNC.
            04  CHAR-2   PIC X.
        03  CHAR-3     PIC X(2).
    02  CHAR-4       PIC X.
```

上記のレコード記述項の領域は、下図のように割り当てられます。


相対番地
境界
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
01 GRP-1
02 GRP-2
03 CHAR-1 PIC X(5)
4バイト
2バイト
4バイト
03 FILLER PIC X 【遊びバイト】
2バイト
03 SYNC2-1 PIC S9(4) BINARY SYNC
4バイト
03 SYNC2-2 PIC S9(4) BINARY SYNC
2バイト
02 FILLER PIC X(2) 【遊びバイト】
4バイト
02 GRP-3 (1)
03 GRP-4 (1)
04 SYNC4-1(1) PIC S9(9) BINARY SYNC
2バイト
04 CHAR-2(1) PIC X
4バイト
03 CHAR-3(1) PIC X(2)
2バイト
03 FILLER PIC X 【遊びバイト】
4バイト
02 GRP-3 (2)
03 GRP-4 (2)
04 SYNC4-1(2) PIC S9(9) BINARY SYNC
2バイト
04 CHAR-2(2) PIC X
4バイト
03 CHAR-3(2) PIC X(2)
2バイト
03 FILLER PIC X 【遊びバイト】
4バイト
02 CHAR-4 PIC X

1.3.5.2 遊びビット

内部ブール項目にSYNCHRONIZED句を指定した場合、その内部ブール項目には、1バイト境界で始まるバイト単位の領域が割り付けられます。このとき、レコードの中に、遊びビットが挿入されることがあります。

遊びビットを挿入する規則

遊びビットは、以下の規則に従って挿入されます。

1. 遊びビットは、以下の位置に挿入されます。
   a. レコードの中に、SYNCHRONIZED句を指定した内部ブール項目がある場合、桁づめされた内部ブール項目の右端と次のバイト境界の間。
   b. レコードの中で、SYNCHRONIZED句の指定のない内部ブール項目の直後に、バイト境界に割り付けられるデータ項目(SYNCHRONIZED句の指定のない内部ブール項目以外のデータ項目)が続く場合、内部ブール項目の右端と次のバイト境界の間。
   c. 集団項目に従属するデータ項目の中で、最後のデータ項目が内部ブール項目の場合、その内部ブール項目の右端と次のバイト境界の間。
2. 遊びビットの大きさは、以下の手順で決定されます。
   a. 最初に、内部ブール項目が占めるブール位置の個数を求めます。1.のb.の場合、SYNCHRONIZED句の指定のない内部ブール項目が連続しているときは、それらの一連の内部ブール項目のブール位置の個数の合計を求めます。1.のc.の場合、最後の内部ブール項目の前にSYNCHRONIZED句の指定のない内部ブール項目が連続しているときは、それらの一連の内部ブール項目のブール位置の個数の合計を求めます。
   b. a.で求めたブール位置の個数を、8で割ります。
   c. b.の除算の余りが0の場合、遊びビットは不要です。余りが0でない場合、遊びビットが挿入されます。b.の除算の余りをrとすると、遊びビットの大きさは、8－rビットです。
3. 遊びビットは、遊びビットの直前のデータ項目のレベル番号と同じレベル番号を持つFILLER項目として挿入されます。遊びビットの大きさは、遊びビットが挿入された集団項目の大きさに数えられます。
4. OCCURS句を指定した集団項目が内部ブール項目を従属する場合も、遊びビットが挿入されます。このとき、表要素の各反復を集団項目とみなして、1.のc.が適用されます。
5. 以下では、遊びビットは自動的に挿入されます。
   － 出力ファイル
   － 作業場所節
   － 定数節

   入力ファイルおよび連絡節では、遊びビットがあるものとみなされるので、遊びビットを考慮して、レコード記述項を書く必要があります。

例

遊びビットの例 (その1)

遊びビットの例を、以下のレコード記述項に従って説明します。

```
01  RECORD-A.
    02  DATA-A1.
        03  DATA-A11  PIC 1(4) BIT.
        03  DATA-A12  PIC 1(3) BIT SYNC.
        03  DATA-A13  PIC 1(5) BIT.
    02  DATA-A2  PIC 1(4) BIT.
```

- DATA-A12は、1バイト境界に割り付けられるので、DATA-A11とDATA-A12との間に、遊びビットが4ビット挿入されます。

・ DATA-A12には、1バイトの領域が割り付けられるので、DATA-A12とDATA-A13との間に、遊びビットが5ビット挿入されます。

・ DATA-A13はDATA-A1に従属する最後の内部ブール項目なので、DATA-A13の後に、遊びビットが3ビット挿入されます。

上記のレコード記述項は、以下のように書いた場合と等価です。

```
01  RECORD-A.
    02  DATA-A1.
        03  DATA-A11  PIC 1(4) BIT.
        03  FILLER    PIC 1(4) BIT.          ←遊びビット
        03  DATA-A12  PIC 1(3) BIT.
        03  FILLER    PIC 1(5) BIT.          ←遊びビット
        03  DATA-A13  PIC 1(5) BIT.
        03  FILLER    PIC 1(3) BIT.          ←遊びビット
    02  DATA-A2  PIC 1(4) BIT.
```

例

遊びビットの例 (その2)

OCCURS句を指定した集団項目での遊びビットの例を、以下のレコード記述項に従って説明します。

```
01  RECORD-B.
    02  DATA-B1 OCCURS 10 TIMES.
        03  DATA-B11  PIC 1(5) BIT.
        03  DATA-B12  PIC 1(5) BIT.
```

DATA-B1の繰返しにおいて、DATA-B1は1バイト境界から始まるように割り付けられます。したがって、DATA-B12の後に、遊びビットが6ビット挿入されます。遊びビットは、DATA-B1の繰返しにおいて、同じように挿入されます。

なお、SYNCHRONIZED句の指定のない内部ブール項目にOCCURS句を指定した場合、表要素の各反復の間に遊びビットは挿入されません。例えば、以下の場合、DATA-C1の各反復において、遊びビットは挿入されません。

```
    02  DATA-C1 PIC 1(3) BIT OCCURS 6.
```

1.4 一意参照

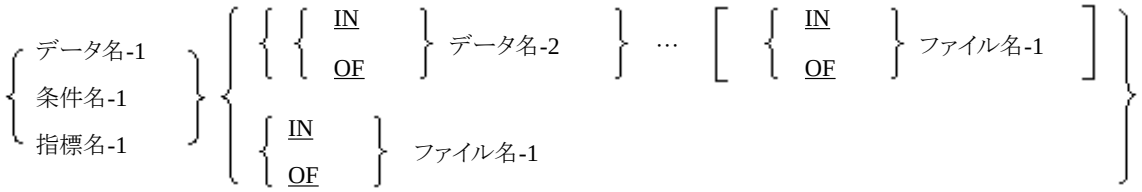
プログラムで使うすべての利用者語は、一意になるように書かなければなりません。利用者語を一意にする方法を「一意参照」といいます。

1.4.1 修飾

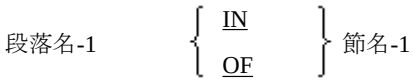
同一の綴りを持つ利用者語が同じ名前の範囲の中に2つ以上存在する場合、利用者語を修飾しなければなりません。修飾は、名前の階層系列の上位にある1つ以上の名前を付加することによって行います。名前の範囲は、“2.3.9 名前の範囲”で説明します。

使い方の例については、“付録H サンプル集”の“H.2 修飾”を参照してください。

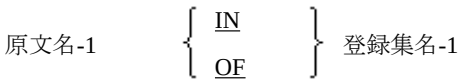
【書き方1】データ名、条件名または指標名を一意にする



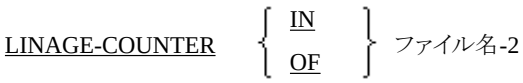
【書き方2】段落名を一意にする



【書き方3】原文名を一意にする

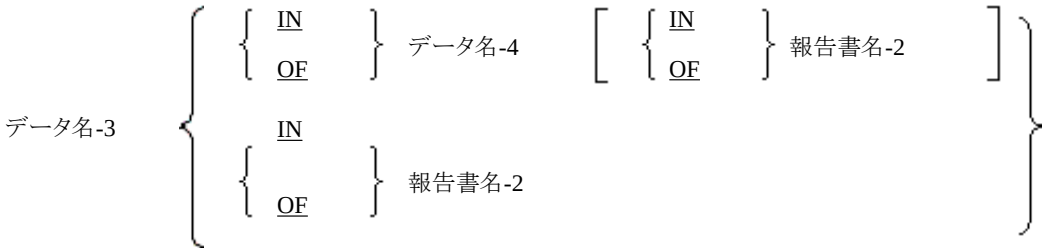


【書き方4】順ファイル機能で使う特殊レジスタを一意にする

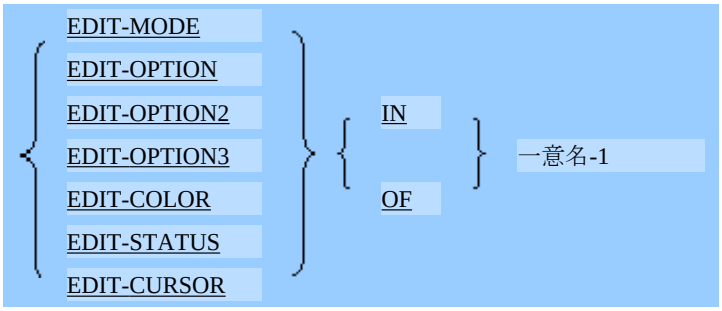


【書き方5】報告書作成機能で使う特殊レジスタを一意にする

【書き方6】報告書の中のデータ名を一意にする



【書き方7】表示ファイル機能で使う特殊レジスタを一意にする



EDIT-OPTION2、EDIT-OPTION3は、[Win32][WinIPF][Winx64][Solaris][Linux][LinuxIPF][Linux64][.NET]固有の機能です。

書き方1～書き方7に共通する規則

1. INまたはOFとそれに続く語を合わせて、「修飾語」といいます。一意でない利用者語を参照する場合、一連の修飾語を書くことによって一意にしなければなりません。
2. 利用者語が一意になるまで修飾語を付加しなければなりませんが、一意になるまでのすべての修飾語を付加する必要はありません。
3. 修飾する必要のない利用者語を修飾することもできます。
4. INとOFは、同義語です。

書き方1の規則

1. 修飾語は、以下のいずれかでなければなりません。
    a. データ名-1をデータ名-2で修飾する場合、データ名-2は、データ名-1を従属する集団項目のデータ名でなければなりません。
    b. 条件名-1をデータ名-2で修飾する場合、データ名-2は、条件名-1に関係付けた条件変数のデータ名でなければなりません。
    c. 指標名-1をデータ名-2で修飾する場合、データ名-2は、INDEXED BY指定に指標名-1を書いたデータ記述項のデータ名でなければなりません。
    d. データ名-1、条件名-1または指標名-1をファイル名-1で修飾する場合、ファイル名-1は、修飾される項目(データ名-1、条件名-1または指標名-1)に関係付けたファイル記述項または整列併合用ファイル記述項のファイル名でなければなりません。
2. 修飾語は、左から右に向かって、階層系列の下位から上位のレベルの順に指定します。
3. データ名-1およびデータ名-2に、レコード名を指定することもできます。
4. 条件名を修飾する場合、修飾語として、条件名に関係付けた条件変数の階層系列を使います。

書き方2の規則

1. 段落名を明に参照する場合、1つの節の中に同じ段落名を定義してはいけません。
2. 段落名を定義した節と同じ節の中で段落名を参照する場合、段落名を修飾する必要はありません。
3. 節名-1に、SECTIONを付けてはいけません。
4. 段落名および節名を別のプログラムから参照することはできません。

書き方3の規則

2つ以上のCOBOL登録集を使う場合、原文名を参照するときには、原文名を登録集名で修飾しなければなりません。

書き方4の規則

2つ以上のファイル記述項にLINAGE句を書いた場合、特殊レジスタLINAGE-COUNTER(行数カウンタ)を参照するときには、LINAGE-COUNTERをファイル名で修飾しなければなりません。

書き方5の規則

1. 2つ以上の報告書記述項を書いた場合、特殊レジスタLINE-COUNTER(行カウンタ)を手続き部で参照するときには、LINE-COUNTERを報告書名で修飾しなければなりません。
2. 報告書節で、LINE-COUNTERを修飾しないで参照した場合、LINE-COUNTERはその報告書名で暗に修飾されます。報告書節で、別の報告書のLINE-COUNTERを参照するときには、LINE-COUNTERをその報告書名で明に修飾しなければなりません。
3. 2つ以上の報告書記述項を書いた場合、特殊レジスタPAGE-COUNTER(ページカウンタ)を手続き部で参照するときには、PAGE-COUNTERを報告書名で修飾しなければなりません。

4. 報告書節で、PAGE-COUNTERを修飾しないで参照した場合、PAGE-COUNTERはその報告書名で暗に修飾されます。報告書節で、別の報告書のPAGE-COUNTERを参照するときには、PAGE-COUNTERをその報告書名で明に修飾しなければなりません。

書き方6の規則

1. 修飾語は、以下のいずれかでなければなりません。
   - データ名-3をデータ名-4で修飾する場合、データ名-4は、データ名-3を従属する集団項目のデータ名でなければなりません。
   - データ名-3を報告書名-2で修飾する場合、報告書名-2は、データ名-3に関係付けた報告書記述項の報告書名でなければなりません。
2. 修飾語は、左から右に向かって、階層系列の下位から上位のレベルの順に指定します。

書き方7の規則

1. 特殊レジスタEDIT-MODE、EDIT-OPTION、EDIT-OPTION2、EDIT-OPTION3、EDIT-COLOR、EDIT-STATUSおよびEDIT-CURSORを参照する場合、必ず修飾しなければなりません。
2. 一意名-1は、項目制御部を持つと定義したデータ項目でなければなりません。項目制御部については、“NetCOBOL使用手引書”を参照してください。
3. 一意名-1は、部分参照することはできません。
4. 一意名-1は、必要な修飾および添字付けをしなければなりません。

1.4.2 添字付け

添字付けは、表要素を一意に参照するために使います。表要素を参照する場合、データ名または条件名に添字を付けます。

使い方の例については、“付録H サンプル集”の“H.3 添字付け”を参照してください。

【書き方】

{ データ名-1 / 条件名-1 } ({ 整数-1 / データ名-2[{ + / − } 整数-2] / 指標名-1[{ + / − } 整数-3] / ALL } …)

参考

“データ名-1”および“条件名-1”は、書き方を明確にするために示しているだけで、添字の一部ではありません。

構文規則

1. データ名-1は、OCCURS句を指定したデータ項目、またはそのようなデータ項目に従属するデータ項目でなければなりません。

2. 条件名-1に関係付けたデータ項目(条件変数)は、OCCURS句を指定したデータ項目、またはそのようなデータ項目に従属するデータ項目でなければなりません。条件変数を参照するために添字が必要な場合、条件変数と同じ組合せの添字を条件名に付けなければなりません。
3. 以下の場合を除いて、表要素は添字付けして参照しなければなりません。
   - SEARCH文の主体
   - REDEFINES句
   - OCCURS句のKEY IS指定
4. 添字の個数は、表要素を含むデータ記述項に書いたOCCURS句の個数と同じでなければなりません。2つ以上の添字が必要な場合、その表の上位の次元から下位の次元の順に、左から順に添字を書きます。
5. 指標名-1は、表要素を含むデータ記述項のOCCURS句のINDEXED BY指定に書いた指標名でなければなりません。
6. データ名-2は、整数項目でなければなりません。
7. データ名-2は、修飾することができます。
8. 整数-1に符号を付けることもできます。符号を付ける場合、正号でなければなりません。
9. 添字のALLは、関数の引数を書く場合にだけ使うことができます。条件名-1の添字としてALLを書くことはできません。
10. 報告書節の中では、合計カウンタ、行カウンタおよびページカウンタを添字として使うことはできません。
11. データ名-1および条件名-1は、修飾することができます。

一般規則

1. 1つの次元は、1つのOCCURS句に対応します。1つの次元の表要素のうちどの要素を参照するかは、出現番号で表します。OCCURS句に指定した反復回数の最大値をnとすると、その次元の最初の表要素は出現番号1で表し、2番目の表要素は出現番号2で表し、その次元の最後の表要素は出現番号nで表します。
2. 添字の個数の最大値、すなわち次元の個数の最大値は7です。
3. 添字に整数-1またはデータ名-2を書いた場合、添字の値は表要素の出現番号を表します。
4. 添字に指標名-1を書いた場合、添字の値は表要素の出現番号に対応する値を表します。
5. 指標名-1の値は、その値を添字として使う前に初期設定しなければなりません。指標名-1の初期値は、VARYING指定付きのPERFORM文、ALL指定付きのSEARCH文およびSET文によって設定することができます。指標名の値は、PERFORM文、SEARCH文およびSET文によってだけ変更することができます。
6. 整数-2を書いた場合、データ名-2の値(出現番号)に整数-2の値を加えた値(演算子が＋のとき)、またはデータ名-2の値から整数-2の値を引いた値(演算子がーのとき)が添字の値になります。
7. 整数-3を書いた場合、指標名-1の値(出現番号に対応する値)に整数-3の値を加えた値(演算子が＋のとき)、または指標名-1の値から整数-3の値を引いた値(演算子がーのとき)が添字の値になります。
8. 添字にALLを書いた場合、添字に対応する次元にあるすべての表要素が参照されます。

1.4.3 部分参照

部分参照は、データ項目または関数値の一部を参照するために使います。部分参照する場合、データ名または関数一意名に部分参照子を付けます。

使い方の例については、“付録H サンプル集”の“H.4 部分参照”を参照してください。

【書き方1】データ項目を部分参照する

データ名-1(最左端文字位置:[長さ])

【書き方2】関数値を部分参照する

FUNCTION 関数名-1[({引数-1} …)] (最左端文字位置:[長さ])

参考

“データ名-1”および“FUNCTION 関数名-1[({引数-1}…)]”は、書き方を明確にするために示してあるだけで、部分参照子の一部ではありません。

構文規則

1. データ名-1は、用途が表示用の基本項目または集団項目でなければなりません。
2. 最左端文字位置および長さは、算術式でなければなりません。
3. データ名-1は、修飾または添字付けすることができます。
4. 関数名-1は、英数字関数でなければなりません。

   ただし、[HP][Win16][Win32][WinIPF][Winx64][Solaris][Linux][LinuxIPF][Linux64][.NET]では日本語関数も指定できます。
5. 強く型付けされた集団項目は、部分参照できません。
6. 強く型付けされた集団項目に従属する基本項目は、英数字および日本語でなければ部分参照できません。

一般規則

1. 部分参照は、データ名-1のデータ項目または関数値を持つ一時的なデータ項目の一部(部分参照元のデータ項目)を、一意なデータ項目(部分参照の結果のデータ項目)として新たに定義します。
2. 最左端文字位置には、部分参照元のデータ項目の最左端文字位置を1として、部分参照を開始する文字位置を設定します。最左端文字位置の値は、以下の範囲にある整数でなければなりません。

1 ≦ 最左端文字位置の値 ≦ 部分参照元のデータ項目の文字位置の個数

3. 長さには、部分参照する範囲を設定します。長さの値は、以下の範囲にある整数でなければなりません。

1 ≦ 長さ ≦ 部分参照元のデータ項目の文字位置の個数 － 最大左端文字の値 ＋ 1

4. 長さを省略した場合、部分参照元のデータ項目の最左端文字位置から始まり最右端文字位置までの部分が、部分参照の結果のデータ項目になります。
5. データ名-1に添字を付けた場合、部分参照子は添字の評価の直後に評価されます。データ名-1の添字にALLを指定した場合は、添字に対応する次元にあるすべての表要素に対して、部分参照子が適用されます。
6. 部分参照の結果のデータ項目は、JUSTIFIED句なしの基本項目とみなされます。
7. 書き方2の部分参照子は、関数値を求めた後、関数値を持つ一時的なデータ項目に対して適用されます。
8. 書き方1の場合、部分参照の結果のデータ項目は、以下の項類と字類を持つものとみなされます。
   - データ名-1の項類が英字、数字、数字編集、英数字または英数字編集の場合、部分参照の結果のデータ項目の項類と字類は英数字であるとみなされます。
   - データ名-1の項類が日本語または日本語編集の場合、部分参照の結果のデータ項目の項類と字類は日本語であるとみなされます。
   - データ名-1の項類がブールの場合、部分参照の結果のデータ項目の項類と字類は英数字であるとみなされます。

－ データ名-1が集団項目の場合、部分参照の結果のデータ項目の項類と字類は英数字であるとみなされます。

9. 書き方2の場合、部分参照の結果のデータ項目の項類と字類は英数字または日本語であるとみなされます。

1.4.4 ポインタ付け

ポインタ付けは、基底場所節で定義したデータ名または条件名を参照するために使います。ポインタ付けする場合、データ名または条件名にポインタ修飾子を付けます。ポインタ修飾子は、基底場所節で定義したデータ項目の記憶領域のアドレスを明示します。

使い方の例については、“付録H サンプル集”の“H.5 ポインタ付け”を参照してください。

【書き方】

```
                 { データ名-1                    }
[( ]…            { データ名-2 ( { 添字 } … )     } ->
                 { ADDR関数                      }

[ { データ名-1                    } ) -> ] … { データ名-3 }
  { データ名-2 ( { 添字 } … )     }          { 条件名-1  }
```

参考

“データ名-3”および“条件名-1”は、書き方を明確にするために示しているだけで、ポインタ修飾子の一部ではありません。また、“－＞”は必要語です。ここでは、他の記号との混同を避けるために、下線を付けていません。

構文規則

1. データ名-1およびデータ名-2は、ポインタデータ項目でなければなりません。
2. 記号“－＞”を「ポインタ修飾記号」といいます。ポインタ修飾記号の前後には、空白を書くこともできます。ポインタ修飾記号は、同一行に書かなければなりません。
3. 2つ以上のポインタ修飾記号を書く場合、2番目以降のポインタ修飾記号の前までを括弧で囲まなければなりません。

   ### 例

   ```
   PTR1->X
   (PTR1->PTR2)-X
   ((PTR2->PTR2)->PTR3)->X
   ```

4. データ名-1およびデータ名-2は、修飾することができます。
5. データ名-3および条件名-1は、基底場所節で定義しなければなりません。
6. データ名-3および条件名-1は、修飾または添字付けすることができます。データ名-3は、部分参照することができます。

一般規則

1. データ名-3が、BASED ON句を省略したデータ項目またはそのようなデータ項目に従属するデータ項目である場合、データ名-3はポインタ修飾子を付けて参照しなければなりません。

2. 条件名-1に関連付けた条件変数が、BASED ON句を省略したデータ項目またはそのようなデータ項目に従属するデータ項目である場合、条件名-1はポインタ修飾子を付けて参照しなければなりません。
3. 基底場所節のデータ記述項にBASED ON句を書くことによって、暗黙のポインタ修飾子を定義することができます。ポインタ修飾子を書かないで暗黙のポインタ修飾子を使ってポインタ付けすることを、「暗にポインタ付けする」といいます。ポインタ修飾子を書くことを、「明にポインタ付けする」といいます。
4. BASED ON句を指定したデータ項目またはそのようなデータ項目に従属するデータ項目は、暗黙のポインタ修飾子以外で明にポインタ付けすることもできます。条件名に関連付けた条件変数が、BASED ON句を指定したデータ項目またはそのようなデータ項目に従属するデータ項目である場合、条件名を、暗黙のポインタ修飾子以外で明にポインタ付けすることもできます。

1.4.5 一意名

データ名に、修飾、添字付け、部分参照またはポインタ付けをして一意にできるようにしたものを、「一意名」といいます。データ名を一意参照する場合、データ名を一意名の形式で書かなければなりません。

【書き方】

[ポインタ修飾子] データ名-1

[{ IN / OF } データ名-2] … [{ IN / OF } { ファイル名-1 / 報告書名-1 }]

[({添字} …)] [部分参照子]

---

1. INとOFは、同義語です。
2. 一意名は、【書き方】で“一意名-n”と示しているところに書くことができます。【書き方】で“一意名-n”と示しているところには、一意参照に必要な以下をしたデータ名を書かなければなりません。
   - 修飾
   - 添字付け
   - 部分参照
   - ポインタ付け
3. ポインタ修飾子は、データ名-1を基底場所節で定義した場合にだけ書くことができます。

1.4.6 条件名の一意参照

条件名は、名前の範囲の規則によって一意参照が保証される場合を除いて、修飾、添字付けまたはポインタ付けして一意にしなければなりません。

使い方の例については、“付録H サンプル集”の“H.6 条件名の一意参照”を参照してください。

【書き方】

[ポインタ修飾子] 条件名-1

[{ IN / OF } データ名-1] … [{ IN / OF } ファイル名-1]

[({添字} …)]

---

1. INとOFは、同義語です。
2. 条件名は、【書き方】で“条件名-n”と示しているところに書くことができます。【書き方】で“条件名-n”と示しているところには、一意参照に必要な以下をした条件名を書かなければなりません。
   - 修飾
   - 添字付け
   - ポインタ付け
3. ポインタ修飾子は、条件名-1を基底場所節で定義した場合にだけ書くことができます。

1.4.7 関数一意名

「関数一意名」は、関数を使うときの書き方です。関数一意名は、関数値を持つ一時的なデータ項目として扱われます。

【書き方】

FUNCTION 関数名-1 [({引数-1} …)] [部分参照子]

---

1. 引数-1は、一意名、定数または算術式でなければなりません。
2. 部分参照子は、以下の場合にだけ書くことができます。
   - 英数字関数
   - 日本語関数

   部分参照子は、関数値を部分参照するときに書きます。
3. 関数一意名を書くことができる場所は、“6.5 関数の全般規則”で説明します。
4. 関数の引数は、引数-1を書いた順に評価されます。
5. 引数-1に、関数一意名または関数一意名を含む式を書くこともできます。引数-1に、関数名-1を書くこともできます。

1.5 正書法

COBOL原始プログラムを書くときの形式を、「正書法」といいます。COBOL原始プログラムは、正書法に従って書かなければなりません。原始文操作機能を使う場合、COBOL登録集も、正書法に従って書かなければなりません。

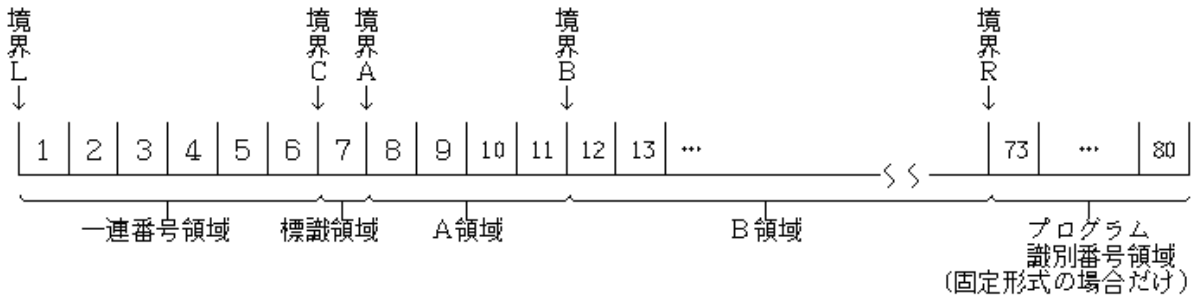
1.5.1 1行の構成

正書法は、1行のどの位置にどの要素を書くかを定めた規則です。正書法には、以下の3種類があります。

- 固定形式
- 可変形式
- 自由形式

なお、自由形式は、[Win32][WinIPF][Winx64][Solaris][Linux][LinuxIPF][Linux64][.NET]固有の機能です。

固定形式または可変形式の1行の構成を、以下に示します。



- 境界Lは、1行の最左端の文字位置のすぐ左です。
- 境界Cは、1行の6番目と7番目の文字位置の間です。
- 境界Aは、1行の7番目と8番目の文字位置の間です。
- 境界Bは、1行の11番目と12番目の文字位置の間です。
- 境界Rは、以下の位置です。
  - 固定形式の場合、1行の72番目の文字位置のすぐ右です。
  - 可変形式の場合、1行の最後の文字位置のすぐ右です。1行の最後の文字位置については、“付録B システムの定量制限”を参照してください。

一連番号領域

一連番号領域は、境界Lと境界Cの間の6文字分です。原始プログラムの行を識別するために使います。一連番号領域には、計算機文字集合中の任意の文字を書くことができます。一連番号領域の内容は、特定の順序で並んでいる必要も、一意である必要もありません。

標識領域

- 標識領域は、境界Cと境界Aの間の1文字分です。標識領域には、星印“＊”、斜線“/”、ハイフン“－”または“D”を書くことができます。
- 標識領域に星印“＊”または斜線“/”を書いた行を、「注記行」といいます。注記行のA領域とB領域の内容は、注釈とみなされます。
- 標識領域のハイフン“－”は、その行の最初の文字が前の行の最後の文字とつながっていることを示します。
- 標識領域に“D”を書いた行を、「デバッグ行」といいます。デバッグ行は、デバッグのための情報を原始プログラムに残すために使います。

A領域

A領域は、境界Aと境界Bの間の4文字分です。

B領域

B領域は、境界Bと境界Rの間です。

プログラム識別番号領域

プログラム識別番号領域は、固定形式の場合にだけ存在します。プログラム識別番号領域は、境界Rのすぐ右(73番目の文字位置)から80番目の文字位置までの8文字分です。

1.5.2 A領域とB領域の正書法

COBOLの語には、A領域から書き始めるものとB領域から書き始めるものがあります。

A領域から書き始める語

以下のものは、A領域から書き始めなければなりません。

- 部の見出し
- 節の見出し
- 段落の見出し
- 段落名
- レベル指示語 (FD、SDおよびRD)
- レベル番号の01および77
- 宣言部分の開始(“DECLARATIVES.”)と終了(“END DECLARATIVES.”)
- プログラム終わり見出し

B領域から書き始める語

以下のものは、B領域から書き始めなければなりません。

- 文
  ただし、COPY文およびREPLACE文は、A領域から書くことができます。
- 記述項
  ただし、レベル指示語で始まる記述項、およびレベル番号が01または77のデータ記述項は、A領域から書き始めなければなりません。レベル番号が01および77以外のデータ記述項は、A領域から書くこともできます。
- 注記項

各部の規則

前述の規則のほかに、各部ごとに以下の規則があります。

1. 見出し部では、1つの行に、段落の見出しと記述項または注記項を書くこともできます。1つの行に、2つ以上の注記項を書くこともできます。1つの注記項を、2行以上にまたがって書くこともできます。
2. 環境部では、1つの行に、段落の見出しと記述項を書くこともできます。1つの行に、2つ以上の記述項を書くこともできます。1つの記述項を、2行以上にまたがって書くこともできます。
3. データ部では、1つの行に、2つ以上の記述項を書くこともできます。1つの記述項は、2行以上にまたがって書くこともできます。レベル番号が1桁の場合、レベル番号を1桁の数字で書くこともできます。見やすさのために、レベルが深くなるごとに、レベル番号の開始位置をずらすこともできます。
4. 手続き部では、1つの行に、段落名と完結文を書くこともできます。1つの行に、2つ以上の完結文を書くこともできます。1つの完結文は、2行以上にまたがって書くこともできます。

1.5.3 空白行

標識領域、A領域およびB領域がすべて空白の行を、「空白行」といいます。空白行は、原始プログラムおよび登録集原文のどこでも書くことができます。

1.5.4 注記行

標識領域に星印“＊”または斜線“/”を書いた行を、「注記行」といいます。注記行は、見出し部の見出しの後の任意の位置、および登録集原文の任意の位置に書くことができます。

注記行のA領域とB領域には、計算機文字集合中の任意の文字を書くことができます。注記行は注釈とみなされますが、翻訳リストには印字されます。

標識領域に星印を書いた注記行は、前の行に続いて印字されます。標識領域に斜線を書いた注記行は、改ページした後に印字されます。

1.5.5 行のつなぎ

文、記述項および句を構成する文字列は、次の行のB領域に続けて書くことができます。継続する行を、「後の行」といい、継続される行を「前の行」といいます。

使い方の例については、“付録H サンプル集”の“H.7 行のつなぎ”を参照してください。

1. 以下の文字列は、後の行に継続して書くことができます。

   ― 日本語利用者語を除くCOBOLの語

   ― 定数

   ― PICTURE句の文字列

   上記の文字列を後の行に継続する場合、後の行の標識領域はハイフン“－”でなければなりません。また、後の行のA領域は、空白でなければなりません。標識領域のハイフンは、その行のB領域の空白でない最初の文字が、前の行(注記行および空白行を除く)の空白でない最後の文字の後に続くことを示します。

2. 以下を継続する場合は、さらに従わばけれればならない規則があります。

   ― 文字定数

   ― 16進文字定数、日本語定数またはブール定数

   上記を継続する場合は、以下の規則にも従わなければなりません。

   1. 後の行のB領域の空白でない最初の文字は、引用符でなければなりません。そして、その引用符のすぐ右の文字位置から、定数の続きを書き始めなければなりません。

   2. 前の行のB領域の最後までの空白は、定数の一部であるとみなされます。前の行のB領域の最後の文字が、後の行の最初の引用符の直後の文字に継続するように、書かなければなりません。

   なお、文字列の途中でなく文字列が終了した後に行を継続する場合は、標識領域にハイフンを書くことはできません。標識領域にハイフンがないと、その行の空白でない文字の前に空白があるものとみなされます。

3. 分離符、すなわち以下の文字列は、1つの行に書かなければなりません。

   ― 仮原文区切り記号“＝＝”

   ― 定数値の開始を示す分離符(“X””、“N””、“NC””、“NX””および“B””) データ名-1

   ― ポインタ修飾記号“－＞”

以下に、行を継続する例を示します。

例

```
標識領域                                                    境界R
  ↓                                                          ↓
   77  XX  PIC X(60) VALUE "ABCDE12345ABCDE12345ABCDE12345ABCDE12345
  -                      "ABCDE12345ABCDE12345".                    …[1]

       MOVE X
  -         X TO YY.                                                …[2]
       MOVE XX TO
              YY.                                                    …[3]
```

[1] 文字定数を継続しています。

[2] 利用者語XXを継続しています。

[3] 語と語の間で行を継続しています。標識領域にハイフンを書くことはできません。

1.5.6 デバッグ行

標識領域に“D”を書いた行を、「デバッグ行」といいます。デバッグ行は、デバッグのための情報を原始プログラムに残すために使います。デバッグ行は、見出し部の実行用計算機段落以降に書くことができます。

デバッグ行のA領域とB領域は、COBOLの文法に従って書かなければなりません。デバッグ行のA領域とB領域が共に空白の場合、デバッグ行は空白行とみなされます。

デバッグ行を翻訳するかどうかは、環境部の翻訳用計算機段落のWITH DEBUGGING MODE句で指定します。デバッグ行は、WITH DEBUGGING MODE句を書いた場合にだけ翻訳され、この句を省略した場合は注記行とみなされます。

翻訳時、WITH DEBUGGING MODE句の有無は、COPY文およびREPLACE文を処理した後に決定されます。

1.5.7 行内注記

文や記述項などを書いた行と同じ行の中に注釈を書きたい場合、行内注記を使って、行の右側に注釈を書くことができます。連続した2文字“＊＞”で始まり、同じ行の境界Rのすぐ左の文字位置で終わる部分を、「行内注記」といいます。行内注記は、見出し部の見出しの行とそれ以降の任意の行、および登録集原文中の任意の行に書くことができます。

行内注記の開始を示す記号“＊＞”の直前には、分離符の空白を書かなければなりません。“＊＞”の後には、計算機文字集合中の任意の文字を書くことができます。行内注記は注釈とみなされます。

1.5.8 自由形式の正書法

以下に、自由形式の正書法について説明します。

使い方の例については、“付録H サンプル集”の“H.8 自由形式の正書法”を参照してください。

なお、自由形式の正書法では、廃要素は使用できません。
自由形式は、[Win32][WinIPF][Winx64][Solaris][Linux][LinuxIPF][Linux64][.NET]固有の機能です。

1.5.8.1 1行の構成

自由形式の正書法では、注記、デバッグ行および継続のための特別な規則があることを除いて、原始プログラムは行のどの文字位置からでも書くことができます。

1行の文字位置の数は、行ごとに最小0から最大251の範囲で変更することができます。(ただし、日本語文字は、1文字で2つ以上の文字位置を占めます)

1.5.8.2 空白行

空白文字以外の文字を含まない行を、「空白行」といいます。空白行は、原始プログラムおよび登録集原文のどこにでも書くことができます。

1.5.8.3 注記

2つの隣合うCOBOL文字“＊＞”は注記を示します。

注記が記述されていて、0または1つ以上の空白だけが注記に先立つ行を「注記行」といいます。

1つ以上のCOBOLの語と、そのすぐ後ろに1つ以上の空白だけが注記に先立つ行の注記は、「行内注記」といいます。行内注記は、見出し部の見出しの行とそれ以降の任意の行、および登録集原文中の任意の行に書くことができます。

行内注記の開始を示す記号“＊＞”の直前には、分離符の空白を書かなければなりません。“＊＞”の後には、計算機文字集合中の任意の文字を書くことができます。行内注記は注釈とみなされます。

1.5.8.4 行のつなぎ

文、記述項および句は、複数の行に渡って書くことができます。この継続する行を「後の行」といい、継続される行を「前の行」といいます。

文字定数、ブール定数および日本語定数については、その一部を後の行に分けて書くことができます。これらの定数が行の終りで閉じていない場合、定数の閉じていない部分は、引用符とそのすぐ後に続くハイフンで終わらなければなりません。

ハイフンは0または1つ以上の分離符の空白が続いていなければなりません。

後の行の空白でない最初の文字は、引用符でなければなりません。また、引用符のすぐ右の文字位置から、定数の続きを書き始めなければなりません。このとき、定数の内容のうち少なくとも1文字は、前の行と後の行に書かれなければなりません。

2文字以上で構成される分離符は、同一行になければなりません。定数中の1つの引用符を示す一対の引用符は、同一行に記述しなければなりません。注記行と空白行は、定数の一部を含む行の間に書くことができます。

1.5.8.5 デバッグ行

3つの隣合うCOBOL文字“＞＞Ｄ”のすぐ後に1つの空白が続いているとき、デバッグ指示子を示します。デバッグ指示子に0または1つ以上の空白だけが先立つ行を「デバッグ行」といいます。デバッグ行は、デバッグのための情報を原始プログラムに残すために使います。デバッグ行は、見出し部の実行用計算機段落以降に書くことができます。デバッグ行は行の終わりで終了します。

デバッグ行は、COBOLの文法に従って書かなければなりません。デバッグ行が0または1つ以上の空白だけの場合、デバッグ行は空白行とみなされます。

デバッグ行を翻訳するかどうかは、環境部の翻訳用計算機段落のWITH DEBUGGING MODE句で指定します。デバッグ行は、WITH DEBUGGING MODE句を書いた場合にだけ翻訳され、この句を省略した場合は注記行とみなされます。

翻訳時、WITH DEBUGGING MODE句の有無は、COPY文およびREPLACE文を処理した後に決定されます。

1.6 プログラムの構成

プログラムは、以下の4つの部とプログラム終わり見出しで構成します。

- 見出し部
- 環境部
- データ部
- 手続き部

プログラムは、見出し部で開始し、プログラム終わり見出しまたはプログラムの最後の行で終了します。

プログラムの開始から終了までの間には、別のプログラムを書くことができます。別のプログラムに含まれるプログラムを、「内部プログラム」といいます。内部プログラムは、プログラム間連絡機能を使う場合に書くことができます。

一番外側のプログラムの開始から終了までの範囲を、「翻訳単位」といいます。1つの翻訳単位の中には1つ以上の内部プログラムを書くことができます。プログラムが別のプログラムを含むことを「プログラムの入れ子」といいます。

内部プログラムはさらに内部プログラムを含むことができます。あるプログラム(プログラムA)のすぐ内側に別のプログラム(プログラムB)を書き、プログラムBのすぐ内側に別のプログラム(プログラムC)を書いた場合、プログラムBはプログラムAに「直接に含まれる」といい、プログラムCはプログラムAに「間接に含まれる」といいます。

以下に、翻訳単位のプログラムの書き方を示します。

【書き方1】翻訳単位

見出し部

[環境部]

[データ部]

[手続き部]

[内部プログラム] …

[プログラム終わり見出し]

【書き方2】内部プログラム

見出し部
[環境部]
[データ部]
[手続き部]
[内部プログラム] …
プログラム終わり見出し

---

1. 翻訳単位の中に内部プログラムを書く場合、それぞれのプログラムに対してプログラム終わり見出しを書かなければなりません。例えば、プログラムAがプログラムBを含み、プログラムBがプログラムCを含む場合、A、BおよびCの各プログラムに対してプログラム終わり見出しを書かなければなりません。
2. 翻訳単位のプログラムは、プログラム終わり見出しによって終わらなければなりません。ただし、一連の翻訳単位のプログラムのうち、最後のプログラムでは、プログラム終わり見出しを省略することができます。
3. それぞれの部の開始は、部の見出しで表します。部の終了は、以下のいずれかで表します。
   - 同じプログラム中の次に続く部の見出し
   - 内部プログラムの始まりを表す見出し部の見出し
   - プログラム終わり見出し
   - その後に行が存在しない物理的な位置

1.6.1 プログラムの結果

プログラム定義の手続き部の見出しにRETURNING指定が書かれた場合、プログラムから呼出し元に制御を戻す際に、RETURNING指定のデータ項目の内容がプログラムの結果となります。この結果は、呼出し元のCALL文のRETURNING指定に書かれた一意名に入ります。

第2章 COBOLの機能

COBOLの機能には、以下の10種類があります。

- 中核機能
- 入出力機能(順ファイル機能、相対ファイル機能および索引ファイル機能)
- プログラム間連絡機能
- 整列併合機能
- 原始文操作機能
- 表示ファイル機能
- 組込み関数機能
- スクリーン操作機能
- コマンド行引数と環境変数の操作機能
- 報告書作成機能

本章では、これらの機能の概要を説明します。

2.1 中核機能

中核機能は、データを変換、比較および演算するための基本的な機能です。中核機能には、以下の機能があります。

機能	文
データの転記	MOVE文
算術演算	COMPUTE文、ADD文、DIVIDE文、MULTIPLY文、SUBTRACT文
選択処理	IF文、EVALUATE文、CONTINUE文
分岐	GO TO文、ALTER文
繰返し処理	PERFORM文、EXIT文、EXIT PERFORM文
表操作	SEARCH文、SET文
データ項目の初期化	INITIALIZE文
文字列操作	INSPECT文、STRING文、UNSTRING文
小入出力	ACCEPT文、DISPLAY文
プログラムの実行の終了	STOP RUN文
ポインタの操作	該当文なし
浮動小数点数の操作	該当文なし

中核機能を使うプログラムでは、必要に応じて、下図に示す記述をします。

プログラムの記述			主な機能
見出し部			プログラムの名前を指定する。
環境部			
	構成部		
		翻訳用計算機段落	デバッグモードを指定する。
		実行用件三期段落	文字の大小順序を指定する。
		特殊名段落	呼び名、符号系名、記号文字、字類名、通貨編集用記号、小数点などを定義する。

プログラムの記述			主な機能	
			記号定数、位置決め単位名、印字モード名などを定義する。	
データ部				
	基底場所節			
		独立データ記述項・レコード記述項	ポインタ付けして参照するデータ項目を定義する。	
	作業場所節			
		独立データ記述項・レコード記述項	データ項目を定義する。	
	定数節			
		独立データ記述項・レコード記述項	値が一定のデータ項目を定義する。	
手続き部				
	手続き部分			
		MOVE文	転記する。	
		COMPUTE文	算術演算を行う。	
		ADD文	加算を行う。	算術演算
		DIVIDE文	除算を行う。	
		MULTIPLY文	乗算を行う。	
		SUBTRACT文	減算を行う。	
		IF文	条件の真理値に従って処理を振り分ける。	選択処理
		EVALUATE文	複数の条件の真理値に従って処理を振り分ける。	
		CONTINUE文	無操作文。	
		GO TO文	手続き部の別の場所へ制御を移す。	分岐
		ALTER文	GO TO文の分岐先を変更する。	
		PERFORM文	手続きを繰り返し実行する。	繰り返し処理
		EXIT文	そとPERFORM文の共通の出口を指定する。	
		EXIT PERFORM文	うちPERFORM文の共通の出口を指定する。	
		SEARCH文	表要素を検索する。	表操作
		SET文	表要素の指標を設定する。	
		INITIALIZE文	データ項目を初期化する。	
		INSPECT文	文字列を検査する。	文字列操作
		STRING文	文字列を連結する。	
		UNSTRING文	文字列を分解する。	
		ACCEPT文	データをハードウェア装置から入力する。	小入出力
		DISPLAY文	データをハードウェア装置に表示する。	
		STOP RUN文	プログラムの実行を終了する。	
プログラム終わり見出し			プログラムの終わりを指定する。	

2.1.1 データの転記

データ項目または定数の内容を別のデータ項目に転記するためには、MOVE文を使います。MOVE文では、受取り側項目の属性に合わせて、桁合わせ、データの型の変換、編集などが行われます。

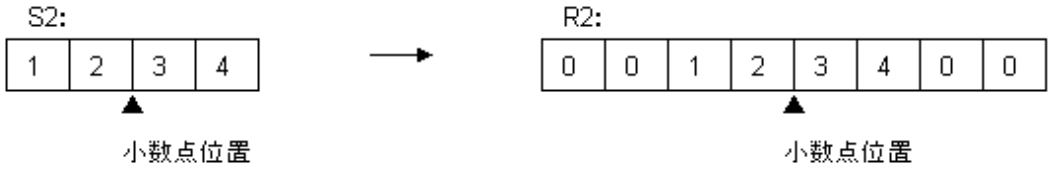
例

MOVE文の例

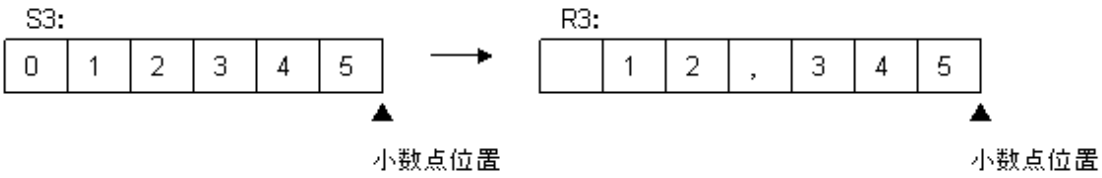
```
*〔データ部〕
  77  S1  PIC X(4).
  77  R1  PIC X(6).
  77  S2  PIC 9(2)V99.
  77  R2  PIC 9(4)V9(4).
  77  S3  PIC 9(6).
  77  R3  PIC ZZZ,ZZ9.
*〔手続き部〕
      MOVE S1 TO R1.        …[1]
      MOVE S2 TO R2.        …[2]
      MOVE S3 TO R3.        …[3]
```

[1]のMOVE文では、S1がR1に左詰めで転記され、残りの部分には空白が補われます。(文字項目の転記)

[2]のMOVE文では、S2とR2の小数点位置を合わせて転記されます。(数字項目の転記)



[3]のMOVE文では、S3とR3の小数点位置を合わせて編集転記されます。先行ゼロ列が空白で置き換えられ、コンマが挿入されます。(数字編集項目の転記)



2.1.2 算術演算

加算を行うためにはADD文、減算を行うためにはSUBTRACT文、乗算を行うためにはMULTIPLY文、除算を行うためにはDIVIDE文、算術演算を行うためにはCOMPUTE文を使います。

COMPUTE文では、ブール演算を行うこともできます。

ADD文、SUBTRACT文、MULTIPLY文、DIVIDE文およびCOMPUTE文を総称して、「算術文」といいます。算術文では、算術演算の結果の四捨五入、および桁あふれ条件の検査を行うことができます。四捨五入するためには、ROUNDED指定を書きます。桁あふれ条件の検査を行うためには、ON SIZE ERROR指定を書きます。

例

算術文の例

```
    ADD C TO A.                          …[1]
    DIVIDE C BY D GIVING A ROUNDED.      …[2]
    COMPUTE A = C / D + E * 100.         …[3]
    COMPUTE Z = X AND Y.                 …[4]
```

[1] A＋Cの結果を、Aに格納します。

[2] C／Dの商を四捨五入して、Aに格納します。

[3] C／D＋E＊100の結果を、Aに格納します。

[4] XとYのブール積を、Zに格納します。

2.1.3 選択処理と分岐

選択処理を行うためには、IF文またはEVALUATE文を使います。IF文は、条件を検査し、その真理値により次の実行文を選択します。EVALUATE文は、条件を検査し、多数の処理のうちの1つを選択します。

ある場所から別の場所へ無条件に分岐するためには、GO TO文を使います。

例

IF文の例

IF文では、条件が真の場合の処理をTHEN指定に書き、条件が偽の場合の処理をELSE指定に書きます。IF文の例を以下に示します。

```
    IF X = Y THEN
      MOVE 0 TO Z            …[1]
    ELSE
      GO TO P1               …[2]
    END-IF.
        :
P1.
        :
```

[1] X＝Yが真の場合、MOVE文を実行します。

[2] X＝Yが偽の場合、GO TO文を実行し、P1に分岐します。

例

EVALUATE文の例

EVALUATE文では、選択すべき処理をWHEN指定に書きます。EVALUATEの直後に書いた対象とWHENの直後に書いた対象が比較され、条件を満足する場合、WHEN指定の中に書いた処理が実行されます。EVALUATE文の例を以下に示します。

```
    EVALUATE MARKS
      WHEN 85 THRU 100 MOVE "A" TO RESULT       …[1]
      WHEN 70 THRU  84 MOVE "B" TO RESULT       …[2]
      WHEN 55 THRU  69 MOVE "C" TO RESULT       …[3]
      WHEN OTHER       MOVE "D" TO RESULT       …[4]
    END-EVALUATE.
```

[1] MARKSの値が85～100の場合、MOVE "A" TO RESULTを実行します。

[2] MARKSの値が70～84の場合、MOVE "B" TO RESULTを実行します。

[3] MARKSの値が55～69の場合、MOVE "C" TO RESULTを実行します。

[4] MARKSが[1]～[3]のどの条件も満足しない場合、MOVE ”D” TO RESULTを実行します。

上記のEVALUATE文は、条件名条件を使って、以下のように書くこともできます。

```
*〔データ部〕
  01  MARKS PIC 9(3).
      88  RESULT-A VALUE 85 THRU 100.
      88  RESULT-B VALUE 70 THRU 84.
      88  RESULT-C VALUE 55 THRU 69.
*〔手続き部〕
     EVALUATE TRUE
       WHEN RESULT-A MOVE "A" TO RESULT
       WHEN RESULT-B MOVE "B" TO RESULT
       WHEN RESULT-C MOVE "C" TO RESULT
       WHEN OTHER    MOVE "D" TO RESULT
     END-EVALUATE.
```

2.1.4 繰返し処理

一連の処理を繰り返すためには、PERFORM文を使います。PERFORM文には、そとPERFORM文と、うちPERFORM文があります。

そとPERFORM文

そとPERFORM文では、繰り返して実行する文を手続き名で指定します。そとPERFORM文は、手続きを繰り返して実行するだけでなく、プログラムの制御を1箇所で制御するために使うこともできます。

例

そとPERFORM文の例

```
      MOVE 0 TO TBL-SUM.
      PERFORM P1 VARYING I FROM 1 BY 1 UNTIL I > 5
                 AFTER J FROM 1 BY 1 UNTIL J > 10         *>...[1]
      *>    :
      PERFORM P2 THRU PX.                                 *>...[2]
      *>      :
P1.                                       *> ↑ [1]のPERFORM文の範囲
      ADD TBL-X (I, J) TO TBL-SUM.        *> ↓
P2.                                       *>     ↑
                                          *>     | [2]のPERFORM文の範囲
PX.                                       *>     |
      EXIT.                               *>     ↓
```

[1] 段落名P1にある文を、繰り返し実行します。Jの初期値を1として、P1にある文を実行するごとにJの値を1ずつ増やし、Jの値が10を超えたときにIの値 (初期値は1)を1だけ増やします。この処理を、Iの値が5を超えるまで繰り返します。

[2] 段落名P2からPXにある文を1回だけ実行します。EXIT文は、手続きの出口を示すためだけの文です。

うちPERFORM文

うちPERFORM文では、繰り返して実行する文をPERFORM文の中に書きます。

例

うちPERFORM文の例

このPERFORM文の実行結果は、前述のそとPERFORM文の[1]の実行結果と同じです。

```
MOVE 0 TO TBL-SUM.
PERFORM VARYING I FROM 1 BY 1 UNTIL I > 5          …[1]
  PERFORM VARYING J FROM 1 BY 1 UNTIL J > 10       …[2]
    ADD TBL-X (I, J) TO TBL-SUM
  END-PERFORM
END-PERFORM.
```

[1] UNTIL指定の後に書いた無条件文(PERFORM文)を、繰り返し実行します。Iの初期値を1として、無条件文を実行するごとにIの値を1ずつ増やし、Iの値が5を超えたときに繰返しを終了します。

[2] UNTIL指定の後に書いた無条件文(ADD文)を繰り返し実行します。Jの初期値を1として、無条件文を実行するごとにJの値を1ずつ増やし、Jの値が10を超えたときに繰返しを終了します。

2.1.5 表操作

同じ属性を持つデータ項目を繰り返し定義する場合、それらをまとめて表として定義することができます。表を定義するためには、データ記述項でOCCURS句を書きます。表の中の要素を、「表要素」といいます。表要素は、添字を付けて参照します。

表要素の反復回数

表要素の反復回数は、固定にすることも可変にすることもできます。反復回数を固定にする場合、OCCURS句でDEPENDING指定を省略します。反復回数を可変にする場合、OCCURS句にDEPENDING指定を書きます。この場合、OCCURS句で、反復回数の最小値と最大値を指定します。表要素の反復回数は、DEPENDING指定に書いたデータ項目に値を設定することによって指定します。OCCURS句にDEPENDING指定を指定したデータ項目を、「可変反復データ項目」といいます。

レコードの中には、可変反復データ項目の後に別のデータ項目を定義することができます。レコードの中で、可変反復データ項目の後に割り付けられる領域を、「レコードの中の可変位置」といいます。

表の定義例

```
01  A.
    02  A1 PIC X.
    02  A2 PIC X(2) OCCURS 2.                        …[1]
    02  A3.
        03  A31 OCCURS 1 TO 3 DEPENDING ON A31-REP.  …[2]
            04  A311 PIC X(4).
            04  A312 PIC X(5) OCCURS 2.              …[3]
        03  A32 PIC X(3).                            …[4]
01  A31-REP PIC 9.                                   …[5]
```

[1] 反復回数が固定の表

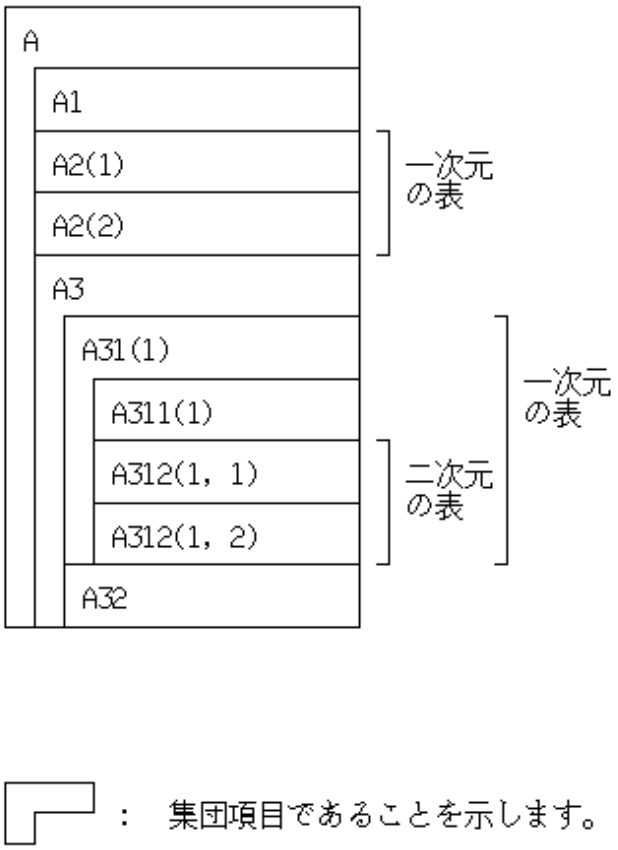
[2] 反復回数が可変の表

[3] 反復回数が固定の表

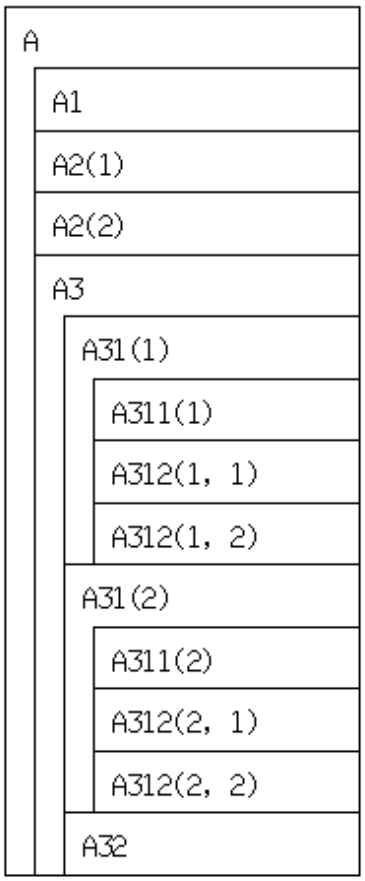
[4] レコードの可変位置

[5] 反復回数を設定するためのデータ項目

〔A31-REP の値が１の場合のA の構成〕



〔A31-REP の値が２の場合のA の構成〕



参考

(n)および(n,m)は、添字です。A312(n,m)の場合、nはA31のデータ記述項に書いたOCCURS句に対応し、mはA312のデータ記述項に書いたOCCURS句に対応します。

可変反復データ項目の参照領域

可変反復データ項目を従属する集団項目を参照する文を実行した場合、可変反復データ項目の領域のうち、以下の領域が処理の対象になります。以下で、XはOCCURS句のDEPENDING指定に書いたデータ名を表し、nは処理が開始されるときのXの値を表します。

- その集団項目がXを従属しない場合、第1番目から第n番目までの表要素が処理の対象になります。
- その集団項目がデータ名-1を従属する場合、その集団項目が送出し側項目として用いられるときは、第1番目から第n番目までの表要素が処理の対象になります。その集団項目が受取り側項目として用いられるときは、反復回数の最大値が、処理の対象になります。

指標名と指標データ項目

表要素に付ける添字として、整数、整数項目または指標名を使うことができます。添字に整数または整数項目を書いた場合、その値は表要素の出現番号を表します。添字に指標名を書いた場合、その値は、表要素の記憶場所を識別するための値を表します。

指標名は、表要素を識別するために使います。指標名は、OCCURS句のINDEXED BY指定で指定します。指標名の領域は、コンパイラによって生成されます。添字に書く指標名は、その指標名を指定した表に含まれる指標名でなければなりません。指標名の値は、SET文で設定します。

指標名の値は、SET文を使って、指標データ項目に退避することができます。指標データ項目は、指標名の値をそのまま格納するためのデータ項目です。指標データ項目は、USAGE IS INDEX句を書いたデータ記述項で定義します。

例

指標名と指標データ項目の使用例

```
*〔データ部〕
  77  B USAGE INDEX.
  01  C.
      02  C1 OCCURS 2 INDEXED BY IDX1.
          03  C11 PIC X(2).
          03  C12 PIC X(3) OCCURS 3 INDEXED BY IDX2.
*〔手続き部〕
     SET IDX1 TO 1.                    …[1]
     SET B    TO IDX1.                 …[2]
     SET IDX2 TO 2.                    …[3]
     MOVE C12(IDX1, IDX2) TO ～.       …[4]
```

[1] 出現番号1に対応する値を、表C1の指標名IDX1に設定します。

[2] 指標名IDX1の値を、指標データ項目Bに退避します。

[3] 出現番号2に対応する値を、表C12の指標名IDX2に設定します。

[4] 表C12の表要素を、IDX1とIDX2で添字付けして参照します。

表の検索

ある条件を満足する表要素を検索するためには、SEARCH文を使います。表要素の値が既に昇順または降順に並んでいる場合は、ALL指定付きのSEARCH文を使うことができます。

例

ALL指定付きのSEARCH文の例

```
    01 TBL-VALUE.                                *> -+  …[1]
      02 FILLER .                                *>  |
        03 FILLER PIC X(02) VALUE "01".          *>  |
        03 FILLER PIC X(20) VALUE "APPLE".       *>  |
      02 FILLER.                                 *>  |
        03 FILLER PIC X(02) VALUE "10".          *>  |
        03 FILLER PIC X(20) VALUE "ORANGE".      *>  |
      02 FILLER.                                 *>  |
        03 FILLER PIC X(02) VALUE "12".          *>  |
        03 FILLER PIC X(20) VALUE "PEACH".       *> -+
    01 TBL-REF REDEFINES TBL-VALUE.                          *> -+  …[2]
      02 TBL-DATA OCCURS 3 TIMES ASCENDING                   *>  |
                 KEY IS G-CODE INDEXED BY IDX.  *>     …[3]      |
        03 G-CODE PIC X(02).                     *> -+  …[4]      |
        03 G-NAME PIC X(20).                     *> -+           -+
PROCEDURE DIVISION.
      SEARCH ALL TBL-DATA                       *> -+  …[5]
        AT END DISPLAY "ERROR"                  *>  |
          WHEN G-CODE (IDX) = "10"              *>  |
            DISPLAY G-NAME (IDX)                *>  |
      END-SEARCH.                               *> -+
```

[1] 表TBL-VALUEの値を定義します。

[2] 表TBL-REFを定義します。TBL-REFは、TBL-VALUEと同じ領域を占めます。

[3] 表TBL-DATAを定義します。反復回数を3とし、TBL-DATAに指標名IDXを関係付け、ASCENDING KEY指定で、G-CODEの値が昇順に並んでいることを指定します。

[4] TBL-DATAに従属する表要素として、G-CODEおよびG-NAMEを定義します。

[5] WHEN指定に書いた条件を満足する表要素を検索し、見つかった場合、WHEN指定の条件の後の無条件文(DISPLAY文)を実行します。見つからなかった場合、AT END指定の無条件文(DISPLAY文)を実行します。

2.1.6 データ項目の初期化

データ項目を初期化するためには、INITIALIZE文を使います。

例

INITIALIZE文の例

```
*〔データ部〕
  01  GRP1.
      02  A PIC X(8).
      02  B PIC S9(4) PACKED-DECIMAL.
      02  C PIC N(4).
*〔手続き部〕
     INITIALIZE GRP1.                              …[1]
     INITIALIZE GRP1 REPLACING NUMERIC DATA BY 1.  …[2]
```

[1] GRP1に従属する基本項目を初期化します。Aに空白、Bにゼロ、Cに日本語の空白が設定されます。

[2] GRP1に従属する基本項目のうち、数字項目だけを初期化します。Bに1が設定され、AおよびCには何も設定されません。

2.1.7 文字列操作

文字列の出現回数を数えたり文字列を置き換えたりするためには、INSPECT文を使います。文字列を連結するためにはSTRING文、文字列を分解するためにはUNSTRING文を使います。

例

INSPECT文の例

```
*〔データ部〕
  77  A PIC X(10) VALUE "AB*DE*FGH*".
  77  B PIC 99.
*〔手続き部〕
     INSPECT A TALLYING B FOR ALL "*".          …[1]
```

[1] Aの内容を検査し、“＊”の出現回数を数えます。INSPECT文の実行前のBの値が0の場合、INSPECT文の実行後のBの値は3になります。

例

STRING文の例

```
*〔データ部〕
  77  A PIC X(5) VALUE "ABCDE".
  77  B PIC X(3) VALUE "123".
  77  C PIC X(2) VALUE "+-".
  77  D PIC X(10).
*〔手続き部〕
     STRING A DELIMITED BY SIZE
```

```
            B DELIMITED BY SIZE
            C DELIMITED BY SIZE
            INTO D.                          …[1]
```

[1] A、BおよびCの内容をこの順に連結して、Dに格納します。STRING文の実行後のDの内容は、“ABCDE123+-”になります。

例

UNSTRING文の例

```
*〔データ部〕
  77  A PIC X(5).
  77  B PIC X(3).
  77  C PIC X(2).
  77  D PIC X(10) VALUE "ABCDE123+-".
*〔手続き部〕
     UNSTRING D INTO A B C.              …[1]
```

[1] Dの内容をA、BおよびCの大きさで区切り、区切った順に格納します。UNSTRING文の実行後、A、BおよびCには、それぞれ、“ABCDE”、“123”および“+-”が格納されます。

2.1.8 小入出力

少量のデータをハードウェア装置から入力するためにはACCEPT文、少量のデータをハードウェア装置に出力するためにはDISPLAY文を使います。ACCEPT文またはDISPLAY文を使う場合、ファイルを定義する必要はありません。ACCEPT文では、日付や時刻を得ることもできます。

例

ACCEPT文およびDISPLAY文の例

```
     DISPLAY "ERROR".                    …[1]
     ACCEPT X.                           …[2]
     ACCEPT C-TIME FROM TIME.            …[3]
```

[1] 文字定数("ERROR")を表示します。

[2] Xの内容を入力します。

[3] 現在の時刻をC-TIMEに格納します。

2.1.9 プログラムの実行の終了

プログラムの実行を終了するためには、STOP RUN文を使います。STOP RUN文は、実行単位の終了を表します。

2.1.10 ポインタの操作

任意のアドレスの領域を参照するためには、ポインタを使います。

領域のアドレスは、ADDR関数によって得ることができ、ポインタデータ項目に設定することができます。ポインタデータ項目は、USAGE IS POINTER句を書いたデータ記述項で定義します。

ポインタを使って参照する領域の構造属性は、データ部の基底場所節で定義します。基底場所節で定義したデータ項目の領域は確保されません。

基底場所節のデータ記述項では、作業場所節のデータ記述項に記述できる句のほかにBASED ON句を書くことができます。BASED ON句は、暗黙のポインタを指定するために書きます。

基底場所節で定義したデータ項目を参照する場合、データ名にポインタ修飾子を付けるか、または暗黙のポインタによって、ポインタ付けしなければなりません。

例

ポインタを操作する例

```
*〔データ部〕
  BASED-STORAGE SECTION.
  77  AA PIC X    BASED ON P1.          …[1]
  77  BB PIC 9(4) BINARY.
  WORKING-STORAGE SECTION.
  77  CC PIC X(100).
  77  P3 USAGE IS POINTER.              …[2]
  LINKAGE SECTION.
  01  P1 USAGE IS POINTER.              …[3]
  01  P2 USAGE IS POINTER.              …[4]
*〔手続き部〕
 PROCEDURE DIVISION USING P1 P2.
     IF AA = "A" THEN                   …[5]
       MOVE ZERO TO P2->BB              …[6]
      :
     MOVE FUNCTION ADDR(CC) TO P3.      …[7]
     CALL "SUB" USING P3.
      :
```

[1] 暗黙のポインタP1を指定します。

[2] ポインタデータ項目P3を定義します。

[3] ポインタデータ項目P1を定義します。

[4] ポインタデータ項目P2を定義します。

[5] AAを暗にポインタ付けして参照します。

[6] BBを明にポインタ付けして参照します。

[7] ADDR関数を使ってCCのアドレスを求めます。

2.1.11 浮動小数点数の操作

浮動小数点数は、“仮数＊(10＊＊指数)”の形式で表現される数値データです。浮動小数点数には、浮動小数点項目と浮動小数点定数があります。

[Win16][Win32][WinIPF][Winx64][.NET]の場合、浮動小数点数の範囲は、単精度の場合、絶対値が約1.18×(10の－38乗)より大きく、約3.4×(10の＋38乗)を超えない数、または0です。倍精度の場合、絶対値が約2.23×(10の－308乗)より大きく、約1.79×(10の＋308乗)を超えない数、または0です。

[DS][HP][Solaris]の場合、浮動小数点項目の範囲は、絶対値が1.18×(10の－38乗)より大きく、7.2×(10の＋75乗)を超えない数、または0です。

[Linux][LinuxIPF][Linux64]の場合、浮動小数点数の範囲は、絶対値が約1.18×(10の－38乗)より大きく、約3.4×(10の＋38乗)を超えない数、または0です。

浮動小数点項目は、以下で“データ名”または“一意名”と示しているところに書くことができ、浮動小数点定数は、以下で“定数”と示しているところに書くことができます。

環境部

- SYMBOLIC CONSTANT句の定数

データ部

- VALUE句の定数
  ただし、VALUE句を指定したデータ項目は、内部浮動小数点項目でなければなりません。
- REDEFINES句およびRENAMES句のデータ名

手続き部

- 条件式中の一意名/定数
  ただし、比較条件の中にだけ書くことができます。
- ADD文の各一意名/定数
- COMPUTE文の各一意名/定数
- DISPLAY文の一意名/定数

  浮動小数点定数または外部浮動小数点項目を書いた場合、変換されることなくそのまま表示されます。

  内部浮動小数点項目を書いた場合、以下の形式の外部浮動小数点項目に変換され、表示されます。

  ー 単精度の場合:　-.9(8)E-99

  ー 倍精度の場合:　-.9(17)E-99
- REMAINDER指定なしのDIVIDE文の各一意名/定数
- INITIALIZE文の各一意名/定数

  INITIALIZE文の初期化対象として浮動小数点項目を指定した場合、浮動小数点項目は以下のように初期化されます。

  ー REPLACING指定なしのINITIALIZE文の初期化対象として浮動小数点項目を指定した場合、浮動小数点項目は0に初期化されます。

  ー REPLACING指定にNUMERICを書いた場合、浮動小数点項目も初期化の対象になります。
- MOVE文の各一意名/定数

  転記において、浮動小数点項目は非整数の数字項目と同様に扱われます。

  注意

  内部浮動小数点の精度は、外部浮動小数点および固定小数点の精度と異なるので、相互に転記を行う場合には、それぞれの共通範囲で使用しなければなりません。
- MULTIPLY文の各一意名/定数
- SUBTRACT文の各一意名/定数
- PROCEDURE DIVISIONのUSING指定のデータ名
- ENTRY文のUSING指定のデータ名
- CALL文のUSING指定の各一意名/定数(ただし、USING BY VALUE指定には書けません。)
- COPY文のREPLACING指定の各一意名/定数
- 引数の型が数字の、数字関数の引数

注意

浮動小数点数はアーキテクチャに依存する機能であり、システム毎に内部形式が異なります。また、浮動小数点数は内部的には2進値により値を表現しているため、全ての値が表現できるわけではなく、その近似値で表されています。従って、ほとんどの場合誤差を含んでいます。浮動小数点数を使用する場合は、これらのことを考慮して使用してください。

2.2 入出力機能

入出力機能には、順ファイル機能、相対ファイル機能および索引ファイル機能があります。

順ファイル機能は、ファイル中のレコードを決まった順序で処理する機能です。順ファイル機能で処理するファイルを、「順ファイル」といいます。順ファイル中の各レコードは、レコードを書き出したときの位置によって識別されます。

相対ファイル機能は、ファイル中の任意のレコードを処理したり、ファイル中のレコードを決まった順序で処理したりする機能です。相対ファイル機能で処理するファイルを、「相対ファイル」といいます。相対ファイル中の各レコードは、相対レコード番号によって識別されます。

索引ファイルは、ファイル中の任意のレコードを処理したり、ファイル中のレコードを決まった順序で処理したりする機能です。索引ファイル機能で処理するファイルを、「索引ファイル」といいます。索引ファイル中の各レコードは、1つまたは複数のキーの値によって識別されます。

入出力機能を使うプログラムでは、必要に応じて、下図に示す記述をします。

プログラムの記述	主な機能
環境部	
入出力節	
ファイル管理段落	
ファイル管理記述項	ファイル名を外部媒体と関係付ける。編成、呼出し法、相対キー、主レコードキー、副レコードキー、ロックモードなどを指定する。
入出力管理段落	複数のファイルで共用する記憶領域を指定する。
データ部	
ファイル節	
ファイル記述項	ファイルの物理的な構造を定義する。
レコード記述項	レコードの構造を定義する。
手続き部	
宣言部分	
USE文およびUSE AFTER STANDARD EXCEPTION手続き	入出力誤り手続きを定義する。
手続き部分	
OPEN文	ファイルを開く。
READ文	ファイルからレコードを読み込む。
WRITE文	ファイルにレコードを書き出す。
REWRITE文	ファイル中のレコードを書き換える。
DELETE文 (*1)	ファイル中のレコードを削除する。
START文 (*1)	ファイルを論理的に位置づける
UNLOCK文	レコードのロックを解除する。
CLOSE文	ファイルを閉じる。

*1 ： DELETE文およびSTART文は、順ファイル機能にはありません。

2.2.1 ファイルの編成

外部媒体上のファイルの論理的な構造を、「編成」といいます。ファイルの編成には、以下の4種類があります。どの編成のファイルを使うかは、ファイル管理記述項のORGANIZATION句で指定します。

- 順編成

- 行順編成
- 相対編成
- 索引編成

順ファイル機能では、以下のファイルを使うことができます。

- 順編成
- 行順編成

相対ファイル機能では、相対編成のファイルを使うことができます。

索引ファイル機能では、索引編成のファイルを使うことができます。

順編成のファイル

順編成のファイルは、レコードの直前直後の位置によって識別されるレコードで構成されます。レコードの論理的な位置は、レコードを書き出したときに決まり、ファイルの終わりにレコードを追加するときを除いて変わることはありません。

順編成のファイルを使う場合、ORGANIZATION句にSEQUENTIAL指定を書くか、またはORGANIZATION句を省略します。

順編成のファイルには、レコード順ファイルと印刷ファイルがあります。レコード順ファイルは、外部記憶装置で入出力操作が可能なファイルです。印刷ファイルは、印刷装置に書き出すためのファイルです。レコード順ファイルと印刷ファイルのどちらのファイルを使うかは、ASSIGN句、WRITE文、レコード記述項の記述などで決まります。決定規則の詳細については、“NetCOBOL使用手引書”を参照してください。

印刷ファイルでは、レコードを帳票定義体に従って編集して出力することもできます。帳票定義体を使う場合、ファイル管理記述項のFORMAT句で、帳票定義体名を設定するデータ項目を指定します。

FORMAT句を指定した印刷ファイルを「FORMAT句付きの印刷ファイル」といい、FORMAT句を指定しない印刷ファイルを「FORMAT句なしの印刷ファイル」といいます。

行順編成のファイル

行順編成のファイルは、区切り文字で区切られたレコードで構成されます。1つのレコードは1行と数え、1行は印刷可能な文字とレコードの区切り文字で構成されます。レコードの論理的な位置は、レコードを書き出したときに決まり、ファイルの終わりにレコードを追加するときを除いて変わることはありません。

行順編成のファイルを使う場合、ORGANIZATION句にLINE SEQUENTIAL指定を書きます。

相対編成のファイル

相対編成のファイルは、相対レコード番号によって識別されるレコードで構成されます。相対レコード番号は、1以上の整数値です。相対ファイル中のレコードは、レコードを書き出すときに、相対レコード番号が示す位置に格納されます。例えば、相対レコード番号を10として書き出したレコードは、相対レコード番号が1から9までの位置にレコードが格納されていなくても、相対レコード番号が10の領域に格納されます。

相対編成のファイルを使う場合、ORGANIZATION句にRELATIVE指定を書きます。レコードを乱に処理する場合は、ACCESS MODE句のRELATIVE KEY指定で、相対キー項目を指定しなければなりません。

索引編成のファイル

索引編成のファイルは、レコードを識別するための索引およびレコードで構成されます。索引編成のファイルは、レコード中のある領域(キー)の値によって識別されます。

索引編成のファイルを使う場合、ORGANIZATION句にINDEXED指定を書きます。キーは、RECORD KEY句またはALTERNATE RECORD KEY句で指定します。RECORD KEY句では主レコードキーを指定し、ALTERNATE RECORD KEY句では副レコードキーを指定します。主レコードキーは、必ず指定しなければなりません。

主レコードキーと副レコードキーを総称して、「レコードキー」といいます。レコードキーの値は、ファイル中で一意にすることも重複を許すこともできます。レコードキーの値の重複を許すためには、以下の句にDUPLICATES指定を書きます。

- RECORD KEY句
- ALTERNATE RECORD KEY句

索引ファイル中のレコードを呼び出すために使われるレコードキーを、「参照キー」といいます。参照キーは、入出力文の実行によって変化します。

2.2.2 ファイル結合子

外部媒体上のファイルとプログラムとの関係付けは、ファイル結合子を通して行われます。ファイル結合子は、ファイルについての情報を持つ記憶領域です。ファイル結合子は、以下の目的で使われます。

- ファイル名と外部媒体上のファイルとの間の連絡を行うため。
- ファイル名と、ファイルに関係付けたレコード領域の間の連絡を行うため。

ファイル管理記述項のSELECT句とASSIGN句を書くことにより、ファイル結合子がファイル名および外部媒体上のファイルに関係付けられます。OPEN文を実行すると、ファイル結合子が確立されます。

2.2.3 入出力文の動作

ファイルに対する処理は、入出力文(OPEN文、READ文、WRITE文、REWRITE文、START文、DELETE文およびCLOSE文)を組み合わせて書きます。

ファイルに対する処理を行うためには、最初にOPEN文を実行してファイルを開かなければなりません。ファイルに対する処理を終了した後、CLOSE文を実行してファイルを閉じます。OPEN文とCLOSE文の間に実行することができる入出力文は、呼出し法とオープンモードに依存します。

呼出し法

ファイル中のレコードを処理する順番を、「呼出し法」といいます。呼出し法には、順呼出し法、乱呼出し法および動的呼出し法の3種類があります。順呼出し法は、一定の順序でレコードを書き出したり読み込んだりする方法です。乱呼出し法は、任意の順序でレコードを書き出したり読み込んだりする方法です。動的呼出し法は、順呼出し法と乱呼出し法を、入出力文の実行の仕方によって切り換える方法です。

呼出し法は、ファイル管理記述項のACCESS MODE句で指定します。順ファイルでは、順呼出し法だけを指定することができます。

オープンモード

ファイルを開くときのモードを「オープンモード」といいます。オープンモードには、入力モード、出力モード、入出力両用モードおよび拡張モードの4種類があります。どのモードでファイルを開くかは、OPEN文で指定します。

順ファイルのオープンモードと入出力文の関係

下表に、順ファイルのオープンモードと入出力文の関係を示します。

入出力文	オープンモード（括弧内は、OPEN文の指定を示す）			
	入力モード（INPUT 指定）	出力モード（OUTPUT 指定）	入出力両用モード（I-O 指定）	拡張モード（EXTEND 指定）
READ文	○	－	○	－
WRITE 文	－	○	－	○
REWRITE文	－	－	○	－

○ ： 実行可能
－ ： 実行不可能

相対ファイルおよび索引ファイルのオープンモードと入出力文の関係

下表に、相対ファイルおよび索引ファイルのオープンモードと入出力文の関係を示します。

呼出し法	入出力文	オープンモード（括弧内は、OPEN文の指定を示す）			
		入力モード（INPUT指定）	出力モード（OUTPUT指定）	入出力両用モード（I-O指定）	拡張モード（EXTEND指定）
順呼出し法	READ文	○	－	○	－
	WRITE文	－	○	－	○
	REWRITE文	－	－	○	－
	START文	○	－	○	－
	DELETE文	－	－	○	－
乱呼出し法	READ文	○	－	○	－
	WRITE文	－	○	○	－
	REWRITE文	－	－	○	－
	START文	－	－	－	－
	DELETE文	－	－	○	－
動的呼出し法	READ文	○	－	○	－
	WRITE文	－	○	○	－
	REWRITE文	－	－	○	－
	START文	○	－	○	－
	DELETE文	－	－	○	－

○ ： 実行可能
－ ： 実行不可能

2.2.4 ファイル位置指示子

ファイル位置指示子は、一連の入出力操作において、次に処理するレコードを定めるための、概念上の指示子です。ファイル位置指示子は、入力モードまたは入出力両用モードで開いたファイルでだけ意味があります。

順ファイルのファイル位置指示子の状態は、CLOSE文、OPEN文およびREAD文の実行によって、影響を受けます。

相対ファイルおよび索引ファイルのファイル位置指示子の状態は、CLOSE文、OPEN文、READ文およびSTART文の実行によって、影響を受けます。

2.2.5 ボリューム指示子

ボリューム指示子は、現在の物理ボリュームを一意にするための、概念上の指示子です。ボリューム指示子は、順ファイルだけに存在する概念です。ボリューム指示子の状態は、CLOSE文、OPEN文、READ文およびWRITE文の実行によって、影響を受けます。

2.2.6 ファイルの共用と排他

大記憶装置の物理ファイルに対して、同時に複数の実行単位で使用可能にするか(共用モード)、または同時に1つの実行単位でだけ使用可能にするか(排他モード)という属性を、COBOLプログラムで与えることができます。

物理ファイルを共用モードと排他モードのどちらで使うかは、ファイル管理記述項のLOCK MODE句とOPEN文で指定します。LOCK MODE句では、翻訳単位中のOPEN文で、ファイル名に関係付けた物理ファイルを共用モードと排他モードのどちらで開くことを可能にするかを指定します。LOCK MODE句で共用モードで開くことが可能と指定したファイルは、共用モードで開くことも排他モードで開くこともできます。LOCK MODE句で、排他モードで開くことが可能と指定したファイルは、排他モードでだけ開くことができます。

OPEN文を実行すると、物理ファイルを共用モードと排他モードのどちらのモードで使うかが決まります。

共用モードで開いたファイルには、共用モードで開くごとに別のファイル結合子が関係付けられます。したがって、1つの物理ファイルに、同時に2つ以上のファイル結合子が関係付けられます。

排他モードで開いたファイルには、同時に1つのファイル結合子だけが関係付けられます。排他モードで開いたファイルはロックされ、別の実行単位で開くことができなくなります。排他モードで開いたファイルに対してCLOSE文を実行すると、ファイルのロックは解除され、別の実行単位で開くことができるようになります。

物理ファイルが共用モードと排他モードのどちらのモードで開かれるかを、下表に示します。

LOCK MODE 句の指定	OPEN文の指定							
	WITH LOCK指定あり				WITH LOCK指定なし			
	INPUT	I-O	OUTPUT	EXTEND	INPUT	I-O	OUTPUT	EXTEND
なし	排他モード				共用モード	排他モード		
AUTOMATIC					共用モード		排他モード	共用モード
MANUAL (*1)					共用モード		排他モード	共用モード
EXCLUSIVE					排他モード			

*1 ： 順ファイルでは、MANUALを指定することはできません。

2.2.7 レコードのロック

共用モードかつ入出力両用モードで開いたファイルでは、READ文で読み込んだレコードに、ロックを設定することができます。

順ファイルの場合、1つの物理ファイルの中の1つのレコードだけに、同時にロックを設定することができます。相対ファイルおよび索引ファイルの場合、1つの物理ファイルの中の1つまたは2つ以上のレコードに、同時にロックを設定することができます。1つのレコードにだけにロックを設定する場合、LOCK MODE句でAUTOMATICを指定します。複数のレコードにロックを設定する場合、LOCK MODE句でMANUALを指定します。

LOCK MODE句でAUTOMATICを指定した場合のレコードのロック

LOCK MODE句でAUTOMATICを指定し入出力両用モードで開いたファイルのファイル結合子は、同時に1つのレコードのロックを保持することができます。WITH LOCK指定付きまたはWITH[NO]LOCK指定なしのREAD文を実行すると、読み込んだレコードにロックが設定され、ファイル結合子のロックの情報が更新されます。WITH NO LOCK指定付きのREAD文を実行した場合は、レコードのロックは設定されません。

WITH LOCK指定付きまたはWITH[NO]LOCK指定なしのREAD文を実行すると、ロックされたレコードに対して、別のファイル結合子によるロックを設定するREAD文、WRITE文、REWRITE文またはDELETE文の実行ができなくなります。ただし、拡張モードで開いたファイルのファイル結合子によるWRITE文の実行には影響を与えません。

設定されたレコードのロックは、READ文を実行したときと同じファイル結合子に対して、以下のいずれかの文を実行したときに解除されます。

- READ文、WRITE文、REWRITE文、DELETE文またはSTART文を実行したとき
- UNLOCK文を実行したとき
- CLOSE文を実行したとき（CLOSE文が暗黙に実行されたときも、ロックが解除されます）

LOCK MODE句でMANUALを指定した場合のレコードのロック

LOCK MODE句でMANUALを指定し入出力両用モードで開いたファイルのファイル結合子は、同時に複数のレコードのロックを保持することができます。WITH LOCK指定付きのREAD文を実行すると、読み込んだレコードにロックが設定され、その情報がファイル結合子に追加されます。WITH NO LOCK指定付きまたはWITH[NO]LOCK指定なしのREAD文を実行した場合は、レコードのロックは設定されません。

WITH LOCK指定付きのREAD文を実行すると、ロックされたレコードの集合に対して、別のファイル結合子によるロックを設定するREAD文、WRITE文、REWRITE文またはDELETE文の実行ができなくなります。ただし、ロックされていないレコードに対する入出力文は実行可能です。

設定されたレコードのロックは、READ文を実行したときと同じファイル結合子に対して、以下のいずれかの文を実行したときに解除されます。

- UNLOCK文を実行したとき
- CLOSE文を実行したとき（CLOSE文が暗黙に実行されたときも、ロックが解除されます。）

なお、複数のレコードのロックを個々に解除することはできません。

2.2.8 入出力状態

入出力状態は、入出力文の結果を示す2文字の概念上の領域です。CLOSE文、DELETE文、OPEN文、READ文、REWRITE文、START文またはWRITE文の実行中に設定されます。

順ファイル機能のFORMAT句付きの印刷ファイルの場合、さらに、入出力状態の詳細情報があります。入出力状態の詳細情報は、入出力文の結果の詳細を示す4文字の概念上の領域です。

入出力状態および詳細情報の値は、入出力文に書いた無条件文の実行前、またはファイルに関連するUSE AFTER STANDARD EXCEPTION手続きの実行前に、入出力を実行するごとに設定されます。

入出力状態および詳細情報の値は、ファイル管理記述項にFILE STATUS句を書くことによって、参照することができます。

入出力状態の1桁目は、入出力状態の分類を表します。入出力状態の分類を、下表に示します。

入出力状態	分類	意味
0x	成功	入出力文の実行が成功しました。
1x	ファイル終了条件	順呼出しのREAD文の実行が、ファイル終了条件によって不成功になりました。
2x (*1)	無効キー条件	入出力文の実行が、無効キー条件によって不成功になりました。
3x	永続誤り条件	入出力文の実行が、ファイルの処理の続行を妨げる誤りによって不成功になりました。
4x	論理誤り条件	入出力文の実行が、実行の順序誤りまたは利用者が定めた限界を超えたことによって、不成功になりました。
9x	その他の誤り	入出力文の実行が、上記以外の誤りによって不成功になりました。

*1 ： 無効キー条件は、順ファイルでは発生しません。

ファイルに関連するUSE AFTER STANDARD EXCEPTION手続きが実行されるかどうかは、入出力状態によって決定されます。入出力状態の分類の成功に含まれるもの以外の条件が発生したとき、そのファイル名に関連したUSE AFTER STANDARD EXCEPTION手続きがあれば、それが実行されます。

ファイル終了条件

ファイル終了条件は、順呼出しのREAD文を実行したときに発生する可能性があります。ファイル終了条件が発生すると、READ文の実行は不成功になりますが、ファイルは影響を受けません。

ファイル終了条件が発生したかどうかは、READ文にAT END指定を書くことによって検出することができます。

無効キー条件

無効キー条件は、相対ファイルまたは索引ファイルに対して、DELETE文、乱呼出しのREAD文、REWRITE文、START文またはWRITE文を実行したときに発生する可能性があります。無効キー条件が発生すると、入出力文の実行は不成功になりますが、ファイルは影響を受けません。

無効キー条件が発生したかどうかは、入出力文にINVALID KEY指定を書くことによって検出することができます。

ファイル属性不整合条件

ファイル属性不整合条件は、OPEN文を実行したときに発生する可能性があります。ファイル属性不整合条件が発生すると、OPEN文の実行は不成功になりますが、ファイルは影響を受けません。

2.2.9 レコード形式

ファイルのレコード形式には、固定長と可変長の2つの形式があります。レコード形式は、物理的な媒体上のレコードの表現形式やレコードに付加される情報に関係なく、プログラムの記述から決まります。

固定長レコード形式の場合、ファイル中のレコードの文字位置の個数はすべて同じです。可変長レコード形式の場合、ファイル中のレコードの文字位置の個数は、レコードごとに異なります。

レコード形式は、RECORD句およびレコード記述項の記述から決まります。また、WRITE文によって書き出すレコードの長さ、およびREAD文によって読み込むレコードの長さも、RECORD句およびレコード記述項の記述から決まります。

下表に、レコード形式、最大レコード長、最小レコード長を示します。

RECORD句	レコード形式	最小レコード長	最大レコード長
書き方1 RECORD 整数-1	固定長レコード形式	整数-1の値	整数-1の値
書き方2 RECORD VARYING [FORM 整数-2 [TO 整数-3]] [DEPENDING ON データ名-1]	可変長レコード形式	レコード記述項の長さの最小値	レコード記述項の長さの最大値と整数-3のうちの大きい方
書き方3 RECORD 整数-4 TO 整数-5 または RECORD句なし	以下の場合、可変長レコード形式。それ以外の場合、固定長レコード形式 ・2つ以上のレコード記述項を書き、レコード記述項の長さが異なる場合 ・レコード記述項にOCCURS DEPENDING ON句を書いた場合 ・レコード記述項にCHARACTER TYPE句を書いた場合(*2)	レコード記述項の長さの最小値(*1)	レコード記述項の長さの最大値(*1)

*1 ： 固定長レコード形式の場合、レコード記述項の長さの最小値と最大値は同じです。
*2 ： [HP][DS][Solaris][Linux][LinuxIPF][Linux64]固有です。

2.2.10 レコード領域

ファイル節のレコード記述項に対して割り当てられる記憶領域を、「レコード領域」といいます。1つのファイルに複数のレコード記述項を関係付けることができますが、レコード領域はレコード記述項ごとに割り当てられるのではなく、ファイルごとに割り当てられます。レコード領域の大きさは、ファイルに関係付けたレコードの最大レコード長と同じです。

レコード領域は、翻訳単位中の複数のファイルで共用することができます。レコード領域を共用するためには、環境部の入出力管理段落のSAME RECORD AREA句に、レコード領域を共用するファイルを指定します。

2.2.11 特殊レジスタ

順ファイルのファイル記述項にLINAGE句を書いた場合、特殊レジスタLINAGE-COUNTERが生成されます。LINAGE-COUNTERは、LINAGE句の整数-1またはデータ名-1の大きさと同じ大きさの符号なし整数項目として扱われます。

LINAGE-COUNTERは、入出力管理システムによって維持され、印字する行数を数えるために使われます。LINAGE-COUNTERの値は、手続き部の文でだけ参照することができます。利用者がLINAGE-COUNTERに値を設定することはできません。

2.3 プログラム間連絡機能

プログラム間連絡機能は、2つ以上のプログラムが互いに連絡する機能です。CALL文を実行することによって、別の翻訳単位のプログラム、または同じ翻訳単位の中の別のプログラムを呼び出すことができます。CALL文を実行するプログラムを、「呼ぶプログラム」といいます。CALL文の実行によって制御が渡されるプログラムを、「呼ばれるプログラム」といいます。

プログラム間連絡機能には、呼ぶプログラムと呼ばれるプログラムの間で制御を受け渡す機能だけでなく、パラメタを受け渡したり、データおよびファイルを共用したりする機能もあります。

プログラム間連絡機能を使うプログラムでは、必要に応じて、下図に示す記述をします。

プログラムの記述	主な機能
見出し部	プログラムの名前を指定し、プログラムに初期化属性および共通属性を与える。
データ部	
ファイル節	
ファイル記述項	ファイル結合子に外部属性を与える。ファイル名およびデータ名に大域属性を与える。
レコード記述項	データレコードに外部属性を与える。データ名に大域属性を与える。
連絡節	
レコード記述項・独立データ記述項	受け取るパラメタを定義する。
作業場所節	
レコード記述項	データレコードに外部属性を与える。データ名に大域属性を与える。
基底場所節・定数節	
レコード記述項	データ名に大域名を与える。
手続き部	
手続き部の見出し	受け取るパラメタを指定する。
手続き部分	
CALL文	別のプログラムを呼び出す。
CANCEL文	別のプログラムを初期状態にする。
ENTRY文	二次入口名を指定する。受け取るパラメタを指定する。
EXIT PROGRAM文	呼び出したプログラムに戻る。
プログラム終わり見出し	プログラムの終わりを指定する。

2.3.1 プログラムの呼出しと復帰

プログラムを呼び出すためには、CALL文を実行します。呼ばれるプログラムから呼ぶプログラムへ復帰するためには、EXIT PROGRAM文を実行します。

実行単位

実行時に相互に連絡して動作する実行用プログラムの集まりを、「実行単位」といいます。1つの翻訳単位またはいくつかの翻訳単位が集まって1つの実行単位になります。

主プログラムと副プログラム

実行単位で最初に起動されるCOBOLプログラムを、「主プログラム」といいます。別のCOBOLプログラムから呼ばれるCOBOLプログラムを、「副プログラム」といいます。内部プログラムは、翻訳単位の一番外側のプログラムが主プログラムか副プログラムかに関係なく、つねに副プログラムです。

呼ばれるプログラムの実行の開始

CALL文を実行すると、呼ばれるプログラムが実行可能な状態になります。そして、CALL文の記述に従って、呼ばれるプログラムの以下の文から実行が始まります。

- CALL文にプログラム名またはプログラム名定数を指定した場合、手続き部の手続き部分の最初の文から実行が始まります。
- CALL文に二次入口名を指定した場合、ENTRY文に続く最初の文から実行が始まります。「二次入口名」とは、ENTRY文に書いた定数のことです。ENTRY文は、翻訳単位の一番外側のプログラムにだけ書くことができます。

呼ばれるプログラムの実行の終了

呼ばれるプログラムの実行を終了するためには、EXIT PROGRAM文またはSTOP RUN文を実行します。これらの文を総称して、「プログラム終了文」といいます。

プログラムの実行を終了して、呼び出したプログラムに戻るためには、EXIT PROGRAM文を実行します。プログラムの実行を終了して、オペレーティングシステムに戻るためには、STOP RUN文を実行します。

2.3.2 大域名と局所名

ファイル名(整列併合用ファイルのファイル名を除く)、レコード名、データ名(画面項目のデータ名を除く)および条件名には、大域属性と局所属性のどちらかの属性を与えることができます。大域属性を持つ名前を「大域名」といい、局所属性を持つ名前を、「局所名」といいます。大域名は、その名前を定義したプログラム、およびそのプログラムが直接または間接に含むプログラムで参照することができます。局所名は、その名前を定義したプログラムでだけ参照することができます。

ファイル記述項にGLOBAL句を書くと、以下の名前が大域名になります。

- ファイル記述項に書いたファイル名
- ファイル記述項に関係付けたレコード記述項の中のすべてのデータ名、レコード名および条件名

レベル番号01のデータ記述項にGLOBAL句を書くと、以下の名前が大域名になります。

- レベル番号01のデータ記述項に書いたデータ名(レコード名)
- レベル番号01のデータ記述項に従属するすべてのデータ名および条件名

ファイル名、レコード名、データ名および条件名を大域名として定義しなかった場合、その名前は局所名になります。

2.3.3 外部属性と内部属性

データ項目およびファイル結合子(整列併合用ファイルのファイル結合子を除く)には、外部属性と内部属性のどちらかの属性を与えることができます。データ項目またはファイル結合子に関連する領域を、実行単位中の複数のプログラムで共用する場合、外部属性を与えます。

ファイル記述項にEXTERNAL句を書くと、そのファイル記述項に書いたファイル名に対応するファイル結合子に、外部属性が与えられます。外部属性を持つファイル結合子を、「外部ファイル結合子」といいます。

レベル番号01のデータ記述項にEXTERNAL句を書くと、そのデータ記述項で定義したデータ項目、およびそのデータ記述項に従属するすべてのデータ項目に、外部属性が与えられます。外部属性を持つデータ項目で構成するレコードを、「外部データレコード」といいます。

同じファイル名を持つ外部ファイル結合子には、実行単位中で1つの記憶領域が割り当てられます。また、同じレコード名を持つ外部データレコードには、実行単位中で1つの記憶領域が割り当てられます。外部ファイル結合子および外部データレコードは、実行単位中のすべてのプログラムから参照することができます。なお、外部ファイル結合子および外部データレコードを内部プログラムで参照する場合、参照する名前を大域名として定義する必要があります。

データ項目およびファイル結合子に外部属性を与えなかった場合、それらは内部属性を持ちます。内部属性を持つデータ項目およびファイル結合子は、1つの翻訳単位の中でだけ参照することができます。外部属性を持たないファイル結合子を、「内部ファイル結合子」といいます。

以下の対象には、外部属性を与えることができません。これらの対象は、つねに内部属性を持ちます。

- 連絡節で定義したデータ項目
- 報告書節で定義したデータ項目
- 基底場所節で定義したデータ項目
- 定数節で定義したデータ項目
- 整列併合用ファイルのレコード記述項で定義したデータ項目
- 画面項目

このコンパイラでは、INDEXED BY指定を書いたデータ記述項を含むレコード記述項にEXTERNAL句を書いても、指標名には外部属性は与えられません。

2.3.4 外部名と内部名

外部プログラムは、外部名と内部名を持ち、内部プログラムは内部名だけを持ちます。

外部名は、外部プログラムに付ける名前で、他のプログラムを参照する場合および他のプログラムから参照される場合に使用されます。定数によって定義、参照できることから、他言語やシステムとの連携時に、COBOLの語の範囲外の名前を使用するための手段となります。

これに対して内部名は、利用者語の規則に従って定義する名前で、翻訳単位内の任意のプログラム名を表現するために利用されます。

外部プログラムに対して明に外部名が定義されてない場合、内部名と同名の外部名が暗に定義されたとみなされます。なお、本書中で明に外部名である旨が説明されてない場合は内部名を指します。

なお、外部名は、[Win32][Solaris][Linux][LinuxIPF][Linux64][.NET][WinIPF][Winx64]固有の機能です。

2.3.5 活性状態

プログラムは、OSからの活性化が成功する、または実行要素からの呼出しが成功すると、活性状態になります。プログラムは再帰的に活性化されてもよい。プログラムのインスタンスは、次のいずれかが実行されるまで、活性状態が持続します。

- STOP文
- 呼び出された同じプログラム内での、暗黙的または明示的なEXIT PROGRAM文

プログラムが活性化される場合は、常に、その内部に含まれるPERFORM文の制御機構が初期状態に設定されます。

2.3.6 プログラムの共通属性

内部プログラムには共通属性を与えることができます。共通属性を持つ内部プログラムは、別のプログラムに直接含まれている場合だけでなく、その別のプログラムに直接または間接に含まれるプログラムから呼び出すことができます。共通属性を持たない内部プログラムは、そのプログラムを直接または間接に含むプログラムだけから呼び出すことができます。

内部プログラムに共通属性を与えるためには、見出し部のプログラム名段落にCOMMON句を書きます。共通属性を持つ内部プログラムを、「共通プログラム」といいます。

2.3.7 プログラムの初期状態

プログラムが実行単位中で最初に呼び出されたとき、初期化処理が行われ、以下の状態になります。このときのプログラムの状態を、「プログラムの初期状態」といいます。

・ 作業場所節で定義したデータ項目が、初期化された状態になります。

VALUE句を指定したデータ項目およびそのデータ項目に従属するデータ項目は、VALUE句の値で初期化されます。VALUE句が適用されないデータ項目の初期値は規定されません。

・ 内部ファイル結合子が、開いた状態でない状態になります。

・ ALTER文で参照するGO TO文が、初期状態になります。

実行単位中で2回目以降に呼び出されたときにプログラムを初期状態にするためには、以下のいずれかの方法を使います。

・ プログラムに初期化属性を与える方法

・ CANCEL文を実行する方法

プログラムの初期化属性

プログラムには初期化属性を与えることができます。初期化属性を持つプログラムは、呼び出されたときにつねに初期状態になります。

プログラムに初期化属性を与えるためには、見出し部のプログラム名段落にINITIAL句を書きます。INITIAL句を書くと、INITIAL句を書いたプログラム、およびそれに直接または間接に含まれるすべてのプログラムに、初期化属性が与えられます。内部プログラムの場合、初期化属性と共通属性の両方を与えることができます。

初期化属性を持つプログラムを、「初期化プログラム」といいます。初期化プログラムは、

・ そのプログラム

・ またはそのプログラムの二次入口点

をCALL文で呼び出したときに、つねに初期状態になります。

CANCEL文による初期化

CANCEL文によって、次に呼び出されたときのプログラムの状態を初期状態にすることができます。CANCEL文を実行すると、CANCEL文に指定したプログラム、およびそのプログラムに直接または間接に含まれるプログラムが、次に呼び出されたときに初期状態になります。

2.3.8 パラメタの受渡し

呼ぶプログラムと呼ばれるプログラムの間でパラメタを受け渡すためには、以下のようにパラメタを指定します。

・ 呼ぶプログラムのCALL文のUSING指定で、受け渡すパラメタを指定します。

また、呼ばれるプログラムから返却値として値を受け取る場合は、RETURNING指定にデータ項目を指定します。C言語を呼び出す場合、関数値はRETURNING指定で受け取ります。

・ 呼ばれるプログラムの手続き部の

　― 見出し

　― またはENTRY文

のUSING指定で、受け取るパラメタを指定します。

また、返却値として値を返す場合はRETURNING指定を用います。

以下は、連絡節で定義します。

　― USING指定のデータ項目

　― RETURNING指定のデータ項目

パラメタは、USING指定にパラメタを書いた順に、呼ぶプログラムと呼ばれるプログラムの間で対応付けられます。対応するパラメタの名前は、同じである必要はありません。

CALL文のUSING指定では、呼び出されたプログラムで変更したパラメタの値を返却するかどうかを、以下のように指定します。

- 呼び出されたプログラムで変更したパラメタの値を、呼び出したプログラムに返却する場合、USING BY REFERENCE指定またはBY指定なしのUSING指定に、パラメタを指定します。
- 呼び出されたプログラムで変更したパラメタの値を、呼び出したプログラムに返却しない場合、

  ― USING BY CONTENT指定

  ― またはUSING BY VALUE指定

  に、パラメタを指定します。

  USING BY VALUE指定は、C言語など、値によるパラメタの受渡しができる言語で書いたプログラムを呼び出す場合に使います。

2.3.9 名前の範囲

利用者語には、利用者語に付ける名前の範囲が定められています。

定義したプログラムでだけ参照可能な利用者語

以下の利用者語は、定義したプログラムでだけ参照することができます。翻訳単位中の2つ以上のプログラムで同じ名前を定義することができますが、定義したプログラムでだけ参照することができます。

- 段落名
- 節名
- 報告書名
- 整列併合用ファイルのファイル名
- 画面項目のデータ名

翻訳単位中のすべてのプログラムで参照可能な利用者語

以下の利用者語は、翻訳単位中のすべてのプログラムで参照することができます。ただし、翻訳時に、参照する実体を翻訳単位に関係付ける必要があります。

- 登録集名
- 原文名

定義したプログラムおよびそれに含まれるプログラムで参照可能な利用者語

以下の利用者語は、定義したプログラムおよびそのプログラムに直接または間接に含まれるプログラムで参照することができます。

- 符号系名
- 字類名
- 条件名(環境部で定義した条件名)
- 呼び名
- 記号文字
- 記号定数
- 位置決め単位名
- 印字モード名

大域属性の有無によって名前の範囲が異なる利用者語

以下の利用者語は、大域属性の有無によって名前の範囲が異なります。

- 条件名(データ部で定義した条件名)

- データ名(画面項目のデータ名を除く)
- レコード名
- ファイル名(整列併合用ファイルのファイル名を除く)

これらの利用者語に大域属性を与えた場合、その利用者語は、定義したプログラムおよびそのプログラムに直接または間接に含まれるプログラムで参照することができます。大域属性を与えなかった場合は、定義したプログラムでだけ参照することができます。

大域属性を持つ利用者語を定義したプログラムに直接または間接に含まれるプログラムでは、大域属性を持つ利用者語と同じ名前の利用者語を定義して、複数の対象を同じ名前で参照することができます。

あるプログラム(プログラムA)が別のプログラム(プログラムB)を直接含み、ある利用者語(利用者語X)を2つ以上のプログラムで定義し、利用者語Xをプログラムで参照する場合、以下の規則を順に適用することによって、参照する対象が識別されます。

- 利用者語Xをプログラムで定義した場合、プログラムの利用者語Xの領域が参照されます。
- 利用者語Xをプログラムで定義しなかった場合、利用者語Xを定義したプログラムが見つかるまで、プログラムAから外側に向かって順に検索されます。そして、利用者語Xを定義したプログラムが初めて見つかったとき、そのプログラムの利用者語Xの領域が参照されます。

指標名の名前の範囲

表に指標名を関係付けた場合、表に付けたデータ名が大域属性を持つときは指標名も大域属性を持ちます。指標名の名前の範囲は、対応する表に付けたデータ名の名前の範囲と同じです。前述の“大域属性の有無によって名前の範囲が異なる利用者語”を参照してください。

2.3.10 プログラム名および二次入口名の名前の範囲

二次入り口名および外部名は、富士通の拡張仕様です。

1つの翻訳単位を構成するプログラムのプログラム名および二次入口名は、互いに同じであってはいけません。

2つ以上の翻訳単位で1つの実行単位を構成する場合、各翻訳単位の一番外側のプログラムのプログラム名および二次入口名は、互いに同じであってはいけません。しかし、内部プログラムには、別の翻訳単位を構成するプログラムのプログラム名または二次入口名と同じ名前を付けることができます。

プログラム名は、そのプログラムの見出し部のプログラム名段落で宣言されます。プログラム名には外部名と内部名があり、CALL文およびCANCEL文で、同一翻訳単位内のプログラムを指す場合には内部名を、同一翻訳単位外のプログラムを指す場合には外部名を参照し、プログラム終わり見出しは内部名を参照します。

実行単位を構成するプログラムに割り当てられる外部名は、一意でなければなりません。内部名は一意である必要はありませんが、実行単位中の2つのプログラムが同一の名前の場合、これらの2つのプログラムの少なくとも1つは、他方を含まない別の翻訳単位中に直接または間接に含まれていなければなりません。

プログラム名および二次入口名は、別のプログラム中のCALL文またはCANCEL文で参照することができます。CALL文およびCANCEL文で参照することができるプログラム名および二次入口名を、以下に示します。

1. 共通属性を持たない内部プログラムのプログラム名は、内部プログラムを直接または間接に含むプログラムだけから参照することができます。
2. 共通属性を持つ内部プログラムのプログラム名は、以下のプログラムから参照することができます。

   ― その内部プログラムを直接または間接に含むプログラム

   ― その内部プログラムを直接含むプログラムに、直接または間接に含まれるプログラム

   ただし、共通属性を持つ内部プログラムは、そのプログラム自身およびそれに含まれるプログラムから参照することはできません。
3. 別翻訳単位のプログラムの外部名および二次入口名は、実行単位中のすべてのプログラムから参照することができます。ただし、そのプログラムが直接または間接に含むプログラムから参照することはできません。

2.3.11 特殊レジスタ

手続き部見出しにRETURNING指定のないプログラムに関連して、特殊レジスタPROGRAM-STATUS (RETURN-CODE)が自動的に生成されます。PROGRAM-STATUSとRETURN-CODEは、同義です。PROGRAM-STATUSは、オペレーティングシステムまたは呼び出したプログラムに復帰コードを渡すために使うことができます。PROGRAM-STATUSは、“PICTURE S9(9) COMPUTATIONAL-5”と定義した数字項目として扱われます。

ただし、[WinIPF][Winx64][LinuxIPF][Linux64]では、PROGRAM-STATUSは“PICTURE S9(18) COMPUTATIONAL-5”と定義した数字項目として扱われます。

プログラムの実行が始まったときのPROGRAM-STATUSの値は、ゼロです。PROGRAM-STATUSに復帰コードを設定した後、プログラム終了文を実行すると、復帰コードが以下のように渡されます。

- オペレーティングシステムに戻った場合、復帰コードがオペレーティングシステムに通知されます。
- 呼び出したプログラムに戻った場合、呼び出したプログラムのPROGRAM-STATUSに復帰コードが格納されます。

2.4 整列併合機能

整列併合機能は、整列機能と併合機能で構成されます。

整列機能は、いくつかのファイルの中のレコードの順序を、特定のキーに従って並べ替える機能です。SORT文によって、レコードの順序を並べ替えることができます。

併合機能は、特定のキーに従って並んでいるレコードを持つ2つ以上のファイルを、併合する機能です。MERGE文によって、2つ以上のファイルを併合することができます。

整列併合機能を使うためには、整列併合用ファイルを定義する必要があります。整列併合用ファイルは、内部的に生成されるファイルです。整列併合用ファイルを、利用者が外部媒体に関係付ける必要はありません。

整列併合機能を使うプログラムでは、必要に応じて、下図に示す記述をします。

プログラムの記述	主な機能
環境部	
入出力節	
ファイル管理段落	
ファイル管理記述項	整列併合用ファイルを外部媒体に関係付ける。
入出力管理段落	整列併合用ファイルを含む複数のファイルで共用する記憶領域を指定する。
データ部	
ファイル節	
整列併合用ファイル記述項	整列併合用ファイルの物理的な構造を定義する。
レコード記述項	整列併合用ファイルのレコードの構造を定義する。
手続き部	
手続き部分	
SORT文	
MERGE文	2つ以上のファイルを併合する。
入力手続き	
RELEASE文	整合操作の最初の段階にレコードを引き渡す。
出力手続き	
RETURN文	整列操作の最後の段階からレコードを引き取る。併合操作中に併合されたレコードを引き取る。

2.4.1 整列の方法

整列する方法には、以下の4つの方法があります。

1. ファイル中のレコードを直接整列して、その結果を別のファイルに直接書き出す方法。この方法を使う場合、SORT文にUSING指定およびGIVING指定を書きます。
2. 整列前のレコードに対して特別な処理を行った後レコードを整列して、その結果を別のファイルに直接書き出す方法。この方法を使う場合、SORT文にINPUT PROCEDURE指定およびGIVING指定を書きます。
3. ファイル中のレコードを直接整列した後、整列後のレコードに対して特別な処理を行う方法。この方法を使う場合、SORT文にUSING指定およびOUTPUT PROCEDURE指定を書きます。
4. 整列前のレコードに対して特別な処理を行った後レコードを整列し、整列後のレコードに対して特別な処理を行う方法。この方法を使う場合、SORT文にINPUT PROCEDURE指定およびOUTPUT PROCEDURE指定を書きます。

2.4.2 併合の方法

併合する前のファイル中のレコードは、併合のときに使うキーに従って並べておく必要があります。併合する方法には、以下の2つの方法があります。

1. ファイル中のレコードを直接併合して、その結果を別のファイルに直接書き出す方法。この方法を使う場合、MERGE文にUSING指定およびGIVING指定を書きます。
2. ファイル中のレコードを直接併合した後、併合後のレコードに対して特別な処理を行う方法。この方法を使う場合、MERGE文にUSING指定およびOUTPUT PROCEDURE指定を書きます。

2.4.3 入力手続き

INPUT PROCEDURE指定のSORT文を書く場合、入力手続きを書く必要があります。入力手続きは、手続き部の手続き部分に書きます。入力手続きでは、以下の処理を繰り返し実行します。

1. 整列前のレコードに対して特別な処理を行います。例えば、レコードを編集したり選択したりします。
2. 次に、そのレコードに対するRELEASE文を実行します。RELEASE文を実行すると、レコードが整列併合用ファイルに書き出されます。

入力手続きの実行が終了した後、一連のRELEASE文によって整列併合用ファイルに書き出されたレコードが、整列されます。

2.4.4 出力手続き

OUTPUT PROCEDURE指定のSORT文、またはOUTPUT PROCEDURE指定のMERGE文を書く場合、出力手続きを書く必要があります。出力手続きは、手続き部の手続き部分に書きます。出力手続きは、整列併合用ファイルの整列または併合が終わった後に実行されます。出力手続きでは、以下の処理を繰り返し実行します。

1. RETURN文を実行します。RETURN文を実行すると、整列併合用ファイルからレコードが読み込まれます。
2. 次に、整列併合用ファイルから読み込んだレコードに対して特別な処理を行います。例えば、レコードを編集したり選択したりします。

2.4.5 整列併合用ファイル

整列併合用ファイルは、整列操作または併合操作を行うために、内部的に生成され割り当てられるファイルです。

整列併合用ファイルは、ファイル機能で使うファイルと異なり、ラベル手続き、ブロック化、バッファ領域、リールなどの概念を持ちません。

整列併合用ファイルは、MERGE文、RELEASE文、RETURN文およびSORT文でだけ参照することができます。

2.4.6 特殊レジスタ

整列併合機能に関連して、以下の2つの特殊レジスタが自動的に生成されます。

- SORT-STATUS
- SORT-CORE-SIZE

2.4.6.1 SORT-STATUS

SORT-STATUSは、“PICTURE S9(4) COMPUTATIONAL-5”と定義した数字項目として扱われます。整列操作または併合操作の復帰コードを参照したり、整列操作または併合操作を中断させたりするために、SORT-STATUSを使うことができます。

SORT-STATUSの初期値はゼロです。SORT文およびMERGE文の実行開始時に、自動的に初期化されます。

復帰コードの値

SORT文またはMERGE文の実行が終了すると、SORT-STATUSに復帰コードが設定されます。復帰コードの値は、以下のいずれかです。

- 0 : 整列操作または併合操作が正常に終了したことを示します。
- 16: 整列操作または併合操作が正常に終了しなかったことを示します。

プログラム中にSORT-STATUSを1つも書かなかった場合、復帰コードがゼロ以外のときは、システム論理操作卓にエラーメッセージが出力され、プログラムは異常終了します。

整列操作または併合操作を中断させる方法

INPUT PROCEDURE指定のSORT文、OUTPUT PROCEDURE指定のSORT文、またはOUTPUT PROCEDURE指定のMERGE文を書いた場合、SORT-STATUSに値を設定することによって、整列操作または併合操作を中断させることができます。

入力手続きの中で、SORT-STATUSに16を設定した後、RELEASE文を実行すると、SORT文の終わりに制御が移ります。

出力手続きの中で、SORT-STATUSに16を設定した後、RETURN文を実行すると、SORT文またはMERGE文の終わりに制御が移ります。

2.4.6.2 SORT-CORE-SIZE

特殊レジスタSORT-CORE-SIZEは、"PICTURE S9(8) COMP-5"と定義した数字項目として扱われます。整列および併合操作においてPowerSORTが使用できる記憶領域の大きさをキロバイト単位で指定します。

例えば、MOVE 32 TO SORT-CORE-SIZE を実行した場合、PowerSORTが使用できる記憶領域の大きさに32キロバイト(=32768バイト)を指定したことになります。

指定された値が実際に有効になるかについては、PowerSORTの"オンラインマニュアル"を参照してください。

SORT-CORE-SIZEの値は、実行時オプションsmsizeおよび翻訳オプションSMSIZEに指定する値の意味と等価ですが、同時に指定された場合の優先順位は、特殊レジスタSORT-CORE-SIZEが一番高く、以降、実行時オプションsmsize、翻訳オプションSMSIZEの順で低くなります。

なお、SORT-CORE-SIZEは[Win32][WinIPF][Winx64][Solaris][Linux][LinuxIPF][Linux64][.NET]固有の機能です。

2.5 原始文操作機能

原始文操作機能は、プログラムの一部を取り込んだり置き換えたりして、翻訳時にプログラムを完成させるための機能です。

原始文操作機能には、COPY文とREPLACE文があります。これらの文は、翻訳時、他の文が翻訳される前に処理されます。実行時には意味を持ちません。

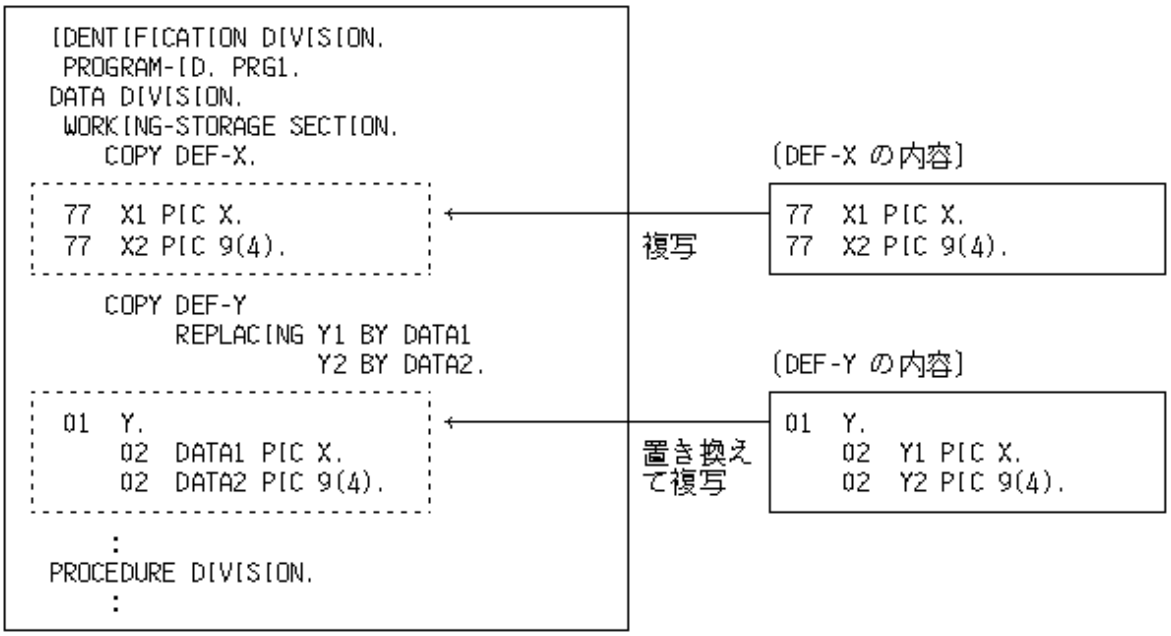
COPY文は、プログラムの一部をCOBOL登録集から複写するために使います。COBOL登録集は、プログラムとは別に作成し、翻訳時にプログラムと関係付けます。

REPLACE文は、プログラムの一部を置き換えるために使います。

COPY文およびREPLACE文は、プログラムおよびCOBOL登録集の任意の位置に書くことができます。

例

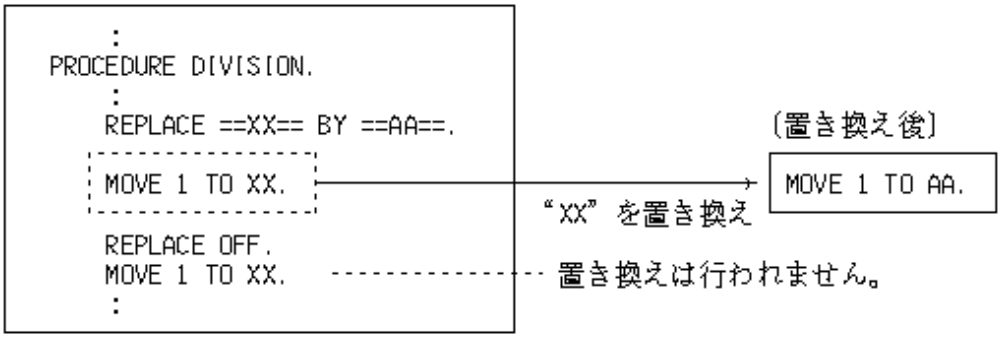
COPY文の例



例

REPLACE文の例

REPLACE文は、置き換えを開始するREPLACE文と置き換えを終了する文(REPLACE OFF文)を対にして使います。



2.6 表示ファイル機能

表示ファイル機能は、表示サービス管理システムまたは通信管理シテスムを介して、データの入出力およびメッセージの送受信を行う機能です。

表示ファイル機能には、以下の機能があります。

- 画面定義体を使って、ディスプレイ装置上のデータを操作する機能
- 帳票定義体を使って、プリンタ装置に帳票を出力する機能

・ 装置とプログラムとの間でメッセージを受け渡す機能

[LinuxIPF][.NET]では、”帳票定義体を使って、プリンタ装置に帳票を出力する機能”だけ使用できます。

画面帳票定義体は、ディスプレイ装置またはプリンタ装置で扱うデータのレイアウト情報です。画面帳票定義体は、プログラムとは別に作成します。

表示ファイル機能では、表示ファイルを使って、データの入出力およびメッセージの送受信を行います。表示ファイルは、装置とプログラムとの間のインタフェースとなるファイルです。表示ファイルに対する入出力文を実行することによって、データの入出力およびメッセージの送受信を行います。

表示ファイル機能を使うプログラムでは、必要に応じて、下図に示す記述をします。

プログラムの記述	主な機能
環境部	
入出力部	
ファイル管理段落	
ファイル管理記述項	表示ファイルのファイル名を外部媒体と関係付ける。あて先種別などの物理属性を指定する。
入出力管理段落	複数の会話にまたがるデータ項目の扱いなど、表示ファイル特有の処理技法を指定する。
データ部	
ファイル節	
ファイル記述項	表示ファイルの物理的な構造を定義する。
レコード記述項	表示ファイルのレコードの構造を定義する。
手続き部	
宣言部分	
USE文およびUSE AFTER STANDARD EXCEPTION手続き	入出力誤り手続きを定義する。
手続き部分	
OPEN文	ファイルを開く。
READ文	ファイルからレコードを読み込む。
WRITE文	ファイルにレコードを書き出す。
CLOSE文	ファイルを閉じる。

2.6.1 あて先種別

データの入出力およびメッセージの送受信は、表示ファイルに対する入出力文を実行することによって行います。どの装置を対象に入出力文を実行するかは、あて先種別で指定します。

あて先種別には、以下のものがあります。

・ ディスプレイ装置

・ プリンタ装置

・ その他の論理装置

あて先種別は、ファイル管理記述項のSYMBOLIC DESTINATION句で指定します。CLOSE文、OPEN文、READ文およびWRITE文を実行する前に、あて先種別を設定する必要があります。

2.6.2 画面帳票定義体

ディスプレイ装置に表示する画面のレイアウトおよびプリンタ装置に印刷する帳票のレイアウトは、画面帳票定義体として、プログラムとは別に定義することができます。画面帳票定義体を使う場合、ファイル管理記述項のFORMAT句で、画面帳票定義体名を設定するデータ項目を指定します。

画面帳票定義体は、プログラムの実行時、レコードを編集するために使われます。WRITE文を実行すると、WRITE文に指定したレコードが画面帳票定義体に従って編集され、装置に出力されます。READ文を実行すると、装置から入力したデータが画面帳票定義体に従って編集され、プログラムに通知されます。

装置とプログラムとの間で受け渡すデータのデータ記述項は、画面帳票定義体から複写することができます。

2.6.3 ファイルの編成と呼出し法

表示ファイルでは、ファイルの編成と呼出し法が以下のように決められています。

表示ファイルの編成は、順編成です。ファイル管理記述項のORGANIZATION句でSEQUENTIAL指定を書くか、ORGANIZATION句を省略します。

表示ファイルは、順呼出し法でだけ呼び出すことができます。ファイル管理記述項のACCESS MODE句でSEQUENTIAL指定を書くか、ACCESS MODE句を省略します。

2.6.4 入出力文の動作

表示ファイルに対する処理は、入出力文(OPEN文、READ文、WRITE文およびCLOSE文)を組み合わせて書きます。

表示ファイルに対する処理を行うためには、最初にOPEN文を実行してファイルを開かなければなりません。ファイルに対する処理を終了した後、CLOSE文を実行してファイルを閉じます。OPEN文とCLOSE文の間に実行することができる入出力文は、オープンモードに依存します。

表示ファイルのオープンモードと入出力文の関係

下表に、表示ファイルのオープンモードと入出力文の関係を示します。

入出力文	オープンモード（括弧内は、OPEN文の指定を示す）		
	入力モード（INPUT指定）	出力モード（OUTPUT指定）	入出力両用モード（I-O指定）
READ文	○	－	○
WRITE文	－	○	○

○ ： 実行可能
－ ： 実行不可能

2.6.5 入出力状態

入出力状態は、入出力文の結果を示す2文字の概念上の領域です。CLOSE文、OPEN文、READ文またはWRITE文の実行中に設定されます。

入出力状態には、さらに詳細情報があります。入出力状態の詳細情報は、入出力文の結果の詳細を示す4文字の概念上の領域です。

入出力状態および詳細情報の値は、入出力文に書いた無条件文の実行前、またはファイルに関連するUSE AFTER STANDARD EXCEPTION手続きの実行前に、入出力を実行するごとに設定されます。

入出力状態および詳細情報の値は、ファイル管理記述項にFILE STATUS句を書くことによって、参照することができます。

入出力状態の1桁目は、入出力状態の分類を表します。入出力状態の分類を、下表に示します。

入出力状態	分類	意味
0x	成功	入出力文の実行が成功しました。
1x	ファイル終了条件	READ文の実行が、ファイル終了条件によって不成功になりました。

入出力状態	分類	意味
3x	永続誤り条件	入出力文の実行が、ファイルの処理の続行を妨げる誤りによって不成功になりました。
4x	論理誤り条件	入出力文の実行が、実行の順序誤りによって不成功になりました。
9x	その他の誤り	入出力文の実行が、上記以外の誤りによって不成功になりました。

ファイルに関連するUSE AFTER STANDARD EXCEPTION手続きが実行されるかどうかは、入出力状態によって決定されます。入出力状態の分類の成功に含まれるもの以外の条件が発生したとき、そのファイル名に関連したUSE AFTER STANDARD EXCEPTION手続きがあれば、それが実行されます。

ファイル終了条件

ファイル終了条件は、READ文を実行したときに発生する可能性があります。ファイル終了条件が発生すると、READ文の実行は不成功になりますが、ファイルは影響を受けません。

ファイル終了条件が発生したかどうかは、READ文にAT END指定を書くことによって検出することができます。

2.6.6 特殊レジスタ

画面帳票定義体を作成するときに、画面または帳票のレイアウトを定義すると同時に、プログラムと装置の両方で使うデータ項目を定義します。このときに、データ項目に項目制御部を付加するかどうかを指定することができます。項目制御部は、データ項目を入出力するときの詳細な情報を、装置とプログラムとの間で受け渡すための領域です。プログラムでは、項目制御部の内容を特殊レジスタを使って参照したり設定したりすることができます。

表示ファイルに関係付けたレコード記述項の中で、項目制御部を持つデータ項目を書いた場合、それぞれのデータ項目に対して以下の7種類の特殊レジスタが生成されます。

- EDIT-MODE
- EDIT-OPTION
- EDIT-OPTION2
- EDIT-OPTION3
- EDIT-COLOR
- EDIT-STATUS
- EDIT-CURSOR

これらの7種類の特殊レジスタは、1文字の英数字項目として扱われます。

これらの特殊レジスタは、手続き部の文でだけ参照することができます。特殊レジスタは、項目制御部を持つデータ項目に関係付けられるので、特殊レジスタを参照する場合、項目制御部を持つデータ項目で修飾しなければなりません。特殊レジスタの修飾の規則については、“1.4.1 修飾”を参照してください。

特殊レジスタの機能は、以下のとおりです。特殊レジスタの値および項目制御部については、“NetCOBOL使用手引書”を参照してください。

1. 特殊レジスタEDIT-MODEは、WRITE文の実行時に、データ項目の処理モードをシステムに通知するために使います。WRITE文に指定したレコード中のデータ項目について、そのデータ項目を処理の対象とするかどうか、および日本語項目を日本語として表示するか英数字として表示するかという情報を指定することができます。
2. 特殊レジスタEDIT-OPTIONは、WRITE文の実行時に、データ項目の処理オプションをシステムに通知するために使います。WRITE文に指定したレコードの中のデータ項目について、明滅表示、強調表示、反転表示および下線表示の有無を指定することができます。
3. 特殊レジスタEDIT-OPTION2は、WRITE文によってレコードを表示ファイルに書き出すときに、一意名－1のデータ項目に対する処理オプション2（背景色）を指定します。
   なお、EDIT-OPTION2は[Win32][WinIPF][Winx64][Solaris][Linux][LinuxIPF][Linux64][.NET]固有の機能です。

4. 特殊レジスタEDIT-OPTION3は、WRITE文によってレコードを表示ファイルに書き出すときに、一意名－1のデータ項目に対する処理オプション3（網がけ）を指定します。
   なお、EDIT-OPTION3は[Win32][WinIPF][Winx64][Solaris][Linux][LinuxIPF][Linux64][.NET]固有の機能です。
5. 特殊レジスタEDIT-COLORは、WRITE文の実行時に、データ項目の表示色、輝度などをシステムに通知するために使います。
6. 特殊レジスタEDIT-STATUSは、READ文の実行時に、データ項目の入力モードをシステムに通知するために使います。READ文に指定したレコード中のデータ項目を、入力処理の対象とするかどうかを指定することができます。また、READ文の実行が成功すると、データ項目の入出力状態(正常入力か異常入力かの区別など)がEDIT-STATUSに設定されます。
7. 特殊レジスタEDIT-CURSORは、READ文の実行時に、データ項目のカーソルの位置付けなどをシステムに通知するために使います。

2.7 組込み関数機能

組込み関数機能は、いくつかの引数の値を基に特定の処理を行い、その結果をプログラムに返却する機能です。関数の値を、「関数値」といいます。

組込み関数機能は、手続き部の文の中で使うことができます。文の中に関数一意名を書くことによって、組込み関数機能を使うことができます。

組込み関数の一覧を、下表に示します。

関数名	関数値
ACOS	引数の逆余弦
ACP-OF	引数をACP(Ansi Code Page)の文字列に変換したもの [.NET]固有
ADDR	引数の先頭アドレス
ANNUITY	毎期の均等払い額
ASIN	引数の逆正弦
ATAN	引数の逆正接
CAST-ALPHANUMERIC	日本語または日本語編集文字列を英数字に変換したもの
CHAR	文字の大小順序において、引数の順序位置にある1文字
COS	引数の余弦
CURRENT-DATE	現在の日付と時刻
DATE-OF-INTEGER	通日形式の日付を標準形式の日付に変換したもの
DAY-OF-INTEGER	通日形式の日付を年日形式の日付に変換したもの
DISPLAY-OF	引数に含まれる日本語文字を対応する英数字文字に置き換えたもの [Win32][WinIPF][Winx64][Solaris][LinuxIPF][Linux64][.NET]固有
ENUM-AND	引数の並びのブール積 [.NET]固有
ENUM-NOT	引数の値のブール否定 [.NET]固有
ENUM-OR	引数の並びのブール和 [.NET]固有
FACTORIAL	引数の階乗
INTEGER	引数の値以下の最大の整数

関数名	関数値
INTEGER-OF-DATE	標準形式の日付を通日形式の日付に変換したもの
INTEGER-OF-DAY	年日形式の日付を通日形式の日付に変換したもの
INTEGER-PART	引数の整数部
LENG	引数の大きさ(バイト数)
LENGTH	引数の長さ(文字位置または日本語文字位置の個数)
LOG	引数の自然対数
LOG10	引数の常用対数
LOWER-CASE	引数の中の英大文字を英小文字に置き換えたもの
MAX	引数の並びの中の最大値
MEAN	引数の並びの算術平均値
MEDIAN	引数の並びの中央値
MIDRANGE	引数の並びの中の最小値と最大値の算術平均値
MIN	引数の並びの中の最小値
MOD	第2引数を法とする第1引数の整数値
NATIONAL	COBOL 文字集合を日本語文字に変換したもの
NATIONAL-OF	引数に含まれる英数字文字を対応する日本語文字に置き換えたもの [Win32][WinIPF][Winx64][Solaris][LinuxIPF][Linux64][.NET]固有
NUMVAL	数字定数の形式の文字列を数値に変換したもの
NUMVAL-C	数字定数の形式の文字列(通貨記号またはコンマを含む)を数値に変換したもの
ORD	文字の大小順序における、引数の順序位置
ORD-MAX	最大値を持つ引数の位置
ORD-MIN	最小値を持つ引数の位置
PRESENT-VALUE	各期末の現在価値
RANDOM	疑似乱数
RANGE	引数の並びの中の最大値と最小値の差
REM	第1引数を第2引数で割ったときの余り
REVERSE	引数の文字列を逆順にしたもの
SIN	引数の正弦
SQRT	引数の平方根
STANDARD-DEVIATION	引数の並びの標準偏差
STORED-CHAR-LENGTH	有効文字の文字位置の個数 [Win32][WinIPF][Winx64][Solaris][Linux][LinuxIPF][Linux64][.NET]固有
SUM	引数の並びの総和
TAN	引数の正接
UCS2-OF	文字のエンコード方式をUCS2に変換したもの [Win32][WinIPF][Winx64][Solaris][Linux][LinuxIPF][Linux64][.NET]固有
UNICODE-OF	引数の値をUNICODEの文字列に変換したもの [.NET]固有
UPPER-CASE	引数の中の英小文字を英大文字に置き換えたもの

関数名	関数値
UTF8-OF	文字のエンコード方式をUTF8に変換したもの [Win32][WinIPF][Winx64][Solaris][Linux][LinuxIPF][Linux64][.NET]固有
VARIANCE	引数の並びの分散
WHEN-COMPILED	プログラムを翻訳したときの日付と時刻

2.8 スクリーン操作機能

スクリーン操作機能は、スクリーン管理システムを介して、ディスプレイ装置にデータを表示したり、ディスプレイ装置からデータを入力する機能です。画面上のどの位置にデータを表示し、画面上のどの位置からデータを入力するかは、データ部の画面節で定義します。画面節では、画面上の領域の配置だけでなく、表示色、入出力属性なども定義することができます。
[WinIPF][Linux][LinuxIPF][Linux64][.NET]では、スクリーン操作機能は使用できません。
スクリーン操作機能を使うプログラムでは、必要に応じて、下図に示す記述をします。

プログラムの記述	主な機能
環境部	
構成節	
特殊名段落	カーソル位置および画面入力状態を受け取るデータ名を指定する。
データ部	
画面節	
画面記述項	画面上の領域の属性を定義する。
手続き部	
手続き部分	
DISPLAY文	画面を表示する。
ACCEPT文	画面からデータを入力する。

2.8.1 画面と画面項目

画面は、行と列の組合せで表される、一定の大きさを持つ領域です。ある時間に表示されている画面を1つの画面とすると、1つの画面には、用途に合わせて、いくつかの領域を配置することができます。

2.8.1.1 画面データ記述項

画面上の領域は、画面項目として、画面節の画面データ記述項で定義します。画面上の領域の配置は、画面項目に行番号と列番号を指定することによって定義します。行番号は、物理画面の上から下に向かって第1行、第2行と数えます。列番号は、1文字の英数字を1列として、物理画面の左から右に向かって、第1列、第2列と数えます。行番号は、LINE NUMBER句で指定し、列番号はCOLUMN NUMBER句で指定します。
画面データ記述項では、さらに、画面上の領域について、入出力属性、表示色、輝度、カーソルの位置付け方法、表示前の画面の消去/非消去などの属性を定義します。
プログラムで設定した値を画面に表示したり、画面から入力したデータをプログラムで受け取るためには、画面項目とは別に、作業用のデータ項目を定義する必要があります。画面項目とデータ項目の対応付けは、画面データ記述項で指定します。

2.8.1.2 画面項目

画面項目には、基本画面項目と集団画面項目の2種類があります。1つの行のある列からある列までの領域は、基本画面項目として定義します。いくつかの基本画面項目をまとめた領域は、集団画面項目として定義します。いくつかの集団画面項目をまとめて、集団画面項目として定義することもできます。
基本画面項目は、1回のACCEPT文およびDISPLAY文で処理することができる最小の単位の画面項目です。ACCEPT文またはDISPLAY文に集団画面項目を指定した場合は、集団画面項目に従属する基本画面項目を一度に処理するこ

とができます。
基本画面項目には、以下の4種類があります。

定数項目

定数項目は、固定的な値を画面に表示するための画面上の領域です。定数項目の値は、画面データ記述項のVALUE句で指定します。DISPLAY文を実行すると、VALUE句で指定した値が表示されます。

入力項目

入力項目は、画面からデータを受け取るための画面上の領域です。画面データ記述項で、1つの入力項目に対して1つのデータ項目を対応付けます。ACCEPT文を実行すると、入力項目に対応付けたデータ項目に、画面から入力したデータが設定されます。入力項目には、プログラムで設定した値を表示することはできません。

出力項目

出力項目は、画面にデータを表示するための画面上の領域です。画面データ記述項で、1つの出力項目に対して1つのデータ項目を対応付けます。出力項目に対応付けたデータ項目に値を設定してDISPLAY文を実行すると、その値が表示されます。出力項目には、画面からデータを入力することはできません。

更新項目

更新項目は、画面からデータを受け取ったり画面にデータを表示したりするための、画面上の領域です。画面データ記述項で、1つの更新項目に対して1つのデータ項目を対応付けます。更新前のデータの値は保持されません。更新項目に対応付けたデータ項目に値を設定してDISPLAY文を実行すると、その値が表示されます。また、ACCEPT文を実行すると、更新項目に対応付けたデータ項目に、画面から入力したデータが設定されます。

2.8.2 画面の入出力操作

画面を表示する場合、DISPLAY文を実行します。画面からデータを入力する場合、ACCEPT文を実行します。
DISPLAY文およびACCEPT文には、基本画面項目および集団画面項目のどちらかを指定することができます。基本画面項目を指定した場合は、基本画面項目だけが入出力操作の対象になります。集団画面項目を指定した場合は、集団画面項目に従属する基本画面項目が入出力操作の対象になります。
ACCEPT文を実行すると、画面から入力したデータが、入力項目または更新項目に対応付けたデータ項目に格納されます。
DISPLAY文を実行すると、出力項目または更新項目に対応付けたデータ項目に設定した値および定数項目の値が、画面に表示されます。このとき、画面項目のBLANK句またはERASE句の指定に従って、現在の画面を消去した後、または現在の画面に重ね合わせて、画面項目が表示されます。

2.8.3 画面入力状態

画面入力状態は、ACCEPT文の実行結果を示す3文字の概念上の領域です。画面入力状態の値は、ACCEPT文に書いた無条件文を実行する前に、ACCEPT文を実行するごとに設定されます。
画面入力状態は、環境部の特殊名段落でCRT STATUS句を書くことによって、参照することができます。
画面入力状態の一覧は、“4.2.3.6 CRT STATUS句”を参照してください。

2.9 コマンド行引数と環境変数の操作機能

コマンド行引数と環境変数の操作機能は、以下の2つの機能で構成されます。

- コマンド行に指定した引数の個数および引数の値を参照する機能

  コマンド行とは、COBOLの実行用プログラムを呼び出すコマンドのことです。

- 環境変数の値を参照したり更新したりする機能

  コマンド行引数と環境変数の操作は、ACCEPT文とDISPLAY文を組み合わせて行います。

ACCEPT文およびDISPLAY文では、処理の種類を指定するために、機能名に対応付けた呼び名を指定します。
コマンド行引数と環境変数の操作機能を使うプログラムでは、必要に応じて、下図に示す記述をします。

プログラムの記述	主な機能
環境部	
構成節	

プログラムの記述	主な機能
特殊名段落	機能名を呼び名に対応付ける。
手続き部	
手続き部分	
DISPLAY文	コマンド行の引数を位置づける。環境変数を位置づける。
ACCEPT文	コマンド行の引数の個数と値を得る。環境変数の値を得る。

2.9.1 コマンド行引数の操作

コマンド行には、引数として、COBOLプログラムに渡すパラメタ、外部スイッチおよび実行時オプションを指定することができます。引数の個数および値は、コマンド行引数の操作機能を使って参照することができます。

2.9.1.1 引数の個数を求める方法

引数の個数を求めるためには、ACCEPT文を実行します。ACCEPT文のFROM指定には、機能名ARGUMENT-NUMBERに対応付けた呼び名を書きます。ACCEPT文を実行すると、引数の個数がACCEPT文の一意名に設定されます。

2.9.1.2 引数の値を参照する方法

引数の値を参照する方法には、以下の2つの方法があります。

- 引数を指定した順に、値を参照する方法
- 引数の位置を指定して、値を参照する方法

参照する引数の位置は、引数位置指示子を使って把握します。

引数位置指示子

引数位置指示子は、次に参照する引数を指定するために使う概念上の指示子です。
コマンド行に指定した引数には、指定した順に左から右へ、初期値を1、増分を1とした番号が与えられます。引数位置指示子の値は、引数に与えられた番号を表します。
引数位置指示子の値は、0～99までの整数です。引数位置指示子の値0は、コマンドに対応します。
引数位置指示子の値は、機能名ARGUMENT-NUMBERに対応付けた呼び名を指定したDISPLAY文、および機能名ARGUMENT-VALUEに対応付けた呼び名を指定したACCEPT文によってだけ影響を受けます。これらのDISPLAY文もACCEPT文も実行されていないときの引数位置指示子の初期値は、1です。

引数を指定した順に値を参照する方法

引数を指定した順に値を参照するためには、ACCEPT文を繰り返し実行します。ACCEPT文のFROM指定には、機能名ARGUMENT-VALUEに対応付けた呼び名を書きます。
ACCEPT文を実行するごとに、現在の引数位置指示子が示す位置にある引数の値が、ACCEPT文の一意名に設定されます。引数位置指示子の値は、ACCEPT文を実行するごとに、1ずつ増やされます。

引数の位置を指定して値を参照する方法

引数の位置を指定して値を参照するためには、DISPLAY文とACCEPT文を、この順に実行します。
DISPLAY文では、参照したい引数の位置を一意名または定数に指定し、機能名ARGUMENT-NUMBERに対応付けた呼び名をUPON指定に書きます。DISPLAY文を実行すると、DISPLAY文の一意名または定数に設定した値が、引数位置指示子に設定されます。
ACCEPT文では、機能名ARGUMENT-VALUEに対応付けた呼び名をFROM指定に書きます。ACCEPT文を実行すると、現在の引数位置指示子が示す位置にある引数の値が、ACCEPT文の一意名に設定されます。また、現在の引数位置指示子の値が、1だけ増やされます。

2.9.2 環境変数の操作

環境変数の値は、環境変数の操作機能を使って参照したり更新したりすることができます。

2.9.2.1 環境変数の値を参照する方法

環境変数の値を参照するためには、DISPLAY文とACCEPT文を、この順に実行します。
DISPLAY文では、参照したい環境変数の名前を一意名または定数に指定し、機能名ENVIRONMENT-NAMEに対応付けた呼び名をUPON指定に書きます。DISPLAY文を実行すると、DISPLAY文の一意名または定数に設定した名前を持つ環境変数が、入出力操作可能になります。
ACCEPT文では、機能名ENVIRONMENT-VALUEに対応付けた呼び名をFROM指定に書きます。ACCEPT文を実行すると、DISPLAY文の実行によって入出力操作可能になった環境変数の値が、ACCEPT文の一意名に設定されます。

2.9.2.2 環境変数の値を更新する方法

環境変数の値を更新するためには、2つのDISPLAY文を実行します。
最初のDISPLAY文では、更新したい環境変数の名前を一意名または定数に指定し、機能名ENVIRONMENT-NAMEに対応付けた呼び名をUPON指定に書きます。DISPLAY文を実行すると、DISPLAY文の一意名または定数に設定した名前を持つ環境変数が、入出力操作可能になります。
2番目のDISPLAY文では、更新する値を一意名または定数に設定し、機能名ENVIRONMENT-VALUEに対応付けた呼び名をUPON指定に書きます。DISPLAY文を実行すると、最初のDISPLAY文の実行によって入出力操作可能になった環境変数が、DISPLAY文の一意名または定数に設定した値で置き換えられます。

2.10 報告書作成機能

報告書作成機能は、報告書の形式をデータ部で定義することによって、報告書を作成するための手続きを簡略化する機能です。報告書の印字位置を決めるために必要な処理は、報告書作成管理システムによって自動的に行われます。このため、ページ番号のカウント、ページ内行数のカウント、印字行の組立て、行送りなどの処理を、手続き部に書く必要がなくなります。
報告書の出力先となるファイルを、「報告書ファイル」といいます。報告書ファイルは、順ファイル機能の印刷ファイルと同じように扱われます。報告書ファイルは、データ部のファイル記述項で定義します。
1つの報告書ファイルには、いくつかの体裁の報告書を出力することができます。報告書の体裁は、1つのページの体裁が1つの報告書の体裁に対応すると関連づけて、データ部の報告書記述項で定義します。報告書ファイルと報告書記述項との関係付けは、ファイル管理記述項のREPORT句で指定します。
報告書の中には、いくつかのデータを出力することができます。報告書に出力するデータは、報告集団記述項で定義します。報告集団記述項は、階層構造で定義します。
[WinIPF][Winx64][LinuxIPF][Linux64][.NET]では報告書作成機能は使用できません。
報告書作成機能を使うプログラムでは、必要に応じて、下図に示す記述をします。

プログラムの記述	主な機能
環境部	
入出力節	
ファイル管理段落	
ファイル管理記述項	報告書ファイルのファイル名を外部媒体と関係付ける。編成、呼出し法などを指定する。
入出力管理段落	報告書ファイルを含む複数のファイルで共用する記憶領域を指定する。
データ部	
ファイル節	
ファイル記述項	報告書ファイルの物理的な構造を定義する。報告書ファイルを報告書名と関係付ける。
報告書節	
報告書記述項	報告書の物理的な構造を定義する。
報告集団記述項	報告書中の個々の項目を定義する。
手続き部	
宣言部分	

プログラムの記述			主な機能
		USE文およびUSE AFTER STANDARD EXCEPTION手続き	入出力誤り手続きを定義する。
		USE BEFORE REPORTING文およびUSE BEFORE REPORTING手続き	報告集団を表示する前に実行する手続きを定義する。
	手続き部分		
		OPEN文	ファイルを開く。
		INITIALIZE文	報告書の処理を開始する。
		GENERATE文	報告書を作成する。
		SUPPRESS文	報告集団の表示を抑制する。
		TERMINATE文	報告書の処理を終了する。
		CLOSE文	ファイルを閉じる。

2.10.1 報告書ファイル

報告書ファイルは、順編成の出力ファイルです。報告書ファイルは、データ部のファイル記述項で定義します。ファイル記述項には必ずREPORT句を書きます。報告書記述項の報告書名をREPORT句に指定することにより、報告書ファイルと報告書記述項を関係付けます。

順ファイルを定義する場合と異なり、報告書ファイルを定義する場合は、ファイル節にレコード記述項を書くことはできません。報告書ファイルのレコードの構造は、報告書節の報告集団記述項で定義します。報告書ファイルのレコードは、報告書作成管理システムの制御によって書き出されます。

報告書ファイルは、OPEN文、GENERATE文、INITIATE文、SUPPRESS文、TERMINATE文、USE AFTER STANDARD EXCEPTION文、USE BEFORE REPORTING文およびCLOSE文で参照することができます。

2.10.2 特殊レジスタ

報告書ファイル機能に関連して、以下の2つの特殊レジスタが生成されます。

- PAGE-COUNTER
- LINE-COUNTER

これらの特殊レジスタは、報告書節で定義した各報告書に対して、1つずつ生成されます。特殊レジスタPAGE-COUNTERを「ページカウンタ」といい、特殊レジスタLINE-COUNTERを「行カウンタ」といいます。

2.10.2.1 ページカウンタ

ページカウンタは、“PIC 9(6) USAGE IS PACKED-DECIMAL”と定義した数字項目として扱われます。ページカウンタの値の範囲は、1～999999です。ページカウンタの値は、報告書作成管理システムによって維持され、報告書のページ番号を付けるためにプログラムで使うことができます。ページカウンタは、報告書節のSOURCE句および手続き部の文でだけ参照することができます。ページカウンタの値は、プログラムで変更することもできます。

2.10.2.2 行カウンタ

行カウンタは、“PIC 9(6) USAGE IS PACKED-DECIMAL”と定義した数字項目として扱われます。行カウンタの値の範囲は、1～999999です。行カウンタの値は、報告書作成管理システムによって維持され、報告書の縦の位置を決めるために使われます。行カウンタは、報告書節のSOURCE句および手続き部の文でだけ参照することができます。行カウンタの値をプログラムで変更することはできません。

第3章 見出し部とプログラム終わり見出し

見出し部では、プログラムを識別する名前とプログラムの属性を指定します。プログラムの最初の部は、見出し部でなければなりません。

プログラムの終わりは、プログラム終わり見出しで指定します。

3.1 見出し部の構成（IDENTIFICATION DIVISION）

見出し部の書き方を、以下に示します。

【書き方】

```
[IDENTIFICATION DIVISION.]
PROGRAM-ID. プログラム記述項.          } プログラム名段落
[AUTHOR. [注記項] … ].                 } AUTHOR段落
[INSTALLATION. [注記項] … ].           } INSTALLATION段落      } 見出し部
[DATE-WRITTEN. [注記項] … ].           } DATE-WRITTEN段落
[DATE-COMPILED. [注記項] … ].          } 翻訳日付段落
[SECURITY. [注記項] … ].               } SECURITY段落
```

構文規則

1. 見出し部の最初の段落は、プログラム名段落でなければなりません。プログラム名段落には、プログラムの名前を書かなければなりません。
2. プログラム名段落の後には、AUTHOR段落、INSTALLATION段落、DATE-WRITTEN段落、翻訳日付段落またはSECURITY段落を書くことができます。

   このコンパイラでは、これらの段落を書く順序は任意です。

   これらの段落は注記項を書くための段落であり、廃要素です。

3.1.1 プログラム名段落（PROGRAM-ID）

プログラムの名前を指定します。

【書き方】

```
PROGRAM-ID. { プログラム名 [AS 定数-1]          }
            { プログラム名定数                  }
[IS { | COMMON | | INITIAL | } PROGRAM ].
```

注意

AS指定は、[Win32][WinIPF][Winx64][Solaris][Linux][LinuxIPF][Linux64][.NET]固有の機能です。

構文規則

1. 以下に、プログラムの名前を指定します。
   - プログラム名
   - プログラム名定数

   内部プログラムの名前は、そのプログラムを直接または間接に含むプログラムの名前と同じであってはいけません。
2. COMMON句は、内部プログラムにだけ書くことができます。
3. プログラム名は、利用者語の規則に従って書かなければなりません。
4. プログラム名定数の記述規則については、“1.2.6 特殊な用途の定数”を参照してください。
5. 定数-1は、表意定数でない、文字定数または日本語文字定数でなければなりません。
6. 定数-1は、そのプログラムが別のプログラム中に含まれる場合、書いてはなりません。

一般規則

1. 以下に指定した文字列が、原始プログラム、実行用プログラムおよびすべての印字出力を識別するための名前になります。
   - プログラム名
   - プログラム名定数
2. 内部プログラムに共通属性を与える場合、COMMON句を書きます。共通属性を持つプログラムを、「共通プログラム」といいます。共通プログラムは、そのプログラムを含むプログラム以外のプログラムから呼び出すことができます。
3. プログラムに初期化属性を与える場合、INITIAL句を書きます。初期化属性を持つプログラムを、「初期化プログラム」といいます。初期化プログラムが呼び出されると、そのプログラムおよびそれに含まれるすべてのプログラムが初期状態になります。
4. 初期化プログラムにENTRY文を書いて二次入口名を指定した場合、そのプログラムが二次入口名を参照して呼び出されたときも、そのプログラムおよびそれに含まれるすべてのプログラムが初期状態になります。
5. プログラム名は、このプログラム定義で宣言されたプログラムに名前を付けます。定数-1が指定された場合、プログラム名が内部名となり、定数-1が外部名になります。定数-1が指定されなかった場合、プログラム名が内部名および外部名になります。

3.1.2 翻訳日付段落（DATE-COMPILED）

原始プログラムリストの見出し部に翻訳日付を挿入します。翻訳日付段落は廃要素です。

【書き方】

DATE-COMPILED. [注記項] …

注意

[.NET]では、翻訳日付段落は注釈としてみなされます。

一般規則

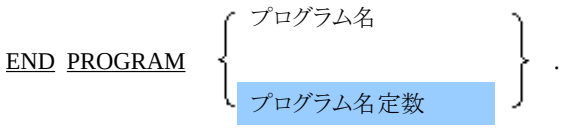
この段落の見出し(DATE-COMPILED)を書くと、翻訳時に出力される原始プログラムリストに翻訳日付が挿入されます。原始プログラムリストでは、翻訳日付段落は以下の形式に置き換えられます。

```
DATE-COMPILED.      翻訳日付.
```

3.2 プログラム終わり見出し(END PROGRAM)

プログラム終わり見出しは、プログラムの終わりを指定します。

【書き方】



構文規則

1. 以下にプログラムの名前を指定します。プログラムの名前は、先行するプログラム名段落で指定したプログラムの名前と同じでなければなりません。
   - プログラム名
   - プログラム名定数
2. 見出し部のプログラム名段落とプログラム終わり見出しは、対にして書かなければなりません。例えば、プログラムAがプログラムBを含む場合、プログラムBのプログラム終わり見出しはプログラムAのプログラム終わり見出しよりも前に書かなければなりません。
3. プログラム名は、利用者語の規則に従って書かなければなりません。
4. プログラム名定数の記述規則については、“1.2.6 特殊な用途の定数”を参照してください。

一般規則

1. プログラム終わり見出しは、プログラムの終わりを指定します。
2. 翻訳単位中に内部プログラムを書く場合、その翻訳単位中のすべてのプログラムに、プログラム終わり見出しを書かなければなりません。
3. 内部プログラムのプログラム終わり見出しの次には、以下のいずれかを書かなければなりません。
   - 別の内部プログラムの見出し部の見出し
   - その内部プログラムを直接に含むプログラムの終わり見出し
4. 2つ以上の翻訳単位を続けて書く場合、以下の規則に従ってプログラム終わり見出しを書かなければなりません。
   - 最後の翻訳単位以外では、翻訳単位の終わりを示すプログラム終わり見出しを書かなければなりません。
   - 最後の翻訳単位の終わりを示すプログラム終わり見出しは、省略することができます。

第4章 環境部

環境部には、使用する計算機の特性によって決まる環境を書きます。環境部に書く情報には、以下のものがあります。

- 文字集合の定義
- 文字の大小順序の定義
- 機能名と呼び名の対応付け
- 記号文字および符号系名の定義
- 記号定数の定義
- ファイルと外部媒体の関係付け

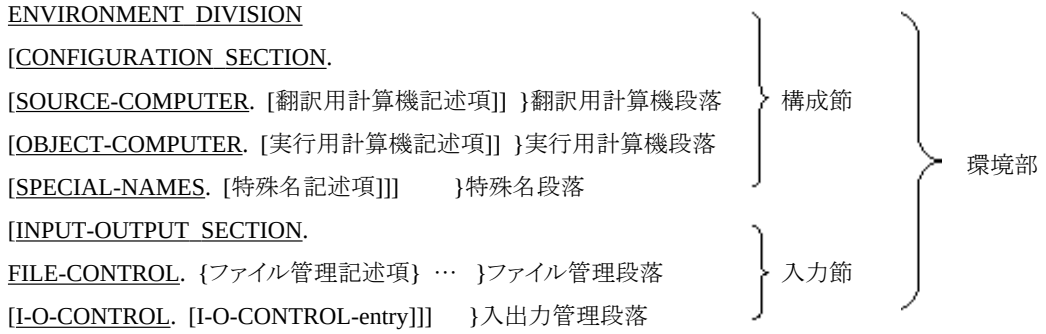
環境部は、見出し部より後に書きます。環境部は、省略することができます。

4.1 環境部の構成（ENVIRONMENT DIVISION）

環境部は、構成節と入出力節との2つの節からなります。

環境部の構成を以下に示します。環境部を構成する節および段落は、以下に示す順に書かなければなりません。

【書き方】



4.2 構成節（CONFIGURATION SECTION）

構成節には、翻訳用計算機と実行用計算機の特性を書きます。

構成節は、一番外側のプログラムにだけ書くことができます。内部プログラムに構成節を書くことはできません。

4.2.1 翻訳用計算機段落（SOURCE-COMPUTER）

翻訳用計算機段落では、プログラムを翻訳するための計算機を指定します。

【書き方】

```
SOURCE-COMPUTER.
    [計算機名 [WITH DEBUGGING MODE句].]
```

構文規則

計算機名の記述規則については、“1.2.2.2 システム名”を参照してください。

一般規則

1. 翻訳用計算機段落のすべての句は、それを明にまたは暗に指定したプログラムおよびその内部プログラムに適用されます。
2. 一番外側のプログラムおよび内部プログラムのどのプログラムにも、翻訳用計算機段落を書かなかった場合、その原始プログラムを翻訳する計算機が翻訳用計算機であるとみなされます。
3. 翻訳用計算機段落の段落名を書き、翻訳用計算機記述項を省略した場合、その原始プログラムを翻訳する計算機が翻訳用計算機であるとみなされます。

4.2.1.1 WITH DEBUGGING MODE句

デバッグ行の扱いを指定します。

【書き方】

```
WITH DEBUGGING MODE
```

一般規則

1. WITH DEBUGGING MODE句を書いた場合、その句を書いたプログラムおよび内部プログラムに書いたデバッグ行は、すべて書いたとおりに翻訳されます。
2. 一番外側のプログラムおよび内部プログラムのどのプログラムにも、WITH DEBUGGING MODE句を書かなかった場合、その原始プログラム中のデバッグ行はすべて注記行とみなされます。
3. 原始プログラムにCOPY文またはREPLACE文を書いた場合、それらの処理が終了した後、WITH DEBUGGING MODE句の有無が検査されます。

4.2.2 実行用計算機段落（OBJECT-COMPUTER）

実行用計算機段落では、プログラムを実行するための計算機を指定します。

【書き方】

```
OBJECT-COMPUTER.
    [計算記名 [MEMORY SIZE句]
        [PROGRAM COLLATING SEQUENCE句].]
```

構文規則

計算機名の記述規則については、“1.2.2.2 システム名”を参照してください。

一般規則

実行用計算機段落のすべての句は、それを明にまたは暗に指定したプログラムおよびその内部プログラムに適用されます。

4.2.2.1 MEMORY SIZE句

記憶容量に関する情報を指定します。MEMORY SIZE句は廃要素です。

【書き方】

MEMORY SIZE 整数-1 { WORDS / CHARACTERS / MODULES }

一般規則

このコンパイラでは、MEMORY SIZE句は注釈とみなされます。

4.2.2.2 PROGRAM COLLATING SEQUENCE句

文字の大小順序を指定します。

使い方の例については、“付録H サンプル集”の“H.9 PROGRAM COLLATING SEQUENCE句”を参照してください。

【書き方】

PROGRAM COLLATING SEQUENCE IS 符号系名-1

構文規則

符号系名-1は、特殊名段落のALPHABET句で文字の大小順序に対応付けなければなりません。

符号系名-1は、ALPHABET句で機能名に対応付けた符号系名であってはいけません。

一般規則

1. PROGRAM COLLATING SEQUENCE句を書いた場合、プログラムの文字の大小順序は、符号系名-1に関係付けた文字の大小順序になります。
2. PROGRAM COLLATING SEQUENCE句を省略した場合、プログラムの文字の大小順序は、固有の文字の大小順序であるとみなされます。
3. PROGRAM COLLATING SEQUENCE句で指定した文字の大小順序は、以下のところで文字比較が行われるとき、条件の真理値を決定するために用いられます。

   ただし、日本語文字比較は除きます。

   a. 明に指定した比較条件
   b. 明に指定した条件名条件
   c. 暗に指定した報告書記述項中のCONTROL句
4. SORT文またはMERGE文でCOLLATING SEQUENCE指定を省略した場合、PROGRAM COLLATING SEQUENCE句で指定したプログラムの文字の大小順序が、SORT文およびMERGE文にも適用されます。
5. PROGRAM COLLATING SEQUENCE句で指定した文字の大小順序は、日本語文字の大小順序に影響を与えません。

4.2.3 特殊名段落（SPECIAL-NAMES）

特殊名段落では、

- 記号定数
- 記号文字
- 符号系名
- 文字集合
- 文字の大小順序

などを定義したり、機能名と呼び名を対応付けたりします。

【書き方】

SPECIAL-NAMES.
[[{機能名-1句} …]
[{機能名-2句} …]
[{機能名-3句} …]
[{ALPHABET句} …]
[{CLASS句} …]
[CRT STATUS句]
[CURRENCY SIGN句]
[CURSOR句]
[DECIMAL-POINT IS COMMA句]
[{POSITIONING UNIT句} …]
[{PRINTING MODE句} …]
[{SYMBOLIC CHARACTERS句} …]
[{SYMBOLIC CONSTANT句} …].]

一般規則

特殊名段落のすべての句は、それを明にまたは暗に指定したプログラム、およびその内部プログラムに適用されます。

4.2.3.1 機能名-1句

データの入出力操作に関連する機能名を、呼び名に対応付けます。

使い方の例については、“付録H サンプル集”の“H.10 機能名-1句”を参照してください。

【書き方】

機能名-1 IS 呼び名-1

構文規則

1. 機能名-1に対応付けた呼び名-1を、ACCEPT文のFROM指定またはDISPLAY文のUPON指定に書く場合、機能名-1は、以下のいずれかでなければなりません。

   ― SYSOUT

   ― CONSOLE

   ― SYSIN

   ― SYSERR

   ― SYSPUNCH

   SYSOUTとSYSPUNCHは同義とみなされます。

2. 機能名-1に対応付けた呼び名-1を、WRITE文のBEFORE ADVANCING指定またはAFTER ADVANCING指定に書く場合、機能名-1は、以下のいずれかでなければなりません。

   ― CHANNEL-01

   ― CHANNEL-02、…、CHANNEL-12

   ― SLC

   ― CTL

   ― STACKER-01、STACKER-02

   CHANNEL-02～CHANNEL-12、STACKER-01およびSTACKER-02は、CHANNEL-01と同義とみなされます。

3. 機能名-1に対応付けた呼び名-1を、CHARACTER TYPE句に書く場合、機能名-1は、以下のいずれかでなければなりません。

   ― HSC

   ― F0202

   ― H0202

   ― F0102

   ― F0201

   ― YX-7P、
   YX-7P-12、YX-7P-21、YX-7P-22、
   YX-7P-H、
   YX-7P-12-H、YX-7P-21-H、YX-7P-22-H、
   YX-9P、YX-9P-12、
   YX-9P-21、YX-9P-22、
   YX-9P-H、YX-9P-22-H、
   YX-9P-12-H、YX-9P-21-H、
   YX-12P、YX-12P-12、
   YX-12P-21、YX-12P-22、
   YX-12P-H、YX-12P-22-H、
   YX-12P-12-H、YX-12P-21-H、
   YA、YA-12、YA-21、YA-22、
   YA-H、YA-12-H、YA-21-H、
   YA-22-H、
   YB、YB-12、YB-21、YB-22、
   YB-H、YB-12-H、YB-21-H、
   YB-22-H、
   YC、YC-12、YC-21、YC-22、
   YC-H、YC-12-H、YC-21-H、
   YC-22-H、
   YD-9P、YD-9P-12、

YD-9P-21、YD-9P-22、
YD-9P-H、YD-9P-22-H、
YD-9P-12-H、YD-9P-21-H、
YD-12P、YD-12P-12、
YD-12P-21、YD-12P-22、
YD-12P-H、YD-12P-22-H、
YD-12P-12-H、YD-12P-21-H、
TX-7P-H、TX-7P-12-H、
TX-7P-21-H、TX-7P-22-H、
TX-7P、TX-7P-12、
TX-7P-21、TX-7P-22、
TX-9P-H、TX-9P-12-H、
TX-9P-21-H、X-9P-22-H、
TX-9P、TX-9P-12、
TX-9P-21、TX-9P-22、
TX-12P-H、TX-12P-12-H、
TX-12P-21-H、TX-12P-22-H、
TX-12P、TX-12P-12、
TX-12P-21、TX-12P-22、
TA、TA-12、TA-21、TA-22、
TB、TB-12、TB-21、TB-22、
TA-H、TA-12-H、TA-21-H、
TA-22-H、
TB-H、TB-12-H、TB-21-H、
TB-22-H、
TC、TC-12、TC-21、TC-22、
TC-H、TC-12-H、TC-21-H、
TC-22-H、
TD-9P、TD-9P-12、
TD-9P-21、TD-9P-22、
TD-9P-H、TD-9P-22-H、
TD-9P-12-H、TD-9P-21-H、
TD-12P、TD-12P-12、
TD-12P-21、TD-12P-22、
TD-12P-H、TD-12P-22-H、
TD-12P-12-H、TD-12P-21-H、
GYX-7P、GYX-7P-12、
GYX-7P-21、GYX-7P-22、
GYX-7P-H、GYX-7P-22-H、
GYX-7P-12-H、GYX-7P-21-H、
GYX-9P、GYX-9P-12、
GYX-9P-21、GYX-9P-22、
GYX-9P-H、GYX-9P-22-H、
GYX-9P-12-H、GYX-9P-21-H、
GYX-12P、GYX-12P-12、
GYX-12P-21、GYX-12P-22、
GYX-12P-H、GYX-12P-22-H、
GYX-12P-12-H、GYX-12P-21-H、
GYA、GYA-12、GYA-21、GYA-22、
GYA-H、GYA-12-H、GYA-21-H、
GYA-22-H、
GYB、GYB-12、GYB-21、GYB-22、
GYB-H、GYB-12-H、GYB-21-H、
GYB-22-H、
GYC、GYC-12、GYC-21、GYC-22、
GYC-H、GYC-12-H、GYC-21-H、
GYC-22-H、

GYD-9P、GYD-9P-12、
GYD-9P-21、GYD-9P-22、
GYD-9P-H、GYD-9P-22-H、
GYD-9P-12-H、GYD-9P-21-H、
GYD-12P、GYD-12P-12、
GYD-12P-21、GYD-12P-22、
GYD-12P-H、GYD-12P-22-H、
GYD-12P-12-H、GYD-12P-21-H、
GTX-7P、GTX-7P-12、
GTX-7P-21、GTX-7P-22、
GTX-9P、GTX-9P-12、
GTX-9P-21、GTX-9P-22、
GTX-12P、GTX-12P-12、
GTX-12P-21、GTX-12P-22、
GTA、GTA-12、GTA-21、GTA-22、
GTB、GTB-12、GTB-21、GTB-22、
GTX-7P-H、GTX-7P-12-H、
GTX-7P-21-H、GTX-7P-22-H、
GTX-9P-H、GTX-9P-12-H、
GTX-9P-21-H、GTX-9P-22-H、
GTX-12P-H、GTX-12P-12-H、
GTX-12P-21-H、GTX-12P-22-H、
GTA-H、GTA-12-H、GTA-21-H、
GTA-22-H、
GTB-H、GTB-12-H、GTB-21-H、
GTB-22-H、
GTC、GTC-12、GTC-21、GTC-22、
GTC-H、GTC-12-H、GTC-21-H、
GTC-22-H、
GTD-9P、GTD-9P-12、
GTD-9P-21、GTD-9P-22、
GTD-9P-H、GTD-9P-22-H、
GTD-9P-12-H、GTD-9P-21-H、
GTD-12P、GTD-12P-12、
GTD-12P-21、GTD-12P-22、
GTD-12P-H、GTD-12P-22-H、
GTD-12P-12-H、GTD-12P-21-H

4.2.3.2 機能名-2句

外部スイッチを使うための機能名を、呼び名に対応付けます。

使い方の例については、“付録H サンプル集”の“H.11 機能名-2句”を参照してください。

【書き方】

機能名-2

{ IS 呼び名-1 [ON STATUS IS 条件名-1 [OFF STATUS IS 条件名-2]]
IS 呼び名-2 [OFF STATUS IS 条件名-2 [ON STATUS IS 条件名-1]]
ON STATUS IS 条件名-1 [OFF STATUS IS 条件名-2]
OFF STATUS IS 条件名-2 [ON STATUS IS 条件名-1]] }

構文規則

1. 機能名-2は、SWITCH-n(nは0～8の整数)でなければなりません。
2. 機能名-2に対応付けた呼び名は、SET文の作用対象に指定することができます。

一般規則

1. SWITCH-nは、外部スイッチを意味します。
   SWITCH-0とSWITCH-8は同義です。同じ外部スイッチを意味します。
2. 外部スイッチのオン状態やオフ状態を調べる場合には、それらを条件名に関連付けます。外部スイッチの検査の方法については、“6.3.3.4 スイッチ状態条件”を参照してください。
3. 外部スイッチの状態は、外部スイッチに関係付けた呼び名を作用対象とするSET文(書き方3のSET文)の実行によって変更することができます。
4. 含むプログラムの特殊名段落で指定した条件名は、含まれるプログラムからも参照することができます。

4.2.3.3 機能名-3句

コマンド行引数と環境変数を操作するための機能名を、呼び名に対応付けます。

使い方の例については、“付録H サンプル集”の“H.12 機能名-3句”を参照してください。

【書き方】

```
機能名-3  IS  呼び名-1
```

構文規則

機能名-3句に指定する名前、およびそれらに対応付けた呼び名を書くことができる場所を下表に示します。

機能名-3句に指定する名前	機能名-3に対応付けた呼び名を書くことができる場所
ARGUMENT-NUMBER	ACCEPT文のFROM指定 および DISPLAY文のUPON指定
ARGUMENT-VALUE	ACCEPT文のFROM指定
ENVIRONMENT-NAME	DISPLAY文のUPON指定
ENVIRONMENT-VALUE	ACCEPT文のFROM指定 および DISPLAY文のUPON指定

一般規則

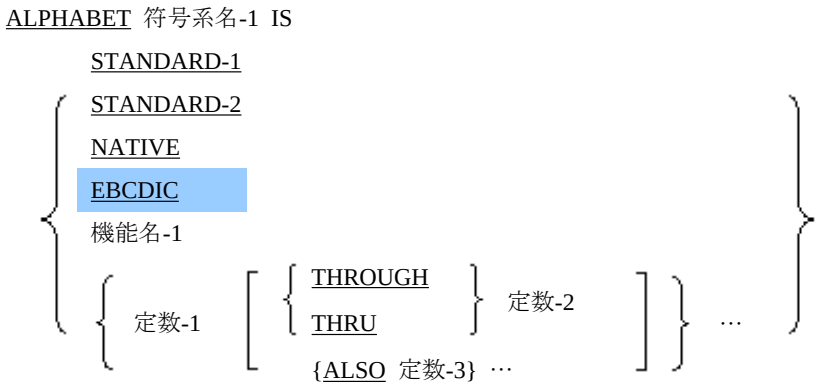
1. ARGUMENT-NUMBERは、コマンド行の引数の個数を入力するため、またはコマンド行の引数を位置付けるために使います。
2. ARGUMENT-VALUEは、コマンド行の引数の値を入力するために使います。
3. ENVIRONMENT-NAMEは、環境変数を位置付けるために使います。
4. ENVIRONMENT-VALUEは、環境変数の値を入力するため、または環境変数に値を設定するために使います。

4.2.3.4 ALPHABET句

符号系名を、文字集合または文字の大小順序に対応付けます。

使い方の例については、“付録H サンプル集”の“H.13 ALPHABET句”を参照してください。

【書き方】



構文規則

1. 機能名-1は、CODE-n(nは0～9の整数)でなければなりません。
2. 定数-1～定数-3は、以下の規則に従うものでなければなりません。
   - 定数-1～定数-3に数字定数を指定する場合、定数は1～256の符号なしの整数でなければなりません。
   - THROUGH指定を書き、定数-1および定数-2に文字定数を指定する場合、定数-1および定数-2の長さは1文字でなければなりません。同様に、ALSO指定を書き、定数-1および定数-3に文字定数を指定する場合、定数-1および定数-2の長さは1文字でなければなりません。
   - 定数-1～定数-3は、記号文字であってはいけません。
   - 定数-1～定数-3は、日本語定数、ブール定数または記号定数であってはいけません。
3. 定数指定の中の各定数に、1つの文字を2回以上指定することはできません。
4. THROUGHとTHRUは同義です。

一般規則

1. ALPHABET句は、文字符号系および文字の大小順序に名前を付けます。
   符号系名-1は、以下の場所に書くことができます。
   a. 実行用計算機段落のPROGRAM COLLATING SEQUENCE句
   b. SORT文のCOLLATING SEQUENCE指定
   c. MERGE文のCOLLATING SEQUENCE指定
   d. ファイル記述項のCODE-SET句
   e. 特殊名段落のSYMBOLIC CHARACTERS句のIN指定
2. STANDARD-1を書いた場合、文字符号系および文字の大小順序は、標準文字集合の規則に従います。標準文字集合の各文字は、固有文字集合の対応する文字に関係付けられます。このコンパイラでは、標準文字集合は、ASCIIに従います。
3. STANDARD-2を書いた場合、文字符号系および文字の大小順序は、国際規格ISO646の規則に従います。標準文字集合の各文字は、固有文字集合の対応する文字に関係付けられます。
4. EBCDICを書いた場合、文字符号系および文字の大小順序は、EBCDICコードの規則に従います。EBCDICコードの各文字は、固有文字集合の対応する文字に関係付けられます。
5. NATIVEを書いた場合またはALPHABET句を省略した場合、文字符号系および文字の大小順序は、固有文字集合および固有の文字の大小順序に従います。
6. 機能名-1に指定するCODE-nは、変換テーブルを意味します。機能名-1を書いた場合、文字符号系および文字の大小順序は、変換テーブルでの定義に従います。

7. 定数指定を書いた場合、文字符号系は固有文字集合になります。文字の大小順序は、定数指定で以下のように定義します。

    a. 定数指定に数字定数を書く場合、各定数に固有文字集合の文字の順序番号を指定します。定数指定に文字定数を書く場合、各定数に、固有文字集合に実在する文字を指定します。文字定数の値が複数個の文字を含む場合は、定数中の各文字の大小順序には、左端から始まって、昇順に連続しているものとします。

    b. 定数指定を2つ以上書いた場合、文字の大小順序は、定数指定を書いた順に昇順になります。

    c. 固有文字集合の文字のうち、定数指定に書かなかった文字は、定数指定に書いたどの文字よりも、大きい順序位置になります。定数指定に書かなかった文字の大小順序は、固有の文字の大小順序に従います。

    d. THROUGH指定を書いた場合、固有文字集合中の定数-1から定数-2までのすべての文字の大小順序は、定数-1から定数-2の順に昇順になります。固有文字集合での文字の大小順序で、定数-1には、定数-2よりも大きい順序位置にある文字を指定することもできます。

    e. ALSO指定を書いた場合、定数-1および定数-3の値によって示される文字は、文字の大小順序で同じ順序位置にあるとみなされます。SYMBOLIC CHARACTERS句のIN指定に符号系名-1を指定した場合、定数-1だけが固有文字集合での文字を表すために使われます。

8. 文字の大小順序で、最大の順序位置を持つ文字が、表意定数HIGH-VALUEの値になります。ただし、この表意定数を、定数-1～定数-3に指定している場合を除きます。最大の文字が2つ以上あるときは、最後に指定した文字が表意定数HIGH-VALUEの値になります。

9. 文字の大小順序で、最小の順序位置を持つ文字が、表意定数LOW-VALUEの値になります。ただし、この表意定数を、定数-1～定数-3に指定している場合を除きます。最小の文字が2つ以上あるときは、最初に指定した文字が表意定数LOW-VALUEの値になります。

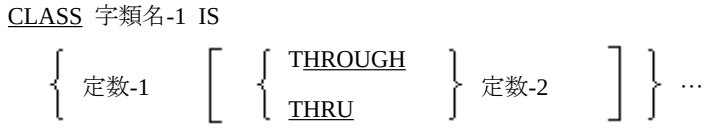
10. 表意定数HIGH-VALUEおよびLOW-VALUEを特殊名段落の定数として指定した場合、それらを参照したときのHIGH-VALUEおよびLOW-VALUEは、それぞれ固有文字集合における最大および最小の文字を表します。

4.2.3.5 CLASS句

字類名を定義します。

使い方の例については、“付録H サンプル集”の“H.14 CLASS句”を参照してください。

【書き方】



構文規則

1. 定数-1および定数-2は、以下の規則に従うものでなければなりません。

    a. 定数-1または定数-2に数字定数を指定する場合、各定数は1～256の符号なし整数でなければなりません。

    b. THROUGH指定を書き、定数-1または定数-2に文字定数を指定する場合、各定数の長さは1文字でなければなりません。

    c. 定数-1および定数-2は、記号文字であってはいけません。

    d. 定数-1および定数-2は、日本語定数、ブール定数または記号定数であってはいけません。

2. THROUGHとTHRUは同義です。

一般規則

1. CLASS句は、文字の組に名前を付けます。字類名-1は、字類条件にだけ書くことができます。

2. 定数-1および定数-2には、以下の定数を指定します。

   a. 定数-1または定数-2に数字定数を指定する場合、各定数に固有文字集合の文字の順序番号を指定します。

   b. 定数-1または定数-2に文字定数を指定する場合、各定数に、固有文字集合に実在する文字を指定します。文字定数の値が複数個の文字を含むならば、その定数中の各文字は字類名によって識別される文字の組に含まれます。

   c. THROUGH指定を書いた場合、固有文字集合の中の定数-1から定数-2までのすべての文字が、字類名-1という名前を持つ文字の組に対応付けられます。さらに、定数-1と定数-2の値の大小関係によって、昇順または降順で固有文字集合の文字を指定することができます。

4.2.3.6 CRT STATUS句

画面入力状態を受け取るためのデータ項目を指定します。

【書き方】

CRT STATUS IS データ名-1

注意

[WinIPF][Linux][LinuxIPF][Linux64][.NET]では、スクリーン操作機能は使用できません。

構文規則

1. データ名-1は、作業場所節で定義した集団項目または英数字項目でなければなりません。データ名-1のデータ項目の文字位置の個数は3文字でなければなりません。
2. データ名-1に集団項目を指定する場合、集団項目の各文字位置は以下のように定義します。
   - 第1文字位置には、1文字の英数字項目、または1桁の符号なし外部10進項目を指定します。
   - 第2文字位置には、1文字の英数字項目、または1桁の符号なし外部10進項目を指定します。
   - 第3文字位置には、1文字の英数字項目を指定します。

一般規則

1. CRT STATUS句は、スクリーン操作機能のACCEPT文で画面入力状態を受け取るためのデータ項目を指定します。
2. データ名-1の内容は、ACCEPT文を実行するごとに設定・更新されます。
3. データ名-1の第1文字位置は、画面入力状態キー1を受け取るための領域です。画面入力状態キー1には、ACCEPT文の終了を引き起こした状態が設定されます。下表に、画面入力状態キー1の意味を示します。

画面入力状態キー1	画面入力状態キー1の意味
0	終了キーまたは最終項目からの戻り
1	利用者定義のファンクションキーによる戻り
2	省略値のファンクションキーによる戻り、または利用者定義のファンクションキーによる戻り
9	エラー

4. 画面入力状態キー1の値“0”は、正常終了を表します。

5. データ名-1の第2文字位置は、画面入力状態キー2を受け取るための領域です。画面入力状態キー2には、ACCEPT文の終了を引き起こした状態の詳細情報のコードが設定されます。下表に、画面入力状態キー2の意味を示します。

画面入力状態キー1	画面入力状態キー2	画面入力状態キー2 の意味
0	0	オペレータによる終了キーの押下
	1	最終項目からの戻り
1	X"00"～X"FF"	ファンクションキー番号
2	X"00"～X"FF"	ファンクションキー番号
9	X"00"	画面内に該当項目がない

6. データ名-1の第3文字位置は、システムが使用する領域です。参照しないでください。

4.2.3.7 CURRENCY SIGN句

通貨編集用文字を指定します。

使い方の例については、“付録H サンプル集”の“H.15 CURRENCY SIGN句”を参照してください。

【書き方】

CURRENCY SIGN IS 定数-1

構文規則

定数-1は、記号文字であってはいけません。

定数-1は、記号定数であってはいけません。

一般規則

1. 定数-1に、PICTURE句で使用する通貨編集用文字を指定します。定数-1は、1文字の文字定数でなければなりません。定数-1に、以下の文字を指定してはいけません。
   - 数字0～9
   - 英大文字A、B、C、D、P、R、S、V、X、Z、英小文字a～z、または空白
   - 英大文字E、G、N
   - 特殊文字“＊”、“＋”、“－”、“,”、“.”、“;”、“(”、“)”、“/”、“＝”または“”“
2. CURRENCY SIGN句を省略した場合、PICTURE句で使う通貨編集用文字は、COBOL文字集合の通貨記号だけであるとみなされます。

4.2.3.8 CURSOR句

カーソル位置の受渡しのためのデータ項目を指定します。

【書き方】

CURSOR IS データ名-1

注意

[WinIPF][Linux][LinuxIPF][Linux64][.NET]では、スクリーン操作機能は使用できません。

構文規則

1. データ名-1は、作業場所節で定義したデータ項目でなければなりません。データ名-1のデータ項目の文字位置の個数は、4文字または6文字でなければなりません。
2. データ名-1は、符号なし外部10進項目、またはそのような基本項目だけを従属する集団項目でなければなりません。

一般規則

1. CURSOR句は、スクリーン操作機能のACCEPT文で、カーソル位置を受け渡すためのデータ項目を指定します。
2. ACCEPT文を実行する前、データ名-1に、位置付けたいカーソル位置を設定します。
3. ACCEPT文の実行後、データ名-1に、利用者が最後に位置付けたカーソルの位置が設定されます。
4. CURSOR句を省略した場合、ACCEPT文の実行前に、カーソルは、画面上の最初の入力項目または更新項目の先頭に位置付けられます。
5. データ名-1に設定する値および返却される値は、以下の意味を持ちます。
   - データ名-1のデータ項目の文字位置の個数が4文字の場合、上位の2文字は行番号を表し、下位の2文字は列位置を表します。
   - データ名-1のデータ項目の文字位置の個数が6文字の場合、上位の3文字は行番号を表し、下位の3文字は列位置を表します。
6. ACCEPTを実行するとき、データ名-1に設定した値が画面上の入力項目および更新項目のどちらの文字位置の範囲にもない場合、カーソルは、画面上の最初の入力項目または更新項目の先頭に位置付けられます。

4.2.3.9 DECIMAL-POINT IS COMMA句

コンマと小数点の機能を入れ替えます。

使い方の例については、“付録H サンプル集”の“H.16 DECIMAL-POINT IS COMMA句”を参照してください。

【書き方】

```
DECIMAL-POINT IS COMMA
```

一般規則

DECIMAL-POINT IS COMMA句を書いた場合、PICTURE句の文字列および数字定数において、コンマの機能と小数点の機能が入れ替わります。

4.2.3.10 POSITIONING UNIT句

位置決め単位名を定義します。

使い方の例については、“付録H サンプル集”の“H.17 POSITIONING UNIT句”を参照してください。

【書き方】

```
POSITIONING UNIT
```

位置決め単位名-1 IS 定数-1 CPI

構文規則

定数-1は、0.01～24.00の符号なし数字定数でなければなりません。定数-1の小数部桁数は2桁以下でなければなりません。

一般規則

1. POSITIONING UNIT句は、印刷装置に印字するときの位置決め単位、すなわち印字行上の印字位置を指定するときの単位に名前を付けます。位置決め単位名-1は、データ記述項のPRINTING POSITION句に指定することができます。詳細については、“5.4.8 PRINTING POSITION句”を参照してください。
2. 定数-1には、位置決め単位の基本単位を指定します。定数-1の単位は、CPI(1インチ当たりの文字数)です。

4.2.3.11 PRINTING MODE句

印字モード名を定義します。

使い方の例については、“付録H サンプル集”の“H.18 PRINTING MODE句”を参照してください。

【書き方】

PRINTING MODE 印字モード名-1

IS [FOR { MOCS / SOCS / ALL }]

[IN SIZE 定数-1 POINT]

[AT PITCH 定数-2 CPI]

[WITH FONT 機能名-1]

[AT ANGLE 整数-1 DEGREES]

[BY FORM 機能名-2]

構文規則

1. 定数-1は、3.0～300.0の符号なし数字定数でなければなりません。定数-1の小数部桁数は、1桁以下でなければなりません。
2. 定数-2は、0.01～24.00の符号なし数字定数でなければなりません。定数-2の小数部桁数は2桁以下でなければなりません。
3. 機能名-1は、以下のいずれかでなければなりません。

MINCHOU	明朝体
MINCHOU-HANKAKU	明朝体半角
GOTHIC	ゴシック体
GOTHIC-HANKAKU	ゴシック体半角
GOTHIC-DP	ゴシックDP
FONT-nnn	システムが定める文字書体番号(nnnはシステムによって定められた書体番号001～999)

4. 整数-1は、0または90でなければなりません。
5. 機能名-2は、以下のいずれかでなければなりません。

F0102	全角長体
F0201	全角平体
F0202	全角倍角
H0102	半角長体
H0201	半角平体
H0202	半角倍角
H	半角
F	全角

6. FOR指定を省略した場合、FOR MOCSを指定したものとみなされます。
7. FONT指定を省略した場合、以下のように指定したものとみなされます。
   - 日本語項目または日本語編集項目を印字する場合、WITH FONT MINCHOUを指定したものとみなれます。
   - 英字項目、英数字項目、英数字編集項目、外部10進項目、外部ブール項目、外部浮動小数点項目または数字編集項目を印字する場合、WITH FONT GOTHICを指定したものとみなされます。
8. ANGLE指定を省略した場合、AT ANGLE 0 DEGREESを指定したものとみなされます。
9. FORM指定を省略した場合、BY FORM Fを指定したものとみなされます。

一般規則

1. PRINTING MODE句は、印刷装置に印字するときの印字モードに、印字モード名-1という名前を付けます。「印字モード」とは、文字サイズ、文字ピッチ、文字書体、文字回転量および文字形態のことです。印字モード名-1は、CHARACTER TYPE句に指定することができます。
2. FOR指定では、印字モード名-1がどの項目の印字モードであるかを指定します。
   - 日本語項目および日本語編集項目に対する印字モードを指定する場合、FOR MOCS指定を書きます。
   - 英字項目、英数字項目、英数字編集項目、外部10進項目、外部ブール項目、外部浮動小数点項目および数字編集項目に対する印字モードを指定する場合、FOR SOCS指定を書きます。
   - 日本語項目、日本語編集項目、英字項目、英数字項目、英数字編集項目、外部10進項目、外部ブール項目、外部浮動小数点項目および数字編集項目に対する印字モードを指定する場合、FOR ALL指定を書きます。
3. SIZE指定の定数-1には、文字サイズを指定します。文字サイズの単位はポイントです。
4. PITCH指定の定数-2には、文字ピッチを指定します。文字ピッチの単位はCPI(1インチ当たりの文字数)です。
5. SIZE指定または PITCH指定のいずれかを省略した場合、文字サイズおよび文字ピッチは、以下のようになります。
   - PITCH指定を省略した場合、SIZE指定値に合わせた文字ピッチをPITCH指定値とします。
   - SIZE指定を省略した場合、PITCH指定値に合わせた文字サイズをSIZE指定値とします。
6. SIZE指定およびPITCH指定の両方を省略した場合、文字サイズおよび文字ピッチは、以下のようになります。
   - 英字項目、英数字項目、英数字編集項目、外部10進項目、外部ブール項目、外部浮動小数点項目または数字編集項目を印字する場合、文字サイズは7ポイント、文字ピッチは10CPIになります。
   - 日本語項目または日本語編集項目を印字する場合、文字サイズは12ポイント、文字ピッチは文字サイズに合わせた文字ピッチになります。
7. FONT指定の機能名-1では、文字書体を指定します。
8. ANGLE指定の整数-1には、文字回転量を反時計回りの回転角度で指定します。整数-1に0を指定した場合、横書きになります。整数-1に90を指定した場合、縦書きになります。

9. FORM指定の機能名-2では、文字形態を指定します。

4.2.3.12 SYMBOLIC CHARACTERS句

記号文字を定義します。

使い方の例については、“付録H サンプル集”の“H.19 SYMBOLIC CHARACTERS句”を参照してください。

【書き方】

SYMBOLIC CHARACTERS

{ {記号文字-1} … { IS / ARE } {整数-1} … } … [IN 符号系名-1]

構文規則

1. 記号文字-1に、同じ記号文字を2回以上指定することはできません。
2. 記号文字-1の並びと整数-1の並びは、1番目の記号文字-1と1番目の整数-1、2番目の記号文字-1と2番目の整数-1というように、記号文字-1と整数-1が対になるように指定します。
3. 記号文字-1の並びの個数と整数-1の並びの個数は、同じでなければなりません。
4. IN指定を省略する場合、整数-1は、固有文字集合に存在する文字の順序位置でなければなりません。IN指定を書く場合、整数-1は、符号系名-1に指定した文字集合に存在する文字の順序位置でなければなりません。
5. 符号系名-1は、ALPHABET句で機能名に対応付けた符号系名であってはいけません。

一般規則

1. IN指定なしのSYMBOLIC CHARACTERS句は、固有文字集合中の、整数-1に指定した順序位置を持つ文字に、記号文字-1という名前を付けます。
2. IN指定付きのSYMBOLIC CHARACTERS句は、符号系名-1に指定した文字集合中の、整数-1に指定した順序位置を持つ文字に、記号文字-1という名前を付けます。

4.2.3.13 SYMBOLIC CONSTANT句

記号定数を定義します。

使い方の例については、“付録H サンプル集”の“H.20 SYMBOLIC CONSTANT句”を参照してください。

【書き方】

SYMBOLIC CONSTANT

{記号定数-1 IS 定数-1} …

構文規則

定数-1は、記号文字または記号定数であってはいけません。

一般規則

1. SYMBOLIC CONSTANT句は、定数-1に、記号定数-1という名前を付けます。

2. 記号定数-1は、以下の場所にだけ書くことができます。

   ー 環境部では、POSITIONING UNIT句の定数、およびPRINTING MODE句の定数および整数

   ー データ部では、OCCURS句の整数、およびVALUE句の定数

   ー 手続き部では、定数が書けるところすべて

4.3 入出力節（INPUT-OUTPUT SECTION）

入出力節には、外部媒体と実行用プログラムとの間で、データを転送し処理するために必要な情報を書きます。

4.3.1 ファイル管理段落（FILE-CONTROL）

ファイル管理段落では、ファイルを外部媒体に関係付けます。

【書き方】

FILE-CONTROL.

{ファイル管理記述項} …

ファイル管理段落には、いくつかのファイル管理記述項を書くことができます。ファイル管理記述項に指定する句および句の規則は、ファイルの種類によって異なります。

順ファイル、相対ファイルおよび索引ファイル機能のファイル管理記述項

順ファイル、相対ファイルおよび索引ファイル機能のファイル管理記述項では、各ファイルに関係する物理属性を定義します。

【書き方1】順ファイル

SELECT句
[ACCESS MODE句]
ASSIGN句
[FILE STATUS句]
[FORMAT句]
[GROUP句]
[LOCK MODE句]
[ORGANIZATION句]
[PADDING CHARACTER句]
[RECORD DELIMITER句]
[RESERVE句].

【書き方2】相対ファイル

SELECT句
[ACCESS MODE句]
ASSIGN句

[FILE STATUS句]

[LOCK MODE句]

ORGANIZATION句

[RESERVE句].

【書き方3】索引ファイル

SELECT句

[ACCESS MODE句]

[{ALTERNATE RECORD KEY句} …]

ASSIGN句

[FILE STATUS句]

[LOCK MODE句]

ORGANIZATION句

RECORD KEY句

[RESERVE句].

構文規則

1. SELECT句は、ファイル管理記述項の最初に書かなければなりません。SELECT句の後に書く句の順序は任意です。
2. データ部のファイル記述項で指定するファイル名は、同じプログラムのファイル管理段落で1回だけ定義しなければなりません。

一般規則

1. SELECT句に指定するファイルのファイル結合子が外部ファイル結合子の場合、そのファイル結合子を参照する実行単位中のすべてのファイル管理記述項は、以下の規則に従わなければなりません。
    a. SELECT句のOPTIONAL指定の有無は同じでなければなりません。
    b. ASSIGN句にデータ名以外を指定する場合、指定する内容は同じでなければなりません。
    c. RESERVE句に指定する整数の値は同じでなければなりません。
    d. ORGANIZATION句で、同じ編成を指定しなければなりません。
    e. ACCESS MODE句で、同じ呼出し法を指定しなければなりません。
    f. LOCK MODE句で、同じロックの方法を指定しなければなりません。
    g. 相対ファイルの場合、ACCESS MODE句のRELATIVE KEY指定に指定するデータ名は、同じ外部データ項目でなければなりません。
    h. 索引ファイルの場合、さらに以下の規則に従わなければなりません。
        - RECORD KEY句に指定するデータ名は、SELECT句のファイル名に関連するレコード中で同じ相対位置を持つものでなければなりません。また、そのデータ名のデータ記述項は、同じ内容で定義しなければなりません。
        - RECORD KEY句に指定するデータ名の個数は、同じでなければなりません。
        - RECORD KEY句のDUPLICATES指定の有無は、同じでなければなりません。
        - ALTERNATE RECORD KEY句に指定するデータ名は、SELECT句のファイル名に関連するレコード中で同じ相対位置を持つものでなければなりません。また、そのデータ名のデータ記述項は、同じ内容で定義しなければなりません。

- ALTERNATE RECORD KEY句に指定するデータ名の個数は、同じでなければなりません。
- ALTERNATE RECORD KEY句のDUPLICATES指定の有無は、同じでなければなりません。

2. 索引ファイルの参照キーの値は、固有文字集合の文字の大小順序に従って比較されます。

整列併合機能のファイル管理記述項

整列併合機能のファイル管理記述項は、整列併合用ファイルに関係する物理属性を定義します。

【書き方】

SELECT句
ASSIGN句.

構文規則

1. ファイル管理記述項の最初にSELECT句を書き、その後にASSIGN句を書かなければなりません。
2. データ部のファイル記述項で指定するファイル名は、同じプログラムのファイル管理段落で1回だけ定義しなければなりません。
3. このコンパイラでは、整列併合用ファイルに対するASSIGN句は注釈とみなされます。

表示ファイル機能のファイル管理記述項

表示ファイル機能のファイル管理記述項では、表示ファイルに関係する物理属性を定義します。

【書き方】

SELECT句
[ACCESS MODE句]
ASSIGN句
[DESTINATION句]
[END KEY句]
[FILE STATUS句]
[FORMAT句]
[GROUP句]
[MESSAGE CLASS句]
[MESSAGE CODE句]
[MESSAGE MODE句]
[MESSAGE OWNER句]
[MESSAGE SEQUENCE句]
[ORGANIZATION句]
[PROCESSING CONTROL句]
[PROCESSING MODE句]
[PROCESSING TIME句]
[SELECTED FUNCTION句]
[SESSION CONTROL句]
[SYMBOLIC DESTINATION句]

[UNIT CONTROL句].

構文規則

1. SELECT句は、ファイル管理記述項の最初に書かなければなりません。SELECT句の後に書く句の順序は、任意です。
2. データ部のファイル記述項で指定するファイル名は、同じプログラムのファイル管理段落で1回だけ定義しなければなりません。
3. あて先種別と指定可能な句のシステムごとの組合せを、下表に示します。

句	システム	あて先種別				
		DSP	PRT	ACM	APL	TRM
DESTINATION-1句	DS	○	○	○	○	○
	HP	□	□	□	□	□
	Solaris	□	□	○	□	□
	Win16	□	□	○	―	―
	Win32	□	□	○	□	□
	Linux	□	□	―	―	―
	LinuxIPF	―	□	―	―	―
	Linux64	□	□	―	―	―
	.NET	―	□	―	―	―
	WinIPF	―	□	―	―	―
	Winx64	□	□	―	―	―
DESTINATION-2句	DS	○	○	○	○	○
	HP	□	□	□	□	□
	Solaris	□	□	○	□	□
	Win16	□	□	■	―	―
	Win32	□	□	○	□	□
	Linux	□	□	―	―	―
	LinuxIPF	―	□	―	―	―
	Linux64	□	□	―	―	―
	.NET	―	□	―	―	―
	WinIPF	―	□	―	―	―
	Winx64	□	□	―	―	―
DESTINATION-3句	DS	○	○	■	○	○
	HP	□	□	■	□	□
	Solaris	□	□	■	□	□
	Win16	□	□	■	―	―
	Win32	□	□	■	□	□
	Linux	□	□	―	―	―
	LinuxIPF	―	□	―	―	―

句	システム	あて先種別				
		DSP	PRT	ACM	APL	TRM
	Linux64	□	□	—	—	—
	.NET	—	□	—	—	—
	WinIPF	—	□	—	—	—
	Winx64	□	□	—	—	—
END KEY句	DS	■	■	■	○	○
	HP	■	■	■	□	□
	Solaris	■	■	■	□	□
	Win16	■	■	■	—	—
	Win32	■	■	■	□	○
	Linux	■	■	—	—	—
	LinuxIPF	—	■	—	—	—
	Linux64	■	■	—	—	—
	.NET	—	■	—	—	—
	WinIPF	—	■	—	—	—
	Winx64	■	■	—	—	—
FILE STATUS句	DS	○	○	○	○	○
	HP	○	○	□	□	□
	Solaris	○	○	○	□	○
	Win16	○	○	○	—	—
	Win32	○	○	○	□	□
	Linux	○	○	—	—	—
	LinuxIPF	—	○	—	—	—
	Linux64	○	○	—	—	—
	.NET	—	○	—	—	—
	WinIPF	—	○	—	—	—
	Winx64	—	○	—	—	—
FORMAT句	DS	○	○	■	■	○
	HP	○	○	■	■	□
	Solaris	○	○	■	■	□
	Win16	○	○	■	—	—
	Win32	○	○	■	■	□
	Linux	○	○	—	—	—
	LinuxIPF	—	○	—	—	—
	Linux64	○	○	—	—	—
	.NET	—	○	—	—	—
	WinIPF	—	○	—	—	—
	Winx64	○	○	—	—	—
GROUP句	DS	○	○	■	■	○

句	システム	あて先種別				
		DSP	PRT	ACM	APL	TRM
	HP	○	○	■	■	□
	Solaris	○	○	■	■	□
	Win16	○	○	■	－	－
	Win32	○	○	■	■	□
	Linux	○	○	－	－	－
	LinuxIPF	－	○	－	－	－
	Linux64	○	○	－	－	－
	.NET	－	○	－	－	－
	WinIPF	－	○	－	－	－
	Winx64	○	○	－	－	－
MESSAGE CLASS句	DS	■	■	○	■	■
	HP	■	■	□	■	■
	Solaris	■	■	○	■	■
	Win16	■	■	○	－	－
	Win32	■	■	○	■	■
	Linux	■	■	－	－	－
	LinuxIPF	－	■	－	－	－
	Linux64	■	■	－	－	－
	.NET	－	■	－	－	－
	WinIPF	－	■	－	－	－
	Winx64	■	■	－	－	－
MESSAGE CODE句	DS	■	■	■	○	■
	HP	■	■	■	□	■
	Solaris	■	■	■	□	■
	Win16	■	■	■	－	－
	Win32	■	■	■	□	■
	Linux	■	■	－	－	－
	LinuxIPF	－	■	－	－	－
	Linux64	■	■	－	－	－
	.NET	－	■	－	－	－
	WinIPF	－	■	－	－	－
	Winx64	■	■	－	－	－
MESSAGE MODE句	DS	○	○	■	○	○
	HP	□	□	■	□	□
	Solaris	□	□	■	□	□
	Win16	■	■	■	－	－
	Win32	□	□	■	□	□
	Linux	□	□	－	－	－

句	システム	あて先種別				
		DSP	PRT	ACM	APL	TRM
	LinuxIPF	―	□	―	―	―
	Linux64	□	□	―	―	―
	.NET	―	□	―	―	―
	WinIPF	―	□	―	―	―
	Winx64	□	□	―	―	―
MESSAGE OWNER句	DS	○	■	■	○	○
	HP	□	■	■	□	□
	Solaris	□	■	■	□	□
	Win16	■	■	■	―	―
	Win32	□	■	■	□	□
	Linux	□	■	―	―	―
	LinuxIPF	―	■	―	―	―
	Linux64	□	■	―	―	―
	.NET	―	■	―	―	―
	WinIPF	―	■	―	―	―
	Winx64	□	■	―	―	―
MESSAGE SEQUENCE句	DS	×	○	×	×	×
	HP	×	□	×	×	×
	Solaris	×	□	×	×	×
	Win16	×	□	×	―	―
	Win32	×	□	×	×	×
	Linux	×	□	―	―	―
	LinuxIPF	―	□	―	―	―
	Linux64	×	□	―	―	―
	.NET	―	□	―	―	―
	WinIPF	―	□	―	―	―
	Winx64	×	□	―	―	―
PROCESSING CONTROL句	DS	○	○	■	○	○
	HP	□	□	■	□	□
	Solaris	□	□	■	□	□
	Win16	□	□	■	―	―
	Win32	□	□	■	□	□
	Linux	□	□	―	―	―
	LinuxIPF	―	□	―	―	―
	Linux64	□	□	―	―	―
	.NET	―	□	―	―	―
	WinIPF	―	□	―	―	―
	Winx64	□	□	―	―	―

句	システム	あて先種別				
		DSP	PRT	ACM	APL	TRM
PROCESSING MODE句	DS	○	○	○	■	○
	HP	○	○	□	■	○
	Solaris	○	○	○	■	○
	Win16	○	○	○	—	—
	Win32	○	○	○	■	□
	Linux	○	○	—	—	—
	LinuxIPF	—	○	—	—	—
	Linux64	○	○	—	—	—
	.NET	—	○	—	—	—
	WinIPF	—	○	—	—	—
	Winx64	○	○	—	—	—
PROCESSING TIME句	DS	■	■	○	■	■
	HP	■	■	□	■	■
	Solaris	■	■	□	■	■
	Win16	■	■	○	—	—
	Win32	■	■	○	■	■
	Linux	■	■	—	—	—
	LinuxIPF	—	■	—	—	—
	Linux64	■	■	—	—	—
	.NET	—	■	—	—	—
	WinIPF	—	■	—	—	—
	Winx64	■	■	—	—	—
SELECTED FUNCTION句	DS	○	○	■	■	○
	HP	○	□	■	■	□
	Solaris	○	□	■	■	□
	Win16	○	□	■	—	—
	Win32	○	□	■	■	□
	Linux	○	□	—	—	—
	LinuxIPF	—	□	—	—	—
	Linux64	○	□	—	—	—
	.NET	—	□	—	—	—
	WinIPF	—	□	—	—	—
	Winx64	○	□	—	—	—
SESSION CONTROL句	DS	○	○	■	○	○
	HP	□	□	■	□	□
	Solaris	□	□	■	□	□
	Win16	□	□	■	—	—
	Win32	□	□	■	□	□

句	システム	あて先種別				
		DSP	PRT	ACM	APL	TRM
	Linux	□	□	―	―	―
	LinuxIPF	―	□	―	―	―
	Linux64	□	□	―	―	―
	.NET	―	□	―	―	―
	WinIPF	―	□	―	―	―
	Winx64	□	□	―	―	―
UNIT CONTROL句	DS	○	○	■	■	■
	HP	○	○	■	■	■
	Solaris	○	○	■	■	■
	Win16	○	○	■	―	―
	Win32	○	○	■	■	■
	Linux	○	○	―	―	―
	LinuxIPF	―	○	―	―	―
	Linux64	○	○	―	―	―
	.NET	―	○	―	―	―
	WinIPF	―	○	―	―	―
	Winx64	○	○	―	―	―

○ ： 指定可能（必ず指定しなければならない場合も含む）で、機能的な意味があります。
□ ： 指定可能ですが、機能的な意味はありません。指定された場合、注釈として扱います。
■ ： 指定することができない組合せです。指定された場合、注釈として扱います。
× ： 指定することができない組合せです。
― ： 組合せとして意味がありません。

一般規則

SELECT句に指定するファイルのファイル結合子が外部ファイル結合子の場合、そのファイル結合子を参照する実行単位中のすべてのファイル管理記述項は、以下の規則に従わなければなりません。

1. ASSIGN句に指定するファイル識別名は、同じでなければなりません。
2. ORGANIZATION句で、同じ編成を指定しなければなりません。
3. ACCESS MODE句で、同じ呼出し法を指定しなければなりません。
4. 以下の句の指定の有無は、同じでなければなりません。
   - ― SYMBOLIC DESTINATION句
   - ― DESTINATION-1句
   - ― DESTINATION-2句
   - ― DESTINATION-3句
   - ― END KEY句
   - ― FORMAT句
   - ― GROUP句
   - ― MESSAGE CLASS句
   - ― MESSAGE CODE句

- MESSAGE MODE句
- MESSAGE OWNER句
- PROCESSING CONTROL句
- PROCESSING MODE句
- PROCESSING TIME句
- SELECTED FUNCTION句
- SESSION CONTROL句
- UNIT CONTROL句

5. 以下の句に指定するデータ項目は、同じ長さでなければなりません。
   - MESSAGE CODE句に指定するデータ項目
   - PROCESSING CONTROL句に指定するデータ項目
   - UNIT CONTROL句に指定するデータ項目

報告書作成機能のファイル管理記述項

報告書作成機能のファイル管理記述項では、報告書ファイルに関係する物理属性を定義します。

【書き方】

SELECT句
[ACCESS MODE句]
ASSIGN句
[FILE STATUS句]
[ORGANIZATION句]
[RESERVE句].

構文規則

1. SELECT句は、ファイル管理記述項の最初に書かなければなりません。SELECT句の後に書く句の順序は、任意です。
2. データ部のファイル記述項で指定するファイル名は、同じプログラムのファイル管理段落で1回だけ定義しなければなりません。また、報告書ファイルのファイル記述項では、REPORT句を書かなければなりません。

一般規則

SELECT句に指定するファイルのファイル結合子が外部ファイル結合子の場合、そのファイル結合子を参照する実行単位中のすべてのファイル管理記述項は、以下の規則に従わなければなりません。

1. SELECT句のOPTIONAL指定の有無は同じでなければなりません。
2. ASSIGN句に指定するファイル識別名は、同じでなければなりません。
3. RESERVE句に指定する整数の値は同じでなければなりません。
4. ORGANIZATION句で、同じ編成を指定しなければなりません。
5. ACCESS MODE句で、同じ呼出し法を指定しなければなりません。

4.3.1.1 ACCESS MODE句（順ファイル・相対ファイル・索引ファイル・表示ファイル・報告書作成）

ファイルの呼出し法を指定します。

表示ファイルは、富士通拡張仕様です。

【書き方1】順ファイル・表示ファイル・報告書作成

```
ACCESS MODE IS SEQUENTIAL
```

【書き方2】相対ファイル

```
ACCESS MODE IS
    { SEQUENTIAL          [RELATIVE KEY IS データ名-1]   }
    { { RANDOM    }       RELATIVE KEY IS データ名-1     }
    { { DYNAMIC   }                                      }
```

【書き方3】索引ファイル

```
ACCESS MODE IS { SEQUENTIAL }
               { RANDOM     }
               { DYNAMIC    }
```

構文規則

書き方2の規則

1. データ名-1は、修飾することができます。
2. データ名-1は、符号なし整数項目でなければなりません。ただし、USAGE句にBINARY-CHAR UNSIGNEDを指定した項目であってはなりません。データ名-1のデータ記述項のPICTURE文字列に“P”を指定することはできません。
3. データ名-1は、以下のいずれかで定義しなければなりません。
   - 作業場所節
   - 連絡節
4. ファイルをSORT文またはMERGE文のUSING指定またはGIVING指定に書く場合、そのファイルに対するACCESS MODE句にRANDOMまたはDYNAMICを書くことはできません。
5. ファイルをSTART文に書く場合、そのファイルに対するACCESS MODE句にRELATIVE KEY指定を書かなければなりません。

書き方3の規則

ファイルをSORT文またはMERGE文のUSING指定またはGIVING指定に書く場合、そのファイルに対するACCESS MODE句にRANDOMおよびDYNAMICを書くことはできません。

一般規則

書き方1～書き方3に共通する規則

1. ACCESS MODE IS SEQUENTIALは、順呼出し法でレコードを呼び出すことを指定します。ACCESS MODE IS SEQUENTIALを指定したファイルおよびACCESS MODE句を省略したファイルを、「順呼出し法のファイル」といいます。
   ACCESS MODE IS RANDOMは、乱呼出し法でレコードを呼び出すことを指定します。ACCESS MODE IS RANDOMを指定したファイルを、「乱呼出し法のファイル」といいます。
   ACCESS MODE IS DYNAMICは、動的呼出し法でレコードを呼び出すことを指定します。ACCESS MODE IS DYNAMICを指定したファイルを、「動的呼出し法のファイル」といいます。
2. ACCESS MODE句を省略した場合、ACCESS MODE IS SEQUENTIALを指定したものとみなされます。
3. 関連するファイル結合子が外部ファイル結合子の場合、そのファイル結合子に関連する実行単位中のすべてのファイル管理記述項では、同じ呼出し法を指定しなければなりません。書き方2の場合、さらに、データ名-1は同じ外部データ項目でなければなりません。

書き方1の規則

ファイルのレコードは、ファイル編成に従った順序で呼び出されます。ファイル編成に従った順序とは、ファイルを生成または拡張するときにレコードを書き出した順のことです。

書き方2の規則

1. 順呼出し法のファイルのレコードは、ファイル編成に従った順序で呼び出されます。ファイル編成に従った順序とは、ファイルに存在するレコードの相対レコード番号の昇順です。
2. 乱呼出し法の場合、呼び出すレコードの相対レコード番号を、データ名-1に設定します。
3. 動的呼出し法のファイルのレコードは、順呼出し法によっても乱呼出し法によっても呼び出すことができます。
4. 相対ファイルに格納されているレコードは、すべて、相対レコード番号で一意に識別されます。相対レコード番号は、そのレコードがファイル中で論理的に何番目の位置にあるかを示します。最初のレコードの相対レコード番号は1であり、それに続くレコードの相対レコード番号は、2、3、4、…というように昇順になります。
5. データ名-1のデータ項目は、プログラムと入出力管理システムの間で相対レコード番号を連絡するために使います。

書き方3の規則

1. 順呼出し法のファイルのレコードは、ファイル編成に従った順序で呼び出されます。ファイル編成に従った順序とは、そのファイルの文字の大小順序に従った参照キーの値の昇順です。
2. 乱呼出し法の場合、呼び出すレコードを、主レコードキー項目または副レコードキー項目で指定します。
3. 動的呼出し法のファイルのレコードは、順呼出し法によっても乱呼出し法によっても呼び出すことができます。

4.3.1.2 ALTERNATE RECORD KEY句（索引ファイル）

副レコードキーを指定します。

【書き方】

{ALTERNATE RECORD KEY IS
{データ名-1} … [WITH DUPLICATES]} …

構文規則

1. データ名-1は、修飾することができます。
2. データ名-1は、ALTERNATE RECORD KEY句を指定したファイル名に関連するレコード記述項で定義しなければなりません。
3. データ名-1に指定可能な項目属性は、ファイルシステムに依存します。詳細については、"NetCOBOL使用手引書"を参照してください。

4. 索引ファイルが可変長レコードを含む場合、各副レコードキーは可変長レコードの最初のn文字の中に含まれなければなりません。ここでnは、そのファイルの最小のレコードの大きさです。詳細については、“5.2.9 RECORD句（順ファイル・相対ファイル・索引ファイル）”を参照してください。
5. データ名-1は、レコード中の可変位置にあってはいけません。
6. データ名-1のデータ項目の長さおよび個数の最大値については“付録B システムの定量制限”を参照してください。

一般規則

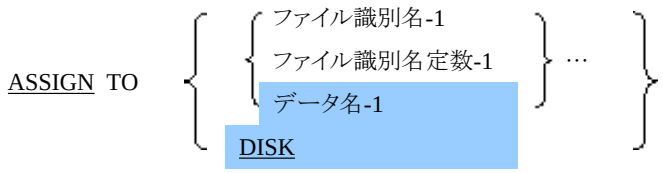
1. ALTERNATE RECORD KEY句は、この句に関連するファイルの副レコードキーを指定します。
2. 以下の内容は、ファイルを生成したときのものと同じでなければなりません。
   - データ名-1のデータ記述項
   - データ名-1のレコード中での相対位置
   - データ名-1の個数
   - ALTERNATE RECORD KEY句の個数
   - DUPLICATES指定の有無
3. DUPLICATES指定を書いた場合、ファイル中に、副レコードキーの値の等しいレコードを2つ以上含むことができます。DUPLICATES指定を省略した場合、ファイル中に、副レコードキーの値の等しいレコードを2つ以上含むことはできません。
4. 1つのファイルに2つ以上のレコード記述項を関係付ける場合、データ名-1は、レコード記述項の1つだけに書かなければなりません。そのレコード記述項以外のどのレコード記述項でも、データ名-1と同じ文字位置は、暗にキーとして参照されます。
5. 関連するファイル結合子が外部ファイル結合子の場合、その実行単位でそのファイル結合子に関連するすべてのファイル記述項では、以下の規則に従わなければなりません。
   a. データ名-1は、レコード中の同じ相対位置になければなりません。
   b. データ名-1のデータ記述項は同じ内容でなければなりません。
   c. データ名-1の個数は同じでなければなりません。
   d. DUPLICATES指定の有無は同じでなければなりません。
   e. ALTERNATE RECORD KEY句の個数は、同じでなければなりません。
6. データ名-1を2つ以上書いた場合、書いた順にそれらを連結したものが、1つの副レコードキーとして扱われます。

4.3.1.3 ASSIGN句（順ファイル・相対ファイル・索引ファイル）

ファイルを外部媒体に関連付けます。

【書き方1】順ファイル

【書き方2】相対ファイル・索引ファイル



注意

PRINTER-n指定は、[Win16][Win32][Solaris][Linux][LinuxIPF][Linux64][.NET][WinIPF][Winx64]固有機能です。

構文規則

1. データ名-1は、DISK、PRINTERまたはPRINTER-nであってはいけません。
2. データ名-1は、256文字以下の英数字項目または集団項目でなければなりません。
   データ名-1は、以下のいずれかで定義しなければなりません。
   - 作業場所節
   - 連絡節
3. ファイル識別名-1の形式を以下に示します。
   - 書き方1の場合

     ```
     [注釈-] 名前
     ```

   - 書き方2の場合

     ```
     名前
     ```

   名前には、ファイルをシステムに通知するための外部名を指定します。名前は、英字と数字だけから構成される、8文字以下の文字列でなければなりません。名前の最初の文字は、英字でなければなりません。
4. ファイル識別名定数-1の記述規則については、“1.2.6 特殊な用途の定数”を参照してください。
5. FORMAT句付きの印刷ファイルに、DISK、PRINTERまたはPRINTER-nを指定することはできません。
6. PRINTER-nのnは、1～9の整数でなければなりません。

一般規則

1. ASSIGN句は、SELECT句に書いたファイル名を、記憶媒体に関連付けます。
2. ファイル識別名-1、ファイル識別名定数-1またはデータ名-1を2つ以上書いた場合、2番目以降の記述は注釈とみなされます。
3. ファイル識別名-1を書いたASSIGN句は、SELECT句に書いたファイル名を、ファイル識別名-1に対応する物理ファイルに関連付けます。ファイル識別名-1には、記憶媒体上の物理ファイルに関係付けた間接的な名前を指定します。
4. ファイル識別名定数-1を書いたASSIGN句は、SELECT句に書いたファイル名を、ファイル識別名定数-1の値が示す物理ファイルに関連付けます。ファイル識別名定数-1には、記憶媒体上の物理ファイルの名前を直接指定します。
5. データ名-1を書いたASSIGN句は、SELECT句に書いたファイル名をデータ名-1の値が示す物理ファイルに関連付けます。データ名-1を指定した場合、OPEN文を実行するときに、記憶媒体上の物理ファイル名を直接示す値を、データ名-1に設定しておかなければなりません。OPEN文を実行すると、ファイル名がデータ名-1の値が示す記憶媒体上の物理ファイルに関連付けられます。

6. DISK指定を書いたASSIGN句は、SELECT句に書いたファイル名が、現在の環境の大記憶装置上の物理ファイル名であることを示します。
7. PRINTERまたはPRINTER-n指定を書いたASSIGN句は、SELECT句に書いたファイル名を印刷装置に関連付けます。
8. ASSIGN句の内容と物理ファイルとの関係については、“NetCOBOL使用手引書”を参照してください。

4.3.1.4 ASSIGN句（整列併合・報告書作成）

ファイルを外部媒体に関連付けます。

【書き方】

ASSIGN TO { ファイル識別名-1 / ファイル識別名定数-1 } …

構文規則

ファイル識別名-1の形式を以下に示します。

```
[注釈-] 名前
```

名前には、ファイルをシステムに通知するための外部名を指定します。名前は、英字と数字だけから構成される、8文字以下の文字列でなければなりません。名前の最初の文字は、英字でなければなりません。

一般規則

1. ASSIGN句は、SELECT句に書いたファイル名を、ファイル識別名-1によって参照される記憶媒体と関連付けます。
2. ファイル識別名-1を2つ以上書いた場合、2番目以降の記述は注釈とみなされます。
3. 整列併合用ファイルに対するASSIGN句は、注釈とみなされます。

4.3.1.5 ASSIGN句（表示ファイル）

ファイルを外部媒体に関連付けます。

【書き方】

ASSIGN TO {ファイル識別名-1} …

構文規則

ファイル識別名-1の形式を、以下に示します。

```
GS-名前
```

名前には、ファイルをシステムに通知するための外部名を指定します。名前は、英字と数字だけから構成される、8文字以下の文字列でなければなりません。名前の最初の文字は、英字でなければなりません。

一般規則

1. ASSIGN句は、SELECT句に書いたファイル名を、ファイル識別名-1によって参照される外部媒体と関連付けます。
2. ファイル識別名-1を2つ以上書いた場合、2番目以降の記述は注釈とみなされます。

4.3.1.6 DESTINATION句（表示ファイル）

あて先名を設定、参照するためのデータ項目を指定します。

【書き方】

[DESTINATION-1 IS データ名-1]
[DESTINATION-2 IS データ名-2]
[DESTINATION-3 IS データ名-3]

構文規則

1. データ名-1、データ名-2およびデータ名-3は、項類が英数字の8文字のデータ項目でなければなりません。
   このデータ項目は、以下のいずれかで定義しなければなりません。
   - 作業場所節
   - 連絡節
2. データ名-1、データ名-2およびデータ名-3は、修飾することができます。
3. データ名-1、データ名-2およびデータ名-3のデータ項目は、レコード中の可変位置にあってはなりません。

一般規則

1. データ名-1、データ名-2およびデータ名-3は、メッセージのあて先名を指定します。あて先名は、表示ファイルに対する入出力文を実行するときに、システムへ通知されます。また、あて先名は、READ文の実行が成功すると、システムにより設定されます。
   なお、これらの句の意味は、システム、入出力文およびメッセージ種別に依存します。詳細については、“NetCOBOL使用手引書”を参照してください。
2. DESTINATION句を省略すると、あて先名として空白がシステムに通知されます。

4.3.1.7 END KEY句（表示ファイル）

セグメント情報を設定、参照するためのデータ項目を指定します。

【書き方】

END KEY IS データ名-1

注意

[.NET]では、END KEY句は指定できません。

構文規則

1. データ名-1は、項類が英数字の1文字のデータ項目でなければなりません。
   このデータ項目は、以下のいずれかで定義しなければなりません。
   - 作業場所節
   - 連絡節

2. データ名-1は、修飾することができます。
3. データ名-1のデータ項目は、レコード中の可変位置にあってはなりません。

一般規則

1. データ名-1のデータ項目には、セグメント情報(メッセージがセグメント化されるか否か、およびセグメント化される場合に先頭、中間、末尾のどのセグメントかの情報)を指定します。セグメント情報は、READ文の実行が成功すると、システムにより設定されます。また、セグメント情報は、WRITE文を実行するときに、システムに通知されます。
2. END KEY句を省略するとWRITE命令の実行時に、メッセージはセグメント化されていないものとして、システムに通知されます。

4.3.1.8 FILE STATUS句(順ファイル・相対ファイル・索引ファイル・表示ファイル・報告書作成)

入出力状態を参照するためのデータ項目を指定します。

表示ファイルは、富士通拡張仕様です。

【書き方】

FILE STATUS IS データ名-1 [データ名-2]

構文規則

1. 以下は修飾することができます。
   - データ名-1
   - データ名-2
2. データ名-1は、2文字の英数字項目または集団項目でなければなりません。
   データ名-1は、以下のいずれかで定義しなければなりません。
   - 作業場所節
   - 連絡節
3. データ名-2は、4文字の英数字項目または集団項目でなければなりません。
   データ名-2は、以下のいずれかで定義しなければなりません。
   - 作業場所節
   - 連絡節
4. データ名-1およびデータ名-2は、レコード中の可変位置にあってはいけません。
5. データ名-2は、以下の場合だけ書くことができます。
   - FORMAT句付きの印刷ファイルのFILE STATUS句
   - 表示ファイルのFILE STATUS句

一般規則

1. データ名-1には、入出力状態を参照するためのデータ項目を指定します。

   データ名-2には、入出力状態の詳細情報を参照するためのデータ項目を指定します。

2. FILE STATUS句を指定したファイルに対する入出力文を実行すると、データ名-1に、入出力状態の値と詳細情報の値が設定されます。

   データ名-2にも、入出力状態の値と詳細情報の値が設定されます。

   入出力状態と詳細情報の値については“NetCOBOL使用手引書”を参照してください。

4.3.1.9 FORMAT句（順ファイル・表示ファイル）

画面帳票定義体名を設定するためのデータ項目を指定します。

【書き方】

FORMAT IS データ名-1

構文規則

1. データ名-1は、修飾することができます。
2. SYMBOLIC DESTINATION句に、明にあて先種別がプリンタ装置であることを指定した場合には、データ名-1は、30文字以下の英数字項目または集団項目でなければなりません。その他の場合には、データ名-1は、8文字の英数字項目または集団項目でなければなりません。データ名-1は、以下のいずれかで定義しなければなりません。
   - 作業場所節
   - 連絡節
3. データ名-1は、レコード中の可変位置にあってはいけません。
4. 順ファイルの場合、FORMAT句は、印刷ファイルに対してだけ指定することができます。FORMAT句を指定した印刷ファイルを「FORMAT句付きの印刷ファイル」といいます。

一般規則

1. データ名-1には、画面帳票定義体名を設定するためのデータ項目を指定します。FORMAT句を指定したファイルに対するWRITE文を実行すると、データ名-1に設定した画面帳票定義体名がシステムに通知されます。
2. データ名-1に空白以外の値を設定した場合、その値が指定する画面帳票定義体に従って、レコードの編集が行われます。
3. データ名-1に空白を設定した場合、レコードの編集は行われません。
4. 表示ファイルの場合、FORMAT句を省略すると、画面帳票定義体名として空白がシステムに通知されます。
5. 画面帳票定義体名については、“NetCOBOL使用手引書”を参照してください。

4.3.1.10 GROUP句（順ファイル・表示ファイル）

項目群名または項目名を設定するためのデータ項目を指定します。

【書き方】

GROUP IS データ名-1

構文規則

1. データ名-1は、修飾することができます。

2. データ名-1は、8文字の英数字項目または集団項目でなければなりません。
   データ名-1は、以下のいずれかで定義しなければなりません。
   - 作業場所節
   - 連絡節
3. データ名-1は、レコード中の可変位置にあってはいけません。
4. GROUP句は、FORMAT句を指定したファイルに対してだけ指定することができます。

一般規則

1. データ名-1には、項目群名または項目名を設定するためのデータ項目を指定します。データ名-1に設定した項目群名または項目名は、FORMAT句を指定したファイルに対する入出力文を実行するときに、システムに通知されます。
2. データ名-1に空白以外の値を設定した場合、その値が指定する項目群または項目が入出力の対象になります。
3. GROUP句を省略した場合、項目群名または項目名として空白がシステムに通知されます。
4. 項目群名または項目名については、“NetCOBOL使用手引書”を参照してください。

4.3.1.11 LOCK MODE句（順ファイル・相対ファイル・索引ファイル）

ファイルのロックの方法を指定します。

【書き方1】順ファイル

```
LOCK MODE IS { AUTOMATIC [WITH LOCK ON RECORD]
             { EXCLUSIVE
```

【書き方2】相対ファイル・索引ファイル

```
LOCK MODE IS { MANUAL WITH LOCK ON MULTIPLE RECORDS
             { AUTOMATIC [WITH LOCK ON RECORD]
             { EXCLUSIVE
```

一般規則

書き方1および書き方2に共通する規則

1. LOCK MODE句を省略した場合、ファイルが入力モードで開かれていなければ、開かれるファイルは排他モードになります。入力モードで開かれるファイルは、WITH LOCK指定なしのOPEN文の実行が成功すると、共用モードになります。
2. LOCK MODE IS EXCLUSIVEを指定した場合、OPEN文の実行が成功すると、そのファイルは排他モードで開いた状態になります。
3. LOCK MODE IS AUTOMATICを指定した場合、WITH LOCK指定なしのOPEN文の実行が成功すると、そのファイルは共用モードで開いた状態になります。
4. WITH LOCK ON RECORD指定は、注釈とみなされます。
5. 排他モードで開いたファイルは、他のファイル結合子によって開くことはできません。
6. 共用モードで開いたファイルは、排他モードを要求しない他のファイル結合子によって開くことができます。
7. 大記憶装置上に存在しないファイルを共用モードで開くことはできません。

8. 出力モードで開かれるファイルは、LOCK MODE句の指定にかかわらず排他モードになります。

書き方1の規則

LOCK MODE IS AUTOMATICを指定した場合、READ文の実行が成功すると、レコードのロックが取得されます。その後、そのファイル結合子に対する入出力文の実行が成功すると、レコードのロックが解除されます。

書き方2の規則

1. LOCK MODE IS MANUALを指定した場合、WITH LOCK指定のないOPEN文の実行が成功すると、そのファイルは共用モードで開いた状態になります。
2. LOCK MODE IS MANUALを指定した場合、レコードのロックは、READ WITH LOCK文によって取得されます。
3. LOCK MODE IS AUTOMATICを指定した場合、READ文の実行が成功すると、レコードのロックが取得されます。その後、そのファイル結合子に対する入出力文の実行が成功すると、レコードのロックが解除されます。

4.3.1.12 MESSAGE CLASS句（表示ファイル）

非同期型メッセージ通信において、メッセージの優先順位を設定するためのデータ項目を指定します。

【書き方】

MESSAGE CLASS データ名-1

注意

[.NET]では、MESSAGE CLASS句は指定できません。

構文規則

1. データ名-1は、その記述中にPICTURE文字“P”を含まない1桁の符号なし外部10進整数項目でなければなりません。
2. データ名-1は、以下のいずれかで定義しなければなりません。
   - 作業場所節
   - 連絡節
3. データ名-1は、修飾することができます。
4. データ名-1のデータ項目は、レコード中の可変位置にあってはなりません。

一般規則

1. データ名-1のデータ項目には、非同期型メッセージ通信における、メッセージの優先順位を指定します。
2. データ名-1のデータ項目に指定する値は、1～9の範囲でなければなりません。値1は、最も高い優先順位であることを意味します。
3. データ名-1のデータ項目の値は、WRITE文でメッセージを送信するときに有効となり、送信されるメッセージは、このデータ項目で指定した処理の優先順位を持ちます。
4. 優先順位を指定して送信されたメッセージは、READ文でそれらのメッセージを受信するときにそのメッセージが持つ優先順位に従って受信されます。
5. 同じ優先順位を持つメッセージは、送信された順に受信されます。
6. MESSAGE CLASS句を省略すると、そのメッセージは最も低い優先順位となります。

4.3.1.13 MESSAGE CODE句（表示ファイル）

理由コードを設定、参照するためのデータ項目を指定します。

【書き方】

MESSAGE CODE IS データ名-1

注意

[.NET]では、MESSAGE CODE句は指定できません。

構文規則

1. データ名-1は、項類が英数字の1～32767文字のデータ項目または符号なし4桁の外部10進項目でなければなりません。
   データ項目は、以下のいずれかで定義しなければなりません。
   - 作業場所節
   - 連絡節
2. データ名-1は、修飾することができます。
3. データ名-1のデータ項目は、レコード中の可変位置にあってはなりません。

一般規則

1. データ名-1のデータ項目は、会話を強制終了するための理由コードを指定します。理由コードは、READ文の実行が成功すると、システムにより設定されます。また、理由コードは、WRITE文を実行するときに、システムに通知されます。
2. MESSAGE CODE句を省略すると、理由コードとして空白がシステムに通知されます。

4.3.1.14 MESSAGE MODE句（表示ファイル）

メッセージ種別を設定、参照するためのデータ項目を指定します。

【書き方】

MESSAGE MODE IS データ名-1

注意

[.NET]では、MESSAGE MODE句は注釈とみなされます。

構文規則

1. データ名-1は、項類が英数字の1文字のデータ項目でなければなりません。
   このデータ項目は、以下のいずれかで定義しなければなりません。
   - 作業場所節

ー 連絡節

2. データ名-1は、修飾することができます。
3. データ名-1のデータ項目は、レコード中の可変位置にあってはなりません。

一般規則

1. データ名-1のデータ項目には、メッセージ種別(入力メッセージ、応答メッセージ、強制メッセージ、送信結果要求通知、会話強制終了などの別)を指定します。メッセージ種別は、READ文の実行が成功すると、システムにより設定されます。また、メッセージ種別は、WRITE文を実行するときに、システムに通知されます。
2. メッセージ種別に指定できる文字はシステムにより異なります。

4.3.1.15 MESSAGE OWNER句（表示ファイル）

送信権情報を設定、参照するためのデータ項目を指定します。

【書き方】

MESSAGE OWNER IS データ名-1

[.NET]では、MESSAGE OWNER句は指定できません。

構文規則

1. データ名-1は、項類が英数字の1文字のデータ項目でなければなりません。
   このデータ項目は、以下のいずれかで定義しなければなりません。
   ー 作業場所節
   ー 連絡節
2. データ名-1は、修飾することができます。
3. データ名-1のデータ項目は、レコード中の可変位置にあってはなりません。

一般規則

1. データ名-1のデータ項目には、送信権情報を指定します。READ文の実行が成功すると、送信権情報として、送信権の有無がシステムにより設定されます。また、WRITE文を実行するときに、送信権を会話相手のプログラムに受け渡すかどうかをシステムに通知します。
2. MESSAGE OWNER句を省略すると、送信権情報として空白がシステムに通知されます。

4.3.1.16 MESSAGE SEQUENCE句（表示ファイル）

帳票の作成通番を含むデータ項目を指定します。

【書き方】

MESSAGE SEQUENCE IS データ名-1

注意

[.NET]では、MESSAGE SEQUENCE句は指定できません。

構文規則

1. データ名-1は、その記述中にPICTURE文字“P”を含まない8桁の符号なし外部10進整数項目でなければなりません。
2. データ名-1は、以下のいずれかで定義しなければなりません。
   - 作業場所節
   - 連絡節
3. データ名-1は修飾することができます。
4. データ名-1のデータ項目は、レコード中の可変位置にあってはなりません。

一般規則

1. WRITE文の実行が成功すると、システムによりデータ名-1のデータ項目に作成通番が設定されます。
2. WRITE文の実行が失敗すると、データ名-1のデータ項目の内容は不定となります。

4.3.1.17 ORGANIZATION句（順ファイル）

ファイルの論理的構成が、以下のどちらであるかを指定します。

- 順編成
- 行順編成

【書き方】

[ORGANIZATION IS] [LINE] SEQUENTIAL

一般規則

1. ORGANIZATION IS SEQUENTIAL句は、ファイルの論理的構成が順編成であることを指定します。ファイル編成は、ファイルを生成するときに決まり、後で変更することはできません。
2. ORGANIZATION IS LINE SEQUENTIAL句は、ファイルの論理的構成が行順編成であることを指定します。ファイル編成は、ファイルを生成する時に決まり、後で変更することはできません。行順編成のファイルの各レコードは、区切り文字で区切られます。1つのレコードは、印刷可能な文字とレコードの区切り文字だけを含むことができます。1つのレコードは1行と数えます。
3. ORGANIZATION句を省略した場合、ORGANIZATION IS SEQUENTIAL句を指定したものとみなされます。

4.3.1.18 ORGANIZATION句（相対ファイル）

ファイルの論理的構成として相対編成を指定します。

【書き方】

[ORGANIZATION IS] RELATIVE

一般規則

ORGANIZATION IS RELATIVE句は、ファイルの論理的構成が相対編成であることを指定します。ファイル編成は、ファイルを生成する時に決まり、後で変更することはできません。

4.3.1.19 ORGANIZATION句（索引ファイル）

ファイルの論理的構成として索引編成を指定します。

【書き方】

[ORGANIZATION IS] INDEXED

一般規則

ORGANIZATION IS INDEXED句は、ファイルの論理的構成が索引編成であることを指定します。ファイル編成は、ファイルを生成するときに決まり、後で変更することはできません。

4.3.1.20 ORGANIZATION句（表示ファイル・報告書作成）

ファイルの論理的構成として順編成を指定します。

表示ファイルは、富士通拡張仕様です。

【書き方】

[ORGANIZATION IS] SEQUENTIAL

一般規則

1. ORGANIZATION IS SEQUENTIAL句は、ファイルの論理的構成が順編成であることを指定します。ファイル編成は、ファイルを生成するときに決まり、後で変更することはできません。
2. ORGANIZATION IS SEQUENTIAL句を省略した場合、この句を指定したものとみなされます。

4.3.1.21 PADDING CHARACTER句（順ファイル）

ブロックの埋め草文字を指定します。

【書き方】

PADDING CHARACTER IS { データ名-1 / 定数-1 }

一般規則

このコンパイラでは、PADDING CHARACTER句は注釈とみなされます。

4.3.1.22 PROCESSING CONTROL句（表示ファイル）

拡張メッセージ制御情報を設定、参照するためのデータ項目を指定します。

【書き方】

PROCESSING CONTROL IS データ名-1

構文規則

1. データ名-1は、項類が英数字の128文字以下のデータ項目でなければなりません。
   このデータ項目は、以下のいずれかで定義しなければなりません。
   - 作業場所節
   - 連絡節
2. データ名-1は、修飾することができます。
3. データ名-1のデータ項目は、レコード中の可変位置にあってはなりません。

一般規則

1. データ名-1のデータ項目には、システム固有のメッセージ制御情報を指定します。データ名-1のデータ項目の形式については、“NetCOBOL使用手引書”を参照してください。
2. READ文の実行が成功すると、システムによりデータ名-1のデータ項目に制御情報が設定されます。また、データ名-1のデータ項目の値は、WRITE文を実行するときに、システムに通知されます。

4.3.1.23 PROCESSING MODE句（表示ファイル）

処理種別を設定するためのデータ項目を指定します。

【書き方】

PROCESSING MODE IS データ名-1

構文規則

1. データ名-1は、2文字の英数字項目または集団項目でなければなりません。
   データ名-1は、以下のいずれかで定義しなければなりません。
   - 作業場所節
   - 連絡節
2. データ名-1は、修飾することができます。
3. データ名-1は、レコード中の可変位置にあってはいけません。

一般規則

1. データ名-1には、処理種別を設定するためのデータ項目を指定します。処理種別とは、アラーム鳴動入力、全画面消去入出力などの種別のことです。表示ファイルに対する入出力文を実行すると、データ名-1に設定した処理種別がシステムに通知されます。
2. PROCESSING MODE句を省略した場合、処理種別として空白がシステムに通知されます。

3. 処理種別については、“NetCOBOL使用手引書”を参照してください。

4.3.1.24 PROCESSING TIME句（表示ファイル）

非同期型メッセージ通信におけるメッセージ送受信の待ち合わせ時間を設定するためのデータ項目を指定します。

【書き方】

PROCESSING TIME データ名-1

注意

[.NET]では、PROCESSING TIME句は指定できません。

構文規則

1. データ名-1は、その記述中にPICTURE文字“P”を含まない4桁の符号なし外部10進整数項目でなければなりません。
2. データ名-1は、以下のいずれかで定義しなければなりません。
   - 作業場所節
   - 連絡節
3. データ名-1は、修飾することができます。
4. データ名-1に指定するデータ項目は、レコード中の可変位置にあってはなりません。

一般規則

1. データ名-1のデータ項目には、メッセージを送信または受信するときの待ち合わせ時間を秒単位で指定します。
2. データ名-1のデータ項目の値は、0～9999の範囲でなければなりません。値0は、無限の待ち時間を意味します。
3. データ名-1のデータ項目の値は以下の場合にだけ有効であり、このデータ項目で指定した時間だけメッセージの送受信を待つことを意味します。
   - 処理種別に待ち合わせ指示を指定したREAD文またはWRITE文
4. PROCESSING TIME句を省略すると、メッセージの待ち時間は無限の値となります。

4.3.1.25 RECORD DELIMITER句（順ファイル）

外部記憶媒体上の可変長レコードの長さを決める方法を指定します。

【書き方】

RECORD DELIMITER IS STANDARD-1

一般規則

このコンパイラでは、RECORD DELIMITER句は注釈とみなされます。

4.3.1.26 RECORD KEY句（索引ファイル）

主レコードキーを指定します。

【書き方】

RECORD KEY IS {データ名-1} …
[WITH DUPLICATES]

構文規則

1. データ名-1は、修飾することができます。
2. データ名-1は、RECORD KEY句を指定したファイル名に関連するレコード記述項で定義しなければなりません。
3. データ名-1に指定可能な項目属性は、ファイルシステムに依存します。詳細については、“NetCOBOL使用手引書”を参照してください。
4. 索引ファイルが可変長レコードを含むとき、主レコードキーはそのレコードの最初のn文字の中に含まれなければなりません。ここでnは、そのファイルの最小のレコードの大きさとします。詳細については、“5.2.9 RECORD句（順ファイル・相対ファイル・索引ファイル）”を参照してください。
5. データ名-1は、レコード中の可変位置にあってはいけません。
6. データ名-1のデータ項目の長さおよび個数の最大値については、“付録B システムの定量制限”を参照してください。

一般規則

1. RECORD KEY句は、この句に関連するファイルの主レコードキーを指定します。
2. DUPLICATES指定を書いた場合、主レコードキーの値は、ファイル中のすべてのレコードで一意である必要はありません。DUPLICATES指定を省略した場合、主レコードキーの値は、ファイル中のすべてのレコードで一意でなければなりません。
3. 以下の内容は、ファイルを生成したときのものと同じでなければなりません。
   a. データ名-1のデータ記述項
   b. データ名-1のレコード中での相対位置
   c. データ名-1の個数
   d. DUPLICATES指定の有無
4. 1つのファイルに2つ以上のレコード記述項を関係付ける場合、データ名-1はそれらのレコード記述項の1つだけに書かなければなりません。そのレコード記述項以外のどのレコード記述項でも、データ名-1と同じ文字位置が、暗にキーとして参照されます。
5. 関連するファイル結合子が外部ファイル結合子の場合、その実行単位でそのファイル結合子に関連するすべてのファイル記述項では、以下の規則に従わなければなりません。
   a. データ名-1は、レコード中の同じ相対位置になければなりません。
   b. データ名-1のデータ記述項は同じ内容でなければなりません。
   c. データ名-1の個数は同じでなければなりません。
   d. DUPLICATES指定の有無は同じでなければなりません。
6. データ名-1を2つ以上書いた場合、データ名-1を書いた順にそれらを連結したものが、1つの主レコードキーとして扱われます。

4.3.1.27 RESERVE句（順ファイル・相対ファイル・索引ファイル・報告書作成）

入出力領域の個数を指定します。

【書き方】

RESERVE 整数-1 [AREA | AREAS]

一般規則

このコンパイラでは、RESERVE句は注釈とみなされます。

4.3.1.28 SELECT句（順ファイル・相対ファイル・索引ファイル・報告書作成）

ファイル名を宣言します。

【書き方】

SELECT [OPTIONAL] ファイル名-1

構文規則

1. SELECT句は、ファイル管理記述項の最初に書かなければなりません。
2. ファイル名-1のファイル記述項は、SELECT句を書いたプログラムの中に書かなければなりません。

一般規則

1. OPTIONAL指定は、ファイル名-1が必ずしも存在するファイルではないことを指定します。OPTIONAL指定を書いたファイルを、「不定ファイル」といいます。
2. OPTIONAL指定は、以下のファイルに対してだけ指定することができます。
   - 順ファイル、相対ファイルおよび索引ファイルの場合、入力モード、入出力モードまたは拡張モードで開くファイル
   - 報告書ファイルの場合、拡張モードで開くファイル

4.3.1.29 SELECT句（整列併合・表示ファイル）

ファイル名を宣言します。

表示ファイルは、富士通拡張仕様です。

【書き方】

SELECT ファイル名-1

構文規則

1. SELECT句は、ファイル管理記述項の最初に書かなければなりません。
2. ファイル名-1のファイル記述項は、SELECT句を書いたプログラムの中に書かなければなりません。

4.3.1.30 SELECTED FUNCTION句（表示ファイル）

アテンション種別を参照するためのデータ項目を指定します。

【書き方】

SELECTED FUNCTION IS データ名-1

構文規則

1. データ名-1は、4文字の英数字項目または集団項目でなければなりません。
   データ名-1は、以下のいずれかで定義しなければなりません。
   - 作業場所節
   - 連絡節
2. データ名-1は、修飾することができます。
3. データ名-1は、レコード中の可変位置にあってはいけません。

一般規則

1. データ名-1には、アテンション種別を参照するためのデータ項目を指定します。READ文が成功すると、アテンション種別がシステムから通知され、データ名-1に設定されます。
2. データ名-1には、アテンション種別として以下の値が設定されます。
   - 画面定義時に変更可能なキー(ファンクションキーなど)
   - ENTERキー
   - 項目セレクト
3. アテンション種別については、“NetCOBOL使用手引書”を参照してください。

4.3.1.31 SESSION CONTROL句（表示ファイル）

セション情報を設定、参照するためのデータ項目を指定します。

【書き方】

SESSION CONTROL IS データ名-1

構文規則

1. データ名-1は、項類が英数字の1文字のデータ項目でなければなりません。
   このデータ項目は、以下のいずれかで定義しなければなりません。
   - 作業場所節
   - 連絡節

2. データ名-1は、修飾することができます。
3. データ名-1のデータ項目は、レコード中の可変位置にあってはなりません。

一般規則

1. データ名-1のデータ項目には、セション情報を指定します。セション情報は、READ文の実行が成功すると、システムにより設定されます。また、セション情報は、WRITE文を実行するときに、システムに通知されます。
2. セション情報としては、データ名-1のデータ項目の値により、次のものが指定できます。
   - 会話の開始
   - 会話の終了
   - 会話の継続
3. SESSION CONTROL句を省略すると、WRITE文の実行時に、会話の継続である旨がシステムに通知されます。

4.3.1.32 SYMBOLIC DESTINATION句（表示ファイル）

あて先種別を指定します。

【書き方】

SYMBOLIC DESTINATION IS { データ名-1 / 定数-1 }

構文規則

1. データ名-1は、3文字の英数字項目または集団項目でなければなりません。
   データ名-1は、以下のいずれかで定義しなければなりません。
   - 作業場所節
   - 連絡節
2. データ名-1は、修飾することができます。
3. データ名-1は、レコード中の可変位置にあってはいけません。

一般規則

1. データ名-1には、あて先種別を設定・参照するためのデータ項目を指定します。定数-1には、あて先種別を設定します。入出力文を実行すると、データ名-1または定数-1に設定したあて先種別がシステムに通知されます。
2. データ名-1および定数-1に設定できる値を、下表に示します。

データ名-1または定数-1の値	あて先種別
DSP (*1)	ディスプレイ装置
PRT	プリンタ装置
ACM (*2)	非同期型メッセージ通信
APL (*3)	プログラム間通信
TRM (*4)	ディスプレイ装置およびプリンタ装置以外の端末装置

*1 ： [DS][HP][Win16][Win32][Winx64][Solaris][Linux][Linux64]で指定できます。
*2 ： [DS][Win16][Win32][Winx64][Solaris][Linux64]で指定できます。

*3 ： [DS][Win32][Solaris]で指定できます。
*4 ： [DS][Win32]で指定できます。

3. SYMBOLIC DESTINATION句を省略した場合、あて先種別はディスプレイ装置である旨が、システムに通知されます。

4.3.1.33 UNIT CONTROL句（表示ファイル）

ユニット制御情報を設定するためのデータ項目を指定します。

【書き方】

UNIT CONTROL IS データ名-1

構文規則

1. データ名-1は、6文字の英数字項目または集団項目でなければなりません。
   データ名-1は、以下のいずれかで定義しなければなりません。
   - 作業場所節
   - 連絡節
2. データ名-1は、修飾することができます。
3. データ名-1は、レコード中の可変位置にあってはいけません。

一般規則

1. データ名-1には、装置に依存した特殊制御を行うためのユニット制御情報を設定するためのデータ項目を指定します。WRITE文を実行すると、データ名-1に設定したユニット制御情報がシステムに通知されます。
2. ユニット制御情報については、“NetCOBOL使用手引書”を参照してください。

4.3.2 入出力管理段落（I-O-CONTROL）

入出力管理段落では、実行用プログラムで使う特殊な制御技法を指定します。

【書き方】

I-O-CONTROL.
[[APPLY { MULTICONVERSATION-MODE / MULTICON } 句]
[APPLY SAVED-AREA句]
[{MULTIPLE FILE TAPE句} …]
[{RERUN句} …]
[{SAME句} …].]

注意

APPLY MULTICONVERSATION-MODE句は[HP][DS]固有機能です。

APPLY SAVED-AREA句は[DS]固有機能です。

構文規則

1. 各句を書く順序は任意です。
2. 各句には、2つ以上の異なる編成のファイルを書くことができます。どの編成のファイルを書くことができるかは、句によって異なります。下表に、入出力管理段落に書くことができる句を示します。

入出力管理段落の句	順ファイル	相対ファイル	索引ファイル	整列併合用ファイル	表示ファイル	報告書ファイル
APPLY MULTICON句	×	×	×	×	○	×
APPLY SAVED-AREA句	×	×	×	×	○	×
MULTIPLE FILE TAPE句	□	×	×	×	×	□
RERUN句	□	□	□	×	×	×
SAME句	○	○	○	○	○	○

○ ： その編成のファイルを書くことができます。
× ： その編成のファイルを書くことはできません。
□ ： その編成のファイルを書くことができますが、注釈とみなされます。

一般規則

このコンパイラでは、MULTIPLE FILE TAPE句およびRERUN句は注釈とみなされます。

4.3.2.1 APPLY MULTICONVERSATION-MODE句（表示ファイル）

複数のあて先に対して多重会話処理を行います。

【書き方】

APPLY { MULTICONVERSATION-MODE / MULTICON }

注意

APPLY MULTICONVERSATION-MODE句は[HP][DS]固有機能です。

構文規則

1. MULTICONVERSATION-MODEとMULTICONは、同義語です。
2. APPLY MULTICONVERSATION-MODE句を、内部プログラムに書くことはできません。

一般規則

1. APPLY MULTICONVERSATION-MODE句は、複数のあて先に対して多重会話処理を行うことを指定します。APPLY MULTICONVERSATION-MODE句を指定すると、複数のあて先との会話処理におけるプログラムの実行の流れが、システムによって自動的に行われます。したがって、利用者は単一のあて先に対する処理を行うのと同様に入出力文を記述することができます。

2. APPLY MULTICONVERSATION-MODE句を指定した場合には、APPLY SAVED-AREA句を指定しなければなりません。
3. APPLY MULTICONVERSATION-MODE句はその句を書いたプログラム、およびその内部プログラムで定義された、すべての表示ファイルに対して有効です。

4.3.2.2 APPLY SAVED-AREA句（表示ファイル）

複数の会話にまたがってデータ項目の値を保証します。

【書き方】

APPLY SAVED-AREA TO {データ名-1} …

注意

APPLY SAVED-AREA句は[DS]固有機能です。

構文規則

1. データ名-1は、レベル番号が01または77の固定長のデータ項目でなければなりません。
   データ名-1は、作業場所節で定義しなければなりません。
2. データ名-1およびデータ名-1に従属するデータ項目に、VALUE句を指定してはいけません。
3. APPLY SAVED-AREA句を、内部プログラムに書くことはできません。

一般規則

1. APPLY SAVED-AREA句は、複数の会話にまたがってデータ名-1の値を保証することを指定します。
2. APPLY SAVED-AREA句は、その句を書いたプログラムおよびその内部プログラムで定義した、すべての表示ファイルに対して有効です。

4.3.2.3 MULTIPLE FILE TAPE句（順ファイル・報告書作成）

複数ファイルリール上のファイルの位置を指定します。MULTIPLE FILE TAPE句は、廃要素です。

【書き方】

MULTIPLE FILE TAPE CONTAINS
{ファイル名-1 [POSITION 整数-1]} …

一般規則

このコンパイラでは、MULTIPLE FILE TAPE句は注釈とみなされます。

4.3.2.4 RERUN句（順ファイル・相対ファイル・索引ファイル）

再開が行われる時点を指定します。RERUN句は、廃要素です。

【書き方】

```
RERUN ON ファイル識別名-1 EVERY
      { [END OF] { REEL } }
      {          { UNIT } } OF ファイル名-1
      { 整数-1 RECORDS    }
```

一般規則

このコンパイラでは、RERUN句は注釈とみなされます。

4.3.2.5 SAME句（順ファイル・相対ファイル・索引ファイル・整列併合・表示ファイル・報告書作成）

異なるファイル間での記憶領域の共用を指定します。

表示ファイルは、富士通拡張仕様です。

【書き方1】ファイルを共用する（SAME AREA句）

```
SAME AREA FOR ファイル名-1 {ファイル名-2} …
```

【書き方2】レコードを共用する（SAME RECORD AREA句）

```
SAME RECORD AREA FOR ファイル名-1 {ファイル名-2} …
```

【書き方3】整列併合用ファイルの記憶領域を最適に割り当てる（SAME SORT / SORT-MERGE AREA句）

```
SAME { SORT       } AREA
     { SORT-MERGE }
       FOR ファイル名-1 {ファイル名-2} …
```

注意

[.NET]では、SAME句は指定できません。

構文規則

1. ファイル名-1およびファイル名-2は、同じプログラムのファイル管理段落に書かなければなりません。
2. ファイル名-1およびファイル名-2のファイル結合子は、外部ファイル結合子であってはいけません。
3. ファイル名-1およびファイル名-2は、以下の編成のファイルでなければなりません。
    a. 書き方1の場合、順ファイル、相対ファイル、索引ファイル、表示ファイルまたは報告書ファイル
    b. 書き方2の場合、順ファイル、相対ファイル、索引ファイル、表示ファイルまたは整列併合用ファイル
    c. 書き方3の場合、整列併合用ファイル、順ファイル、相対ファイルまたは索引ファイル

4. ファイル名-1およびファイル名-2は、同じ編成法または同じ呼出し法である必要はありません。
5. SORTとSORT-MERGEは同義語です。
6. 1つのプログラムに2つ以上のSAME句を書くことができます。その場合、以下の規則に従わなければなりません。
    a. 同じファイル名を2つ以上のSAME AREA句に書くことはできません。
    b. 同じファイル名を2つ以上のSAME RECORD AREA句に書くことはできません。
    c. SAME AREA句に書いたファイル名を、SAME RECORD AREA句にも書く場合、そのSAME AREA句に書いた他のファイル名も、すべて同じSAME RECORD AREA句に書かなければなりません。しかし、SAME AREA句に書いていないファイル名は、どのSAME RECORD AREA句にも書くことができます。
    d. 同じ整列併合用ファイルのファイル名を、2つ以上のSAME SORT/SORT-MERGE AREA句に書いてはいけません。
    e. 整列併合用ファイルでないファイル名を、SAME AREA句と、SAME SORT/SORT-MERGE AREA句の両方に書いた場合、そのSAME AREA句に書いた他のファイル名もすべて同じSAME SORT/SORT-MERGE AREA句に書かなければなりません。
7. SAME AREA句およびSAME RECORD AREA句に、ファイル記述項にRECORD CONTAINS 0を書いたファイルを書くことはできません。

一般規則

書き方1の規則

SAME AREA句は、この句に書いた2つ以上のファイルが、処理中に同じ記憶領域を共用することを指定します。この句に書いたファイルに割り当てられる記憶領域は、すべて共用されます。この句に書いたファイルは、同時に開いた状態にしてはいけません。

書き方2の規則

SAME RECORD AREA句は、この句に書いた2つ以上のファイルが、現在のレコードを処理するのに同じ記憶領域を共用することを指定します。複数のレコードで共用する領域を「共用領域」といいます。この句に書いた2つ以上のファイルは、同時に開いた状態にすることができます。共用領域中のレコードは、この句に書いたファイルのうち、出力モードで開いたファイルのレコード、および入力モードで開いたファイルの最近読まれたレコードとして用いられます。これは、出力モードで開いたファイルのレコード領域を、入力モードで開いたファイルのレコード領域で再定義することと同じです。両方のレコードは、左端の文字位置にそろえられます。

書き方3の規則

SAME SORT/SORT-MERGE AREA句は、SORT文またはMERGE文に対する記憶領域の割当てを最適に行うことを指定します。この句に書くファイルの少なくとも1つは、整列併合用ファイルでなければなりません。

ただし、このコンパイラでは、システムが記憶領域を割り当てるので、SAME SORT/SORT-MERGE AREA句は、注釈とみなされます。

第5章 データ部

データ部では、実行用プログラムが処理するためのデータを定義します。実行用プログラムが処理するデータには、以下のものがあります。

- 外部記憶装置から入力したり、外部記憶装置や印刷装置に出力したりするファイルのデータ
- 整列併合するためのファイルのデータ
- 報告書を作成するためのファイルのデータ
- 表示装置に表示したり表示装置から入力したりするデータ
- 利用者が定義するデータ

5.1 データ部の構成（DATA DIVISION）

データ部は、環境部より後に書きます。データ部は省略することができます。

データ部は、以下の7つの節からなります。

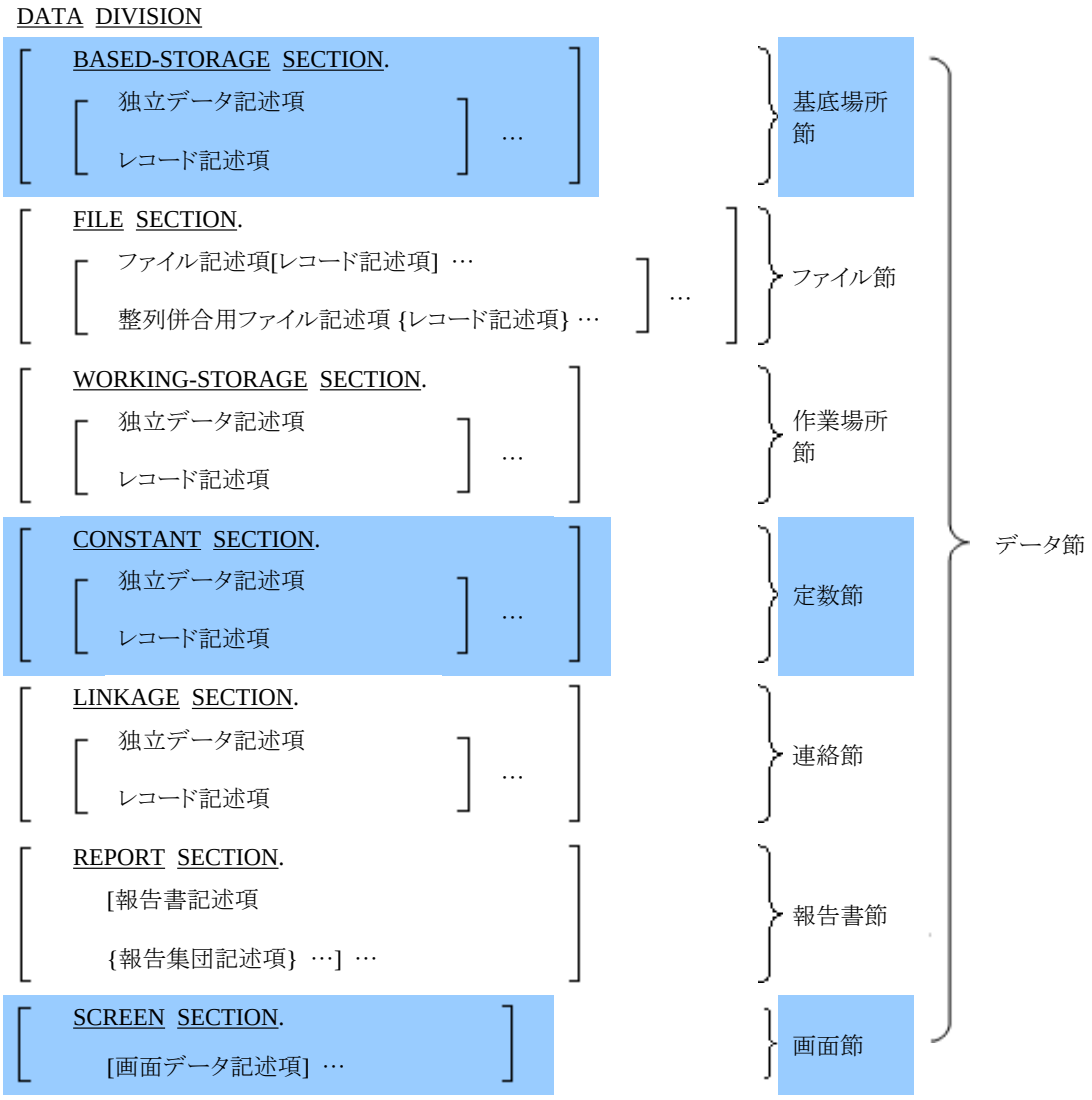
- 基底場所節
- ファイル節
- 作業場所節
- 定数節
- 連絡節
- 報告書節
- 画面節

注意

[WinIPF][LinuxIPF][Linux64][.NET]では、報告書節と画面節は記述できません。
[Winx64]では、報告書節は記述できません。
[Linux]では、画面節は記述できません。

データ部の構成を以下に示します。データ部を構成する節および記述項は、以下に示す順に書かなければなりません。

【書き方】



基底場所節(BASED-STORAGE SECTION)

基底場所節では、他の言語、他のプログラム、または同じプログラムの他の節で確保された領域に存在するデータを操作するためのデータまたはレコードのデータ構造を定義します。

基底場所節に定義したデータ項目は、ポインタデータ項目によってポインタ付けして参照しなければなりません。

基底場所節には、節の見出し(BASED-STORAGE SECTION)の後に、独立データ記述項またはレコード記述項を書きます。

ファイル節(FILE SECTION)

ファイル節では、順ファイル、相対ファイル、索引ファイル、整列併合用ファイルおよび表示ファイルのファイル構造とレコードのデータ構造を定義します。また、報告書ファイルのファイル構造も定義します。

ファイル節には、節の見出し(FILE SECTION)の後に、以下の記述項を書きます。ファイル節で定義したファイルは、環境部のファイル管理記述項で外部媒体と関連付けなければなりません。

- 順ファイル機能、相対ファイル機能、索引ファイル機能または表示ファイル機能を使う場合、1つのファイルに対して、1つのファイル記述項と、それに続く1つ以上のレコード記述項を書きます。レコード記述項は、ファイル記述項の直後に書かなければなりません。

- 整列併合機能を使う場合、1つの整列併合用ファイルに対して、1つの整列併合用ファイル記述項と、それに続く1つ以上のレコード記述項を書きます。レコード記述項は、整列併合用ファイル記述項の直後に書かなければなりません。
- 報告書作成機能を使う場合、1つの報告書ファイルに対して、1つのファイル記述項を書きます。報告書ファイルのファイル節では、レコードのデータ構造は定義しません。報告書ファイルのレコードのデータ構造は、データ部の報告書節で定義します。

作業場所節(WORKING-STORAGE SECTION)

作業場所節では、実行用プログラムで使うデータやレコードのデータ構造を定義します。

作業場所節には、節の見出し(WORKING-STORAGE SECTION)の後に、独立データ記述項またはレコード記述項を書きます。

定数節(CONSTANT SECTION)

定数節では、実行用プログラムで使う定数を定義します。定数節で定義したデータ項目の値は、プログラムの実行中に変化することはありません。

定数節には、節の見出し(CONSTANT SECTION)の後に、独立データ記述項またはレコード記述項を書きます。

連絡節(LINKAGE SECTION)

連絡節では、呼ぶプログラムと呼ばれるプログラムの両方で参照するデータを定義します。連絡節は、呼ばれるプログラムに書きます。以下は、連絡節で定義しなければなりません。

- 手続き部の見出しのUSING指定に書いたデータ項目
- 手続き部のENTRY文のUSING指定に書いたデータ項目

連絡節には、節の見出し(LINKAGE SECTION)の後に、独立データ記述項またはレコード記述項を書きます。

報告書節(REPORT SECTION)

報告書節では、報告書ファイルに書き出す報告書を定義します。報告書作成機能を使う場合、報告書節を書かなければなりません。報告書節で定義した報告書は、データ部のファイル節で、報告書ファイルと対応付けなければなりません。

報告書節には、節の見出し(REPORT SECTION)の後に、報告書記述項と報告集団記述項を書きます。1つの報告書に対して、1つの報告書記述項と、それに続く1つ以上の報告集団記述項を書きます。報告集団記述項は、報告書記述項の直後に書かなければなりません。

画面節(SCREEN SECTION)

画面節では、画面の構造および画面上に表示するデータまたは画面から入力するデータを定義します。スクリーン操作機能を使う場合、画面節を書かなければなりません。

画面節には、節の見出し(SCREEN SECTION)の後に、1つ以上の画面データ記述項を書きます。

独立データ記述項

互いに階層関係を持たず、それ以上細分化する必要のない項目は、独立データ記述項で定義することができます。独立データ記述項は、レベル番号が77のデータ記述項です。独立データ記述項には、以下のすべての句を書かなければなりません。

- レベル番号77
- データ名
- PICTURE句、またはUSAGE IS INDEX句、USAGE IS COMP-1句、USAGE IS COMP-2句およびUSAGE IS POINTER句のいずれかの句

レコード記述項

レコードに関連したデータ記述項のすべての組を、「レコード記述項」といいます。互いに階層関係を持つデータは、レコード記述項にまとめて定義します。1つのレコード記述項では、いくつかの集団項目と基本項目を定義します。他のデータ項目と階層関係をもたないデータを定義する場合、1つのレコード記述項で1つの基本項目だけを定義することもできます。

レコードの構成は、レベル番号で表します。レコードに属するすべてのデータ項目には、01～49のレベル番号を割り当てます。レコードは最も包括的なデータ項目なので、レコードのレベル番号は01から割り当てます。レベル番号01のデータ項目より下位のデータ項目のレベル番号は、連続している必要はありません。

集団項目に従属するデータ項目のデータ記述項は、集団項目のデータ記述項の後に書きます。集団項目は、その集団項目のレベル番号に等しいかより小さいレベル番号を持つデータ記述項が出てくるまでのすべてのデータ項目を包括します。集団項目に直接従属するデータ項目のレベル番号はすべて等しく、その集団項目のレベル番号より大きくなければなりません。

5.2 ファイル記述項

ファイル記述項では、ファイルの物理的な構造を定義します。ファイル記述項は、ファイル節および報告書節に書くことができます。

【書き方1】順ファイル

```
FD ファイル-1
    [BLOCK CONTAINS句]
    [CODE-SET句]
    [CONTROL RECORDS句]
    [DATA RECORDS句]
    [EXTERNAL句]
    [GLOBAL句]
    [LABEL RECORDS句]
    [LINAGE句]
    [RECORD句]
    [VALUE OF句].
```

【書き方2】相対ファイル・索引ファイル

```
FD ファイル名-1
    [BLOCK CONTAINS句]
    [DATA RECORDS句]
    [EXTERNAL句]
    [GLOBAL句]
    [LABEL RECORDS句]
    [RECORD句]
    [VALUE OF句] .
```

【書き方3】表示ファイル

```
FD ファイル名-1
```

[EXTERNAL句]
[GLOBAL句]
[RECORD句] .

【書き方4】報告書ファイル（報告書作成機能で使うファイル）

FD ファイル名-1
[BLOCK CONTAINS句]
[CODE-SET句]
[EXTERNAL句]
[GLOBAL句]
[LABEL RECORDS句]
[RECORD句]
[REPORT句]
[VALUE OF句] .

構文規則

1. ファイル記述項の最初にレベル指示語FDを書き、その後にファイル名-1を書かなければなりません。ファイル名-1の後に書く句の順序は任意です。
2. 報告書ファイル以外の場合、ファイル記述項の後に、1つのレコード記述項を書かなければなりません。報告書ファイルの場合、ファイル記述項の後には、レコード記述項を書くことはできません。
3. 報告書ファイルの場合、ファイル名-1は、順編成のファイルでなければなりません。

一般規則

1. ファイル記述項は、ファイル名-1をファイル結合子に関連付けます。
2. DATA RECORDS句、LABEL RECORDS句およびVALUE OF句は、廃要素です。
3. このコンパイラでは、BLOCK CONTAINS句、CODE-SET句、DATA RECORDS句、LABEL RECORDS句およびVALUE OF句は、注釈とみなされます。

5.2.1 BLOCK CONTAINS句（順ファイル・相対ファイル・索引ファイル・報告書作成）

ブロックの大きさを指定します。

【書き方】

[BLOCK CONTAINS
[整数-1 TO] 整数-2 { RECORDS / CHARACTERS }

一般規則

このコンパイラでは、BLOCK CONTAINS句は注釈とみなされます。

5.2.2 CODE-SET句（順ファイル・報告書作成）

外部記憶媒体上のデータ表現に使う文字符号を指定します。

【書き方】

CODE-SET IS 符号系名-1

構文規則

1. CODE-SET句は、ファイルに関連するレコード記述項に含まれるすべてのデータ項目の用途が表示用であり、かつすべての符号付き数字項目にSIGN IS SEPARATE句を指定した場合にだけ、書くことができます。
2. 符号系名-1は、特殊名段落のALPHABET句で定数に対応付けた符号系名であってはいけません。
3. 印刷ファイルに対して、CODE-SET句を指定することはできません。

一般規則

このコンパイラでは、CODE-SET句は注釈とみなされます。

5.2.3 CONTROL RECORDS句（順ファイル）

ファイルに含まれる制御レコードの名前を指定します。

【書き方】

CONTROL { RECORD IS / RECORDS ARE } {データ名-1} …

構文規則

1. データ名-1は、ファイルに関連するレコード記述項の中の、レベル番号が01のデータ名でなければなりません。
2. データ名-1は、ファイルに関連するレコード記述項の中で一意でなければなりません。
3. データ名-1は、修飾することができます。
4. CONTROL RECORDS句は、FORMAT句付きの印刷ファイルに対してだけ指定することができます。

一般規則

1. データ名-1に指定したレコードをWRITE文で書き出すとき、そのレコードは制御レコードとして扱われます。制御レコードは、システムに対してレコードの編集方法を通知します。
2. 1つのファイルに関連する、制御レコードは、そのファイルに関連する他のレコードとレコード領域を共用します。

5.2.4 DATA RECORDS句（順ファイル・相対ファイル・索引ファイル）

データレコードの名前を列挙します。

DATA RECORDS句は廃要素です。

【書き方】

DATA { RECORD IS / RECORDS ARE } {データ名-1} …

構文規則

データ名-1は、ファイルに関連するレコード記述項の中の、レベル番号が01のデータ名でなければなりません。

一般規則

このコンパイラでは、DATA RECORDS句は注釈とみなされます。

5.2.5 EXTERNAL句（順ファイル・相対ファイル・索引ファイル・表示ファイル・報告書作成）

ファイル結合子に外部属性を与えます。

表示ファイルは、富士通拡張仕様です。

【書き方】

IS EXTERNAL

一般規則

EXTERNAL句は、関連するファイルのファイル結合子が、外部ファイル結合子であることを指定します。実行単位中の外部ファイル結合子は、ファイル名で対応付けられます。同じファイル名の外部ファイル結合子は、同じレコード領域を参照します。

5.2.6 GLOBAL句（順ファイル・相対ファイル・索引ファイル・表示ファイル・報告書作成）

ファイル名が大域名であることを指定します。

表示ファイルは、富士通拡張仕様です。

【書き方】

IS GLOBAL

構文規則

SAME RECORD AREA句に指定したファイルに関連するファイル記述項に、GLOBAL句を書くことはできません。

一般規則

GLOBAL句は、関連するファイル名が大域名であることを指定します。

5.2.7 LABEL RECORDS句（順ファイル・相対ファイル・索引ファイル・報告書作成）

ファイルのラベルの有無を指定します。LABEL RECORDS句は廃要素です。

【書き方】

```
LABEL  { RECORD IS   }  { STANDARD }
       { RECORDS ARE }  { OMITTED  }
```

一般規則

このコンパイラでは、LABEL RECORDS句は注釈とみなされます。

5.2.8 LINAGE句（順ファイル）

論理ページの構成を定義します。

【書き方】

```
LINAGE IS { データ名-1 } LINES
          { 整数-1    }
    [ WITH FOOTING AT { データ名-2 } ]
                      { 整数-2    }
    [ LINES AT TOP    { データ名-3 } ]
                      { 整数-3    }
    [ LINES AT BOTTOM { データ名-4 } ]
                      { 整数-4    }
```

構文規則

1. データ名-1～データ名-4は、符号なし整数項目でなければなりません。
2. データ名-1～データ名-4は、修飾することができます。
3. 整数-2は、整数-1以下でなければなりません。
4. 整数-3および整数-4には、ゼロを指定することもできます。
5. LINAGE句は、FORMAT句なしの印刷ファイルにだけ指定することができます。
6. 論理ページを構成する行の個数の最大値については、“付録B システムの定量制限”を参照してください。

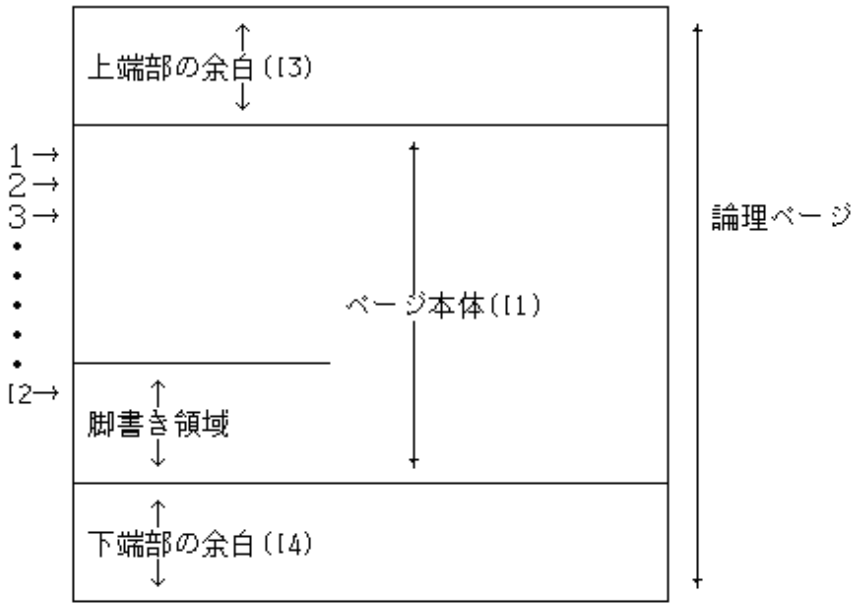
一般規則

1. LINAGE句の指定とそれに対応する論理ページの構成を、以下に示します。

```
*〔LINAGE句の指定〕
    LINAGE IS I1 LINES
        WITH FOOTING AT I2
```

```
        LINES AT TOP I3
        LINES AT BOTTOM I4
```

〔論理ページの構成〕



- 上端部の余白の次の行から始まるI1行分の領域を、「ページ本体」といいます。
- ページ本体の最初の行から数えてI2行目の行から始まり、ページ本体の最後の行までの領域を、「脚書き領域」といいます。

2. 論理ページの大きさは、FOOTING指定以外の各指定に書いた値の合計です。LINES AT TOP指定を省略した場合、“LINES AT TOP 0”を指定したものとみなされます。同様に、LINES AT BOTTOM指定を省略した場合、“LINES AT BOTTOM 0”を指定したものとみなされます。

3. FOOTING指定を省略した場合、ページあふれ条件と独立にページ終了条件が発生することはありません。論理ページの大きさを物理ページの大きさに合わせて定義する必要はありません。

4. 整数-1またはデータ名-1には、論理ページに書いたり行送りしたりできる行数を設定します。この値は、正でなければなりません。

5. 整数-2またはデータ名-2には、ページ本体中の脚書き領域が始まる行の位置を、ページ本体の最初の行を1として設定します。整数-2またはデータ名-2の値は正であり、かつ整数-1またはデータ名-1の値以下でなければなりません。

6. 整数-3またはデータ名-3には、論理ページの上端部の余白の行数を設定します。これらにゼロを設定することもできます。

7. 整数-4またはデータ名-4には、論理ページの下端部の余白の行数を設定します。これらにゼロを設定することもできます。

8. 整数-1、整数-3および整数-4を書いた場合、OPEN OUTPUT文を実行したときに、これらの値に従って、論理ページの各部分を構成する行数が決定されます。整数-2を指定した場合、OPEN OUTPUT文を実行したときに、整数-2の値に従って、脚書き領域が決定されます。これらの整数の値は、プログラムの実行中に、LINAGE句を指定したファイルに書き出されるすべての論理ページに対して適用されます。

9. データ名-1、データ名-3およびデータ名-4を書いた場合、論理ページの各部分を構成する行数が以下のように決定されます。
   - OPEN OUTPUT文を実行したときに、これらのデータ項目の値に従って、最初の論理ページの各部分を構成する行数が決定されます。
   - ADVANCING PAGE指定付きのWRITE文を実行したとき、またはページあふれ条件が発生したときに、これらのデータ項目の値に従って、次の論理ページの各部分を構成する行数が決定されます。

10. データ名-2を書いた場合、OPEN OUTPUT文の実行によって、データ名-2の値に従って、最初の論理ページの脚書き領域が決定されます。また、ADVANCING PAGE指定付きのWRITE文を実行したとき、またはページあふれ条件が発生したときに、このデータ項目の値に従って、次の論理ページの脚書き領域が決定されます。

11. LINAGE句を書くと、特殊レジスタLINAGE-COUNTERが自動的に作成されます。LINAGE-COUNTERには、つねに、現在の論理ページのページ本体中の行位置を指す値が設定されます。LINAGE-COUNTERの規則を、以下に示します。

    ー LINAGE句を指定したファイルごとに、1つのLINAGE-COUNTERが作成されます。

    ー LINAGE-COUNTERの値は、手続き部の文で参照することができます。しかし、LINAGE-COUNTERに値を設定することはできません。LINAGE-COUNTERの値の設定は、入出力管理システムが行います。LINAGE句を指定したファイルを2つ以上定義した場合、LINAGE-COUNTERをファイル名で修飾しなければなりません。

    ー LINAGE-COUNTERの値は、LINAGE句を指定したファイルに対するWRITE文の実行中に、以下の規則に従って自動的に変更されます。

      - ADVANCING PAGE指定付きのWRITE文を実行すると、LINAGE-COUNTERの値は1に戻されます。
      - ADVANCING指定に整数または一意名を指定したWRITE文を実行すると、LINAGE-COUNTERの値は、ADVANCING指定に指定した整数または一意名の値だけ増加します。
      - ADVANCING指定なしのWRITE文を実行すると、LINAGE-COUNTERの値は1だけ増加します。
      - 次の論理ページの最初の行位置に来たときは、LINAGE-COUNTERの値は1に戻されます。

    ー LINAGE句を指定したファイルに対してOPEN OUTPUT文を実行すると、LINAGE-COUNTERの値は1に戻されます。

12. LINAGE句を指定したファイルに関連するファイル結合子が外部ファイル結合子の場合、そのファイル結合子に関連する、実行単位中のすべてのファイル記述項は、以下の規則に従わなければなりません。

    ー そのファイル結合子に関連する、実行単位中のすべてのファイル記述項に、LINAGE句を書かなければなりません。

    ー 整数-1～整数-4を書く場合、対応する各整数にすべて同じ値を指定しなければなりません。

    ー データ名-1～データ名-4を書く場合、それらのデータ名は、外部データ項目でなければなりません。また、対応する各データ名は、すべて同じ名前でなければなりません。

5.2.9 RECORD句（順ファイル・相対ファイル・索引ファイル）

レコードの大きさとレコードの形式を指定します。

【書き方1】固定長レコードの大きさを指定する

```
RECORD CONTAINS 整数-1 CHARACTERS
```

【書き方2】可変長レコードの大きさを指定する

```
RECORD IS VARYING IN SIZE
    [[FROM 整数-2] [TO 整数-3] CHARACTERS]
    [DEPENDING ON データ名-1]
```

【書き方3】レコードの大きさとレコードの形式をレコード記述項で指定する

```
RECORD CONTAINS
    整数-4 TO 整数-5 CHARACTERS
```

構文規則

書き方1の規則

書き方1のRECORD句を指定したファイルに関連するレコード記述項で、文字位置の個数が整数-1を超えるレコードを定義することはできません。

書き方2の規則

1. 書き方2のRECORD句を指定したファイルに関連するレコード記述項で、文字位置の個数が整数-2の値より小さいレコードを定義することはできません。また、文字位置の個数が整数-3の値より大きいレコードを定義することはできません
2. 整数-3は、整数-2より大きくなければなりません。
3. データ名-1は、以下のいずれかで定義した符号なし整数項目でなければなりません。
   - 作業場所節
   - 連絡節
4. 以下のいずれかの場合、DEPENDING ON指定を書くことはできません。
   - ファイルに関連するレコード記述項に、CHARACTER TYPE句またはPRINTING POSITION句を書いた場合
   - ファイルに関連するレコードをFROM指定付きのWRITE文に書き、FROM指定にいずれかのデータ項目を指定した場合
     - CHARACTER TYPE句またはPRINTING POSITION句を指定したデータ項目
     - CHARACTER TYPE句またはPRINTING POSITION句を指定したデータ項目に従属するデータ項目
     - CHARACTER TYPE句またはPRINTING POSITION句を指定したデータ項目を従属するデータ項目

書き方3の規則

整数-5は、整数-4より大きくなければなりません。

一般規則

書き方3およびRECORD句を省略した場合に共通する規則

1. 書き方3のRECORD句を書いた場合、またはRECORD句を省略した場合、データレコードの大きさは、レコード記述項での記述に従って決定されます。
2. ファイルに関連するファイル結合子が外部ファイル結合子の場合、そのファイル結合子に関連する、実行単位中のすべてのレコード記述項は、同じ長さでなければなりません。

書き方1の規則

1. 書き方1のRECORD句は、固定長レコードの大きさを定義するために使います。整数-1には、ファイルに関連するレコードの文字位置の個数を指定します。

   ただし、整数-1に0を指定した場合、データレコードの大きさは、レコード記述項での記述に従って決定されます。
2. 整数-1には、コンパイラがレコード中に挿入する制御用の領域(行制御、印字文字種の指定など)の大きさを加えない値を指定します。
3. ファイルに関連するファイル結合子が外部ファイル結合子の場合、そのファイル結合子に関連する、実行単位中のすべてのファイル記述項では、整数-1に対して同じ値を指定しなければなりません。

書き方2の規則

1. 書き方2のRECORD句は、可変長レコードの大きさを定義するために使います。整数-2には、ファイルに関連するレコードの文字位置の個数の最小値を設定します。整数-3には、ファイルに関連するレコードの文字位置の個数の最大値を指定します。

   ただし、実行時にシステムによって文字位置の個数が指定される場合、その値が文字位置の個数の最大値になります。

2. 整数-2および整数-3には、コンパイラがレコード中に挿入する制御用の領域(行制御、印字文字種の指定など)の大きさを加えない値を指定します。

3. ファイルに関連するファイル結合子が外部ファイル結合子の場合、そのファイル結合子に関連する、実行単位中のすべてのファイル記述項では、整数-2および整数-3に対して同じ値を指定しなければなりません。

4. RECORD句で指定したレコードの大きさがレコード記述項の文字位置の個数より大きい場合、レコードを書き出すとき、レコード記述項の文字位置の個数を超える部分には以下の値が設定されます。

   - レコード記述項にCHARACTER TYPE句またはPRINTING POSITION句を書いた場合、レコード記述項の文字位置の個数を超える部分には空白が設定されます。
   - レコード記述項にCHARACTER TYPE句およびPRINTING POSITION句のどちらも書かなかった場合、レコード記述項の文字位置の個数を超える部分の内容は規定されません。

5. 「レコード記述項の大きさ」とは、REDEFINES句またはRENAMES句を指定したデータ項目を除く、すべての基本項目の文字位置の個数の合計に、桁づめのための無名項目の文字位置の個数を加えた値です。レコード記述項にDEPENDING ON指定付きのOCCURS句を書いた場合は、以下の規則に従って、レコード記述項の大きさの最小値と最大値が決定されます。

   - レコード記述項の大きさの最小値は、表要素の最小の個数を用いて決定されます。
   - レコード記述項の大きさの最大値は、表要素の最大の個数を用いて決定されます。

6. FROM指定を省略した場合、整数-2にレコード記述項の大きさの最小値を指定したものとみなされます。

7. TO指定を省略した場合、整数-3にレコード記述項の大きさの最大値を指定したものとみなされます。

8. DEPENDING ON指定を書いた場合、データ名-1の値は以下のように使われます。

   - RELEASE文、REWRITE文またはWRITE文を実行する前に、書き出すレコードの文字位置の個数を、データ名-1に設定しなければなりません。これらの文を実行するとき、書き出すレコードの文字位置の個数は、データ名-1の値によって決定されます。
   - READ文またはRETURN文の実行が成功すると、読み込まれたレコードの文字位置の個数がデータ名-1に設定されます。
   - READ文にINTO指定を書いた場合、暗黙のMOVE文の送出し側データ項目の文字位置の個数として、データ名-1に設定された値が使われます。
   - DELETE文、RELEASE文、REWRITE文、START文またはWRITE文を実行しても、データ名-1の値は変更されません。また、READ文またはRETURN文の実行が不成功だった場合も、データ名-1の値は変更されません。

9. DEPENDING ON指定を省略した場合、RELEASE文、REWRITE文またはWRITE文を実行するとき、書き出すレコードの文字位置の個数は、以下の規則に従って決定されます。

   - そのレコードが可変反復データ項目を含まない場合、そのレコードの文字位置の個数によって決定されます。
   - そのレコードが可変反復データ項目を含む場合、実行時の可変反復データ項目の反復回数(OCCURS句のDEPENDING ON指定のデータ名の値)から算出される表の大きさと、可変反復データ項目以外のデータ項目の大きさとの合計によって決定されます。

10. DEPENDING ON指定を省略し、READ文にINTO指定を書いた場合、暗黙のMOVE文の送出し側データ項目の文字位置の個数として、DEPENDING ON指定を書いたときにデータ名-1に設定されるべき値が使われます。

5.2.10 RECORD句（表示ファイル）

レコードの大きさとレコードの形式を指定します。

【書き方1】可変長レコードの大きさを指定する

```
RECORD IS VARYING IN SIZE
    [[FROM 整数-2] [TO 整数-3] CHARACTERS]
    [DEPENDING ON データ名-1]
```

【書き方2】レコードの大きさとレコード形式をレコード記述項で指定する

```
RECORD CONTAINS
    整数-4 TO 整数-5 CHARACTERS
```

書き方1と書き方2は、それぞれ順ファイルのRECORD句の書き方2と書き方3と同じです。

“5.2.9 RECORD句（順ファイル・相対ファイル・索引ファイル）”を参照してください。

5.2.11 RECORD句（報告書作成）

レコードの大きさとレコードの形式を指定します。

【書き方1】固定長レコードの大きさを指定する

```
RECORD CONTAINS 整数-1 CHARACTERS
```

【書き方2】可変長レコードの大きさを指定する

```
RECORD CONTAINS 整数-4 TO 整数-5 CHARACTERS
```

書き方1と書き方2は、それぞれ順ファイルのRECORD句の書き方1と書き方3と同じです。

“5.2.9 RECORD句（順ファイル・相対ファイル・索引ファイル）”を参照してください。

5.2.12 REPORT句（報告書作成）

報告書の名前を報告書ファイルに関連付けます。

【書き方】

```
{ REPORT IS  }
{ REPORTS ARE } {報告書名-1} …
```

構文規則

1. 報告書名-1は、同じプログラムの報告書節の報告書記述項に書いた報告書名でなければなりません。報告書名-1の並びを書く順序は、任意です。

2. 報告書記述項に書いた1つの報告書名は、1つのREPORT句にだけ書くことができます。
3. REPORT句を指定したファイルを「報告書ファイル」といいます。報告書ファイルは、USE文、CLOSE文、OUTPUT指定のOPEN文、またはEXTEND指定のOPEN文にだけ書くことができます。

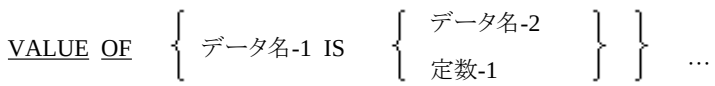
一般規則

1. 1つの報告書ファイルに2種類以上の報告書を関連付ける場合、報告書名-1を2つ以上書きます。
2. 1つの報告書ファイルに対するINITIATE文を実行してからTERMINATE文を実行するまでの間に、その報告書ファイルを参照する入出力文を実行することはできません。
3. ファイルに関連するファイル結合子が外部ファイル結合子の場合、そのファイル結合子に関連する、実行単位中のすべてのファイル記述項では、そのファイルを報告書ファイルとして定義しなければなりません。

5.2.13 VALUE OF句（順ファイル・相対ファイル・索引ファイル・報告書作成）

ラベルレコード中のデータ項目の値を定義します。VALUE OF句は廃要素です。

【書き方】



構文規則

1. データ名-2は、必要ならば修飾しなければなりません。データ名-2は、指標データ項目であってはいけません。
2. データ名-2は、作業場所節で定義したデータ項目でなければなりません。

一般規則

このコンパイラでは、VALUE OF句は注釈とみなされます。

5.3 整列併合用ファイル記述項

整列併合用ファイル記述項では、整列併合用ファイルの物理的な構造を定義します。整列併合用ファイル記述項は、ファイル節に書くことができます。

【書き方】

```
SD ファイル名-1
   [DATA RECORDS句]
   [RECORD句].
```

構文規則

1. 整列併合用ファイル記述項の最初にレベル指示語SDを書き、その後にファイル名-1を書かなければなりません。ファイル名-1の後に書く句の順序は任意です。
2. 整列併合用ファイル記述項の後に、少なくとも1つのレコード記述項を書かなければなりません。しかし、整列併合用ファイルに対して、入出力文を実行することはできません。

一般規則

1. 整列併合用ファイルのDATA RECORDS句の規則は、順ファイルの場合と同じです。“5.2.4 DATA RECORDS句（順ファイル・相対ファイル・索引ファイル）”を参照してください。
2. 整列併合用ファイルのRECORD句の規則は、順ファイルの場合と同じです。“5.2.9 RECORD句（順ファイル・相対ファイル・索引ファイル）”を参照してください。

5.4 データ記述項

データ記述項では、データ項目を定義します。

データ記述項は、以下に書くことができます。

- 基底場所節
- ファイル節
- 作業場所節
- 定数節
- 連絡節

データ記述項は、レコード記述項および独立データ記述項を構成する記述項です。

データ記述項には、データ名を定義する場合の書き方、データ名を再命名する場合の書き方、および条件名を定義する場合の書き方の3つの書き方があります。

【書き方1】データ名を定義する

```
レベル番号 [ データ名-1 ]
           [ FILLER   ]
  [BASED ON句]
  [BLANK WHEN ZERO句]
  [CHARACTER TYPE句]
  [EXTERNAL句]
  [GLOBAL句]
  [JUSTIFIED句]
  [OCCURS句]
  [PICTURE句]
  [PRINTING POSITION句]
  [REDEFINES句]
  [SIGN句]
  [SYNCHRONIZED句]
  [TYPE句]
  [TYPEDEF句]
  [USAGE句]
  [VALUE句].
```

【書き方2】データ名を再命名する

```
66 データ名-1
```

RENAMES句.

【書き方3】条件名を定義する

88 条件名-1
VALUE句.

構文規則

書き方1の規則

1. レベル番号は、01～49のいずれか、または77でなければなりません。
2. 書き方1のデータ記述項の先頭には、レベル番号を書かなければなりません。データ名またはFILLERを書く場合、それらはレベル番号の直後に書かなければなりません。REDEFINES句を書く場合、データ名またはFILLERを書くときはそれらの直後に書き、データ名またはFILLERを省略するときはレベル番号の直後に書かなければなりません。

   TYPEDEF句を書く場合、データ名を指定し、その直後にTYPEDEF句を書かなければなりません。

   その他の句は、任意の順序で書くことができます。
3. 指標データ項目、内部浮動小数点項目、int型2進整数データ項目およびポインタデータ項目以外の基本項目を定義する場合、PICTURE句を書かなければなりません。

   以下を定義する場合は、PICTURE句を書くことはできません。

   - 指標データ項目
   - int型2進整数データ項目
   - 内部浮動小数点項目
   - ポインタデータ項目
4. 集団項目を定義する場合、BLANK WHEN ZERO句、JUSTIFIED句、PICTURE句およびSYNCHRONIZED句を書くことはできません。ただし、[Win32][WinIPF][Winx64][Solaris][LinuxIPF][Linux64][.NET] では、SYNCHRONIZED句を集団項目に指定することもできます。
5. EXTERNAL句は、作業場所節のレベル番号が01のデータ記述項にだけ、書くことができます。
6. EXTERNAL句とREDEFINES句は、同じデータ記述項に書くことはできません。
7. GLOBAL句は、レベル番号が01のデータ記述項にだけ書くことができます。
8. GLOBAL句またはEXTERNAL句を書く場合、データ名-1を書かなければなりません。
9. EXTERNAL句またはGLOBAL句を指定したファイル記述項に関連するレコード記述項を書く場合、データ名-1を書かなければなりません。
10. TYPEDEF句は、レベル番号が01のデータ記述項にだけ書くことができます。また、TYPEDEF句をREDEFINES句、EXTERNAL句と同時に指定することはできません。
11. TYPE句は、EXTERNAL句、GLOBAL句、OCCURS句、BASED ON句、TYPEDEF句、VALUE句を除く句を指定したデータ記述項に指定してはなりません。

書き方2の規則

書き方2のデータ記述項の先頭には、レベル番号66を書かなければなりません。レベル番号の後に書く句の順序は、データ名-1、RENAMES句の順でなければなりません。

書き方3の規則

書き方3のデータ記述項の先頭には、レベル番号88を書かなければなりません。レベル番号の後に書く句の順序は、条件名-1、VALUE句の順でなければなりません。

一般規則

書き方1の規則

1. データ名-1には、データ項目の名前を指定します。明に参照しないデータ項目を定義する場合は、FILLERを用います。
2. データ名-1およびFILLERを省略した場合、FILLERを指定したものとみなされます。この項目を「無名項目」といいます。
3. データ記述項でFILLERを用いて定義した項目を、「FILLER項目」といいます。FILLER項目は、直接参照することはできません。
4. レベル指示語FDまたはSDの下位に、レベル番号01を持つ記述項が2つ以上従属している場合には、これらが同じ記憶領域を暗に再定義します。

書き方3の規則

1. 書き方3のデータ記述項を「条件名記述項」といいます。条件名-1に条件の名前を指定し、VALUE句に条件名-1の値または値の範囲を指定します。
2. 条件名記述項は、条件変数のデータ記述項の直後に書かなければなりません。条件変数は、以下のデータ項目であってはいけません。
   - 条件名
   - レベル番号66のデータ項目
   - JUSTIFIED 句、SYNCHRONIZED句、またはUSAGE IS DISPLAY以外のUSAGE句を指定したデータ項目を従属する集団項目
   - 指標データ項目
   - 集団項目として定義された型、または条件名を含む型を参照するデータ記述項

基底場所節のデータ記述項での注意

1. 基底場所節では、条件名記述項(レベル番号が88)以外のデータ記述項で、VALUE句を書いてはいけません。基底場所節で定義したデータ項目の初期値は規定されません。
2. 基底場所節のデータ記述項では、EXTERNAL句、CHARACTER TYPE句およびPRINTING POSITION句を書いてはいけません。
3. 基底場所節で定義したデータ項目は、書き方の中で“一意名”が現れるところすべてに書くことができます。
4. 基底場所節で定義したデータ項目は、以下の箇所の書き方の中で、“データ名”が現れるところに書くことができます。ただし、これらのデータ名を明にポインタ付けしてはいけません。
   - REDEFINES句のデータ名
   - RENAMES句のデータ名
   - RENAMES句のTHROUGH指定のデータ名
   - OCCURS句のKEY指定のデータ名
   - SEARCH ALL文のWHEN指定のデータ名
5. SEARCH ALL文のWHEN指定のデータ名に基底場所節で定義したデータ項目を指定する場合、SEARCH ALL文の一意名は、暗にポインタ付けしなければなりません。
6. 基底場所節では、STRONG指定のある型宣言を参照できません。

ファイル節のレコード記述項での注意

ファイル節では、条件名記述項(レベル番号が88)以外のデータ記述項で、VALUE句を書いてはいけません。ファイル節で定義したデータ項目の初期値は規定されません。

作業場所節のデータ記述項での注意

作業場所節で定義する指標データ項目以外のデータ項目には、初期値を与えることができます。初期値を与える場合、VALUE句を書きます。VALUE句を省略した場合、データ項目の初期値は規定されません。

定数節のデータ記述項での注意

1. 定数節で定義するデータ項目には、VALUE句を書いて、初期値を与えなければなりません。
2. 定数節で定義したデータ項目は、手続き部の受取り側項目を除いて、作業場所節中のデータ項目が書けるところにはどこでも書くことができます。

連絡節のデータ記述項での注意

連絡節では、条件名記述項(レベル番号が88)以外のデータ記述項で、VALUE句を書いてはいけません。連絡節で定義したデータ項目の初期値は規定されません。

5.4.1 BLANK WHEN ZERO句

データ項目の値がゼロのとき、値を空白に置き換えます。

使い方の例については、“付録H サンプル集”の“H.23 BLANK WHEN ZERO句”を参照してください。

【書き方】

BLANK WHEN { ZERO / ZEROS / ZEROES }

構文規則

1. BLANK WHEN ZERO句は、基本項目にだけ指定することができます。
2. BLANK WHEN ZERO句は、数字項目または数字編集項目にだけ指定することができます。
3. 数字項目にBLANK WHEN ZERO句を指定する場合、数字項目は符号なしの外部10進項目でなければなりません。
4. ZERO、ZEROSおよびZEROESは同義語です。

一般規則

1. BLANK WHEN ZERO句は、データ項目の値がゼロのとき、その値を空白に置き換えることを指定します。
2. 数字項目にBLANK WHEN ZERO句を指定した場合、その数字項目の項類は数字編集であるとみなされます。

5.4.2 CHARACTER TYPE句

印字するときの文字の形式を指定します。

【書き方1】日本語項目または日本語編集項目の文字サイズを指定する

[CHARACTER TYPE IS] { MODE-1 / MODE-2 / MODE-3 } [BY 呼び名-1]

【書き方2】日本語項目または日本語編集項目の文字サイズ、文字ピッチ、文字書体、文字回転量および文字形態を指定する

CHARACTER TYPE IS 呼び名-2

【書き方3】任意のデータ項目の文字サイズ、文字ピッチ、文字書体、文字回転量および文字形態を指定する

CHARACTER TYPE IS
{ 印字モード名-1 / [印字モード名-2] … DEPENDING ON データ名-1 }
[OR
{ 印字モード名-3 / [印字モード名-4] … DEPENDING ON データ名-2 }]

構文規則

書き方1～書き方3に共通する規則

1. CHARACTER TYPE句は、どのレベル番号のデータ記述項にも書くことができます。
2. レベル番号が01または77以外のREDEFINES句を指定したデータ項目、およびそのようなデータ項目に従属するデータ項目に、CHARACTER TYPE句を指定することはできません。
3. TYPEDEF句を指定したデータ項目、およびそのようなデータ項目に従属するデータ項目に、CHARACTER TYPE句を指定することはできません。
4. CHARACTER TYPE句を指定したデータ項目を含むレコード記述項の中に、DEPENDING ON指定付きのOCCURS句を書くことはできません。

書き方1および書き方2に共通する規則

1. CHARACTER TYPE句は、日本語項目、日本語編集項目、日本語項目を従属する集団項目、および日本語編集項目を従属する集団項目にだけ指定することができます。
   ただし、日本語項目または日本語編集項目を従属する集団項目に指定する場合、日本語項目または日本語編集項目は再定義する項目に従属するものであってはなりません。
2. CHARACTER TYPE句を集団項目に指定した場合、CHARACTER TYPE句は、集団項目に従属するすべての日本語項目および日本語編集項目に適用されます。
3. 集団項目および集団項目に従属するデータ項目の両方にCHARACTER TYPE句を指定する場合、同じ意味を持つCHARACTER TYPE句を指定しなければなりません。

書き方1の規則

1. 呼び名-1は、環境部の特殊名段落で、以下の機能名に対応付けなければなりません。
   - HSC、F0202、H0202、F0102、F0201
2. 呼び名-1を以下の機能名に対応付けた場合、MODE-1またはMODE-2だけを指定することができます。
   - HSC、H0202

書き方2の規則

呼び名-2は、環境部の特殊名段落で、以下の文字列で始まる機能名に対応付けなければなりません。

- GTA、GTB、GTC、GTD、GTX、GYA、GYB、GYC、GYD、GYX
- TA、TB、TC、TD、TX、YA、YB、YC、YD、YX

書き方3の規則

1. 書き方3のCHARACTER TYPE句を指定したデータ項目に従属するデータ項目、および書き方3のCHARACTER TYPE句を指定したデータ項目を従属するデータ項目に、書き方1または書き方2のCHARACTER TYPE句を指定することはできません。
2. 印字モード名-1～印字モード名-4は、特殊名段落のPRINTING MODE句で定義しなければなりません。
3. OR指定付きのCHARACTER TYPE句は、集団項目にだけ指定することができます。
4. CHARACTER TYPE句を基本項目に指定する場合、以下の規則に従わなければなりません。
   - 基本項目の用途は表示用でなければなりません。
   - 基本項目が日本語項目または日本語編集項目の場合、印字モード名-1および印字モード名-2は、特殊名段落で、FOR SOCSを指定したPRINTING MODE句に対応付けてはいけません。
   - 基本項目が英字項目、英数字項目、英数字編集項目、外部10進項目、外部ブール項目、外部浮動小数点項目または数字編集項目の場合、印字モード名-1および印字モード名-2は、特殊名段落で、FOR MOCSを指定したPRINTING MODE句に対応付けてはいけません。
5. CHARACTER TYPE句を集団項目に指定した場合、CHARACTER TYPE句は、その集団項目に従属する、用途が表示用のデータ項目すべてに適用されます。ただし、集団項目に従属するデータ項目のうち、書き方3のCHARACTER TYPE句を指定したデータ項目およびそのデータ項目に従属する項目には、適用されません。
6. 1つのCHARACTER TYPE句に印字モード名-2を2つ以上繰り返し指定する場合、それらに対する特殊名段落のPRINTING MODE句のFOR指定は、すべて同じでなければなりません。
7. OR指定を書く場合、以下の規則に従わなければなりません。
   - 印字モード名-1～印字モード名-4は、特殊名段落で、FOR ALLを指定したPRINTING MODE句に対応付けてはいけません。
   - 1つのCHARACTER TYPE句に印字モード名-4を2つ以上繰り返し指定する場合、それらに対する特殊名段落のPRINTING MODE句のFOR指定は、すべて同じでなければなりません。
   - 印字モード名-3または印字モード名-4に対する特殊名段落のPRINTING MODE句のFOR指定は、印字モード名-1または印字モード名-2に対する特殊名段落のPRINTING MODE句のFOR指定と同じであってはいけません。
8. データ名-1およびデータ名-2は、修飾することができます。
9. データ名-1およびデータ名-2は、整数項目でなければなりません。
10. データ名-1およびデータ名-2は、以下のいずれかで定義したデータ項目でなければなりません。
    - 作業場所節
    - ファイル節
    - 定数節
    - 連絡節
11. データ名-1およびデータ名-2は、レコード中の可変位置にあってはいけません。
12. EXTERNAL句を指定したレコード記述項に含まれるデータ記述項にCHARACTER TYPE句を書く場合、データ名-1およびデータ名-2は外部属性を持つデータ項目でなければなりません。また、そのデータ記述項と、データ名-1およびデータ名-2のデータ記述項は、同じデータ部に書かなければなりません。

13. GLOBAL句を指定したレコード記述項に含まれるデータ記述項にCHARACTER TYPE句を書く場合、データ名-1およびデータ名-2は大域名でなければなりません。また、そのデータ記述項と、データ名-1およびデータ名-2のデータ記述項は、同じデータ部に書かなければなりません。

一般規則

書き方1の規則

1. 書き方1のCHARACTER TYPE句は、日本語項目または日本語編集項目を印字するときの文字サイズを指定します。
2. MODE-1、MODE-2またはMODE-3は印字する文字サイズを意味し、それぞれ12ポイント、9ポイントおよび7ポイントに対応します。

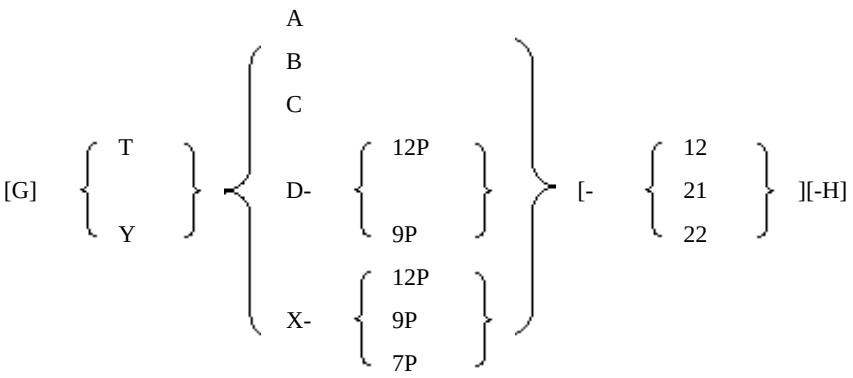
   呼び名-1に対応付ける機能名は、以下の意味を持ちます。

呼び名-1に対応付ける機能名	意味
HSC	半角文字
F0202	倍角文字
H0202	半角の倍角文字
F0102	長体文字
F0201	平体文字

書き方2の規則

1. 書き方2のCHARACTER TYPE句は、日本語項目または日本語編集項目を印字するときの、文字サイズ、文字ピッチ、文字書体、文字印字方向および文字形態を指定します。
2. 呼び名-2に対応付ける機能名は、以下の形式と意味を持ちます。

   形式



   意味

ゴシック体文字の指定	G	ゴシック体文字
文字印字方向の指定	T	縦書き（文字を反時計回りに90度回転して印字）
	Y	横書き（通常の印字）
文字サイズおよび文字ピッチの指定	A	9ポイントの文字を2ピッチで印字
	B	9ポイントの文字を1.5ピッチで印字
	C	9ポイントの文字を7.5CPIで印字
	D-12P	12ポイントの文字を6CPIで印字
	D-9P	9ポイントの文字を6CPIで印字
	X-12P	12ポイントの文字を文字ピッチで印字

	X-9P	9ポイントの文字を文字ピッチで印字
	X-7P	7ポイントの文字を文字ピッチで印字
文字形態の指定	12	長体文字
	21	平体文字
	22	倍角文字
	H	半角文字

- 12P、9Pおよび7Pは、それぞれ書き方1のCHARACTER TYPE句のMODE-1、MODE-2およびMODE-3と同義です。
- 12、21および22は、それぞれ機能名F0101、F0201およびF0202と同義です。
- Hは、機能名HSCと同義です。また、22-Hは、機能名H0202と同義です。

書き方3の規則

1. 書き方3のCHARACTER TYPE句は、任意のデータ項目を印字するときの文字サイズ、文字ピッチ、文字書体、文字回転量および文字形態を指定します。
2. DEPENDING ON指定を指定した場合、データ名-1の値をnとすると、印字モード名-2の並びのうちの第n番目の印字モード名-2が有効になります。同様に、データ名-2の値をnとすると、印字モード名-4の並びのうちの第n番目の印字モード名-2が有効になります。
3. 印字モード名-2の並びの数をmとすると、データ名-1の値は、1～mでなければなりません。同様に、印字モード名-4の並びの数をmとすると、データ名-2の値は、1～mでなければなりません。
4. CHARACTER TYPE句を指定したデータ項目を使用するファイルに関連付けられたファイル結合子が外部ファイル結合子の場合、そのファイル結合子を参照する実行単位中のすべてのプログラムは、書き方3のCHARACTER TYPE句を指定したデータ項目を含んでいなければなりません。

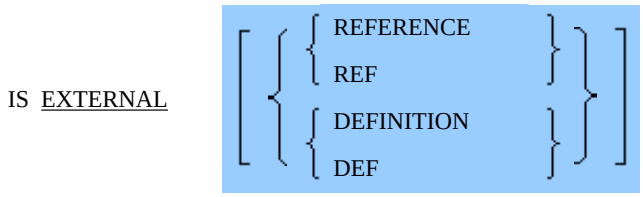
書き方1～書き方3に共通する規則

1. CHARACTER TYPE句は、WRITE文でレコードを印字するときに意味を持ちます。
2. 文字サイズ、文字ピッチ、文字書体、文字回転量および文字形態については、“4.2.3.11 PRINTING MODE句”を参照してください。

5.4.3 EXTERNAL句

レコードに外部属性を与えます。

【書き方】



注意

REFERENCE、REF、DEFINITIONおよびDEF指定は[DS][Solaris][LinuxIPF][Linux64]固有機能です。

構文規則

1. EXTERNAL句は、作業場所節のレベル番号が01のデータ記述項にだけ、書くことができます。
2. EXTERNAL句を指定したレベル番号01のデータ名は、同じプログラム中の、別のEXTERNAL句を指定したレベル番号01のデータ名と同じであってはいけません。
3. EXTERNAL句を書いたデータ記述項、またはそのデータ記述項に従属するデータ記述項に、VALUE句を書くことはできません。EXTERNAL句を書いたデータ記述項、またはそのデータ記述項に従属するデータ記述項に関連する条件名記述項には、VALUE句を書くことができます。
4. REFERENCE、REF、DEFINITIONおよびDEFの各指定は、作業場所節のEXTERNAL句を含むレコード記述項にだけ指定できます。
5. REFERENCEとREFおよびDEFINITIONとDEFはそれぞれ同義語であり、どちらも書くことができます。
6. [Linux]では、EXTERNAL句はint型2進整数データ項目には指定できません。

一般規則

1. EXTERNAL句は、レコードに外部属性を与えます。EXTERNAL句を指定したデータ項目、およびそのデータ項目に従属するすべてのデータ項目に、外部属性が与えられます。
2. レコード記述項にEXTERNAL句を指定したレコードを、「外部データレコード」といいます。実行単位中の外部データレコードは、レコード名で対応付けられます。同じレコード名の外部データレコードは、同じ記憶領域を参照します。同じレコード名の外部データレコードの文字位置の個数は、同じでなければなりません。
3. 外部データレコードをREDEFINES句の右辺に書くことができます。外部データレコードは、実行単位中の任意のプログラムで任意に再定義することができます。
4. EXTERNAL句とGLOBAL句の両方を書いたデータ記述項の名前は、そのデータ記述項を書いたプログラムに含まれるプログラム中で、外部属性を持つレコードの名前として用いることはできません。
5. EXTERNAL REFERENCEは、同じ実行単位内に存在する他プログラムに同じデータ名で記述された、以下のいずれかのデータ記述項が定義するレコードを参照し、処理することを可能とします。
   - C言語プログラムのextern指定のない外部変数
   - DEFINITION指定のEXTERNAL句を含むデータ記述項
6. EXTERNAL DEFINITIONは、同じ実行単位内に存在する他プログラムに同じデータ名で記述された、以下のデータ記述項から参照および処理できるレコードを定義します。
   - C言語プログラムのextern指定された外部変数
   - REFERENCE指定のEXTERNAL句を含むデータ記述項
7. 1つの実行単位を構成するプログラムにおいて、DEFINITION指定のEXTERNAL句を含む記述項の左辺として指定されたデータ名は、他のDEFINITION指定のEXTERNAL句を含む記述項の左辺として指定されたデータ名、およびREFERENCE、REF、DEFINITIONおよびDEFの指定のないEXTERNAL句を含む記述項の左辺として指定されたデータと同じであってはいけません。

5.4.4 GLOBAL句

データ名が大域名であることを指定します。

使い方の例については、“付録H サンプル集”の“H.24 GLOBAL句”を参照してください。

【書き方】

IS GLOBAL

構文規則

1. GLOBAL句は、以下の節の、レベル番号が01のデータ記述項にだけ書くことができます。
   - ファイル節
   - 作業場所節
   - 連絡節
   - 定数節
2. 同じデータ部では、同じデータ名を指定した2つのデータ記述項に、GLOBAL句を書くことはできません。
3. SAME RECORD AREA句に指定したファイルに関連するレコード記述項に、GLOBAL句を書くことはできません。

一般規則

1. GLOBAL句は、データ名が大域名であることを指定します。GLOBAL句を指定したデータ項目、およびそのデータ項目に従属するデータ項目の名前が、大域名になります。
2. 大域名は、大域名を定義したプログラムに直接または間接に含まれるプログラムの中で、再度定義することなく参照することができます。
3. REDEFINES句を指定したデータ項目にGLOBAL句を指定した場合、REDEFINES句の左辺のデータ名だけが大域名になります。

5.4.5 JUSTIFIED句

受取り側データ項目の右端に合わせて転記することを指定します。

使い方の例については、“付録H サンプル集”の“H.25 JUSTIFIED句”を参照してください。

【書き方】

{ JUSTIFIED / JUST } RIGHT

構文規則

1. JUSTIFIED句は、基本項目にだけ指定することができます。
2. JUSTIFIED句は、以下にだけ指定することができます。
   - 英字項目
   - 英数字項目
   - 日本語項目
3. JUSTIFIEDとJUSTは同義語です。

一般規則

1. JUSTIFIED句は、受取り側データ項目の右端に合わせて転記することを指定します。
2. 転記において、受取り側データ項目にJUSTIFIED句を指定した場合、送出し側のデータが受取り側データ項目の右端に合わせて格納されます。送出し側データ項目と受取り側データ項目の大きさが異なる場合、以下の規則に従って転記されます。
   - 送出し側データ項目の大きさが受取り側データ項目の大きさより大きい場合、送出し側データ項目の左端の文字が切り捨てられます。

ー 送出し側データ項目の大きさが受取り側データ項目の大きさより小さい場合、送出し側データ項目の左端の余りの部分に空白が埋められます。

3. JUSTIFIED句を省略した場合、データを基本項目に格納するときの標準桁よせの規則が適用されます。

5.4.6 OCCURS句

同じデータ構造の繰返し回数を指定します。

使い方の例については、“付録H サンプル集”の“H.26 OCCURS句”を参照してください。

【書き方1】一定の繰返し回数を指定する

```
OCCURS 整数-2 TIMES
    [ { ASCENDING  } KEY IS {データ名-2} …  ] …
      { DESCENDING }
  [INDEXED BY {指標名-1} … ]
```

【書き方2】可変の繰返し回数を指定する

```
OCCURS [ 整数-1 ] TO 整数-2 TIMES
         DEPENDING ON データ名-1
    [ { ASCENDING  } KEY IS {データ名-2} …  ] …
      { DESCENDING }
    [INDEXED BY {指標名-1} … ]
```

構文規則

書き方1および書き方2に共通する規則

1. OCCURS句は、レベル番号が01、66、77または88のデータ記述項に指定することはできません。
2. OCCURS句を指定したデータ項目に従属するデータ項目に、OCCURS DEPENDING ON句を指定することはできません。

   ただし、このコンパイラでは、OCCURS句を指定したデータ項目に従属するデータ項目に、OCCURS DEPENDING ON句を指定することができます。
3. データ名-1およびデータ名-2は、修飾することができます。
4. KEY IS指定を書く場合、データ名-2は以下の規則に従わなければなりません。
   - ー データ名-2は、OCCURS句を書いたデータ記述項自身のデータ名、またはそのデータ記述項に従属するデータ記述項のデータ名でなければなりません。
   - ー OCCURS句を書いたデータ記述項自身のデータ名をデータ名-2に指定する場合、データ名-2を2つ以上書くことはできません。
5. データ名-2は、添字付けをしてはいけません。
6. データ名-2がOCCURS句を指定したデータ記述項およびそれに従属するデータ記述項の範囲内で一意でない場合、データ名-2を修飾しなければなりません。
7. 整数-2の最大値については、“付録B システムの定量制限”を参照してください。

8. OCCURS句を書いたデータ記述項自身のデータ名をデータ名-2に指定する場合を除き、データ名-2のデータ記述項にOCCURS句を書くことはできません。
9. KEY IS指定を書いたデータ記述項で、そのデータ記述項に従属するデータ記述項のデータ名をデータ名-2に指定した場合、KEY IS指定を書いたデータ記述項とデータ名-2のデータ記述項の間に、別のKEY IS指定付きのOCCURS句を持つデータ記述項を書くことはできません。
10. データ名-2のデータ項目の長さは、256バイト以下でなければなりません。
11. データ名-2は、ブール項目または指標データ項目であってはいけません。
12. データ名-2の並びの個数の最大については、“付録B システムの定量制限”を参照してください。
13. 指標名-1を指定したデータ項目、および指標名-1を指定したデータ項目に従属するデータ項目は、添字として指標名-1を付けて参照することができます。
14. 指標名の記憶領域は、コンパイラにより自動的に割り付けられます。指標名はデータではなく、データの階層にも属しません。
15. 指標名-1の並びの個数の最大については、“付録B システムの定量制限”を参照してください。
16. EXTERNAL句が有効なデータ記述項に、INDEXED BY指定付きのOCCURS句を書くことはできません。

書き方2の規則

1. 整数-1は0以上でなければなりません。整数-2は整数-1より大きくなければなりません。

   整数-1が省略された場合、0が指定されたとみなします。
2. データ名-1は、整数項目でなければなりません。
3. データ名-1は、OCCURS句を書いたデータ記述項の最初の文字位置から、そのデータ記述項を含むレコード記述項の最後の文字位置までの範囲内で定義することはできません。
4. レコード記述項の中で、書き方2のOCCURS句を指定したデータ記述項の後には、そのデータ記述項に従属するデータ記述項しか書くことができません。

   ただし、このコンパイラでは、書き方2のOCCURS句を書いたデータ記述項の後に、そのデータ記述項に従属しないデータ記述項を書くことができます。
5. 定数節のデータ項目に、書き方2のOCCURS句を指定することはできません。
6. EXTERNAL句を指定したレコード記述項に含まれるデータ記述項に、KEY IS指定付きのOCCURS句を書く場合、データ名-1は外部属性を持つデータ項目でなければなりません。また、そのデータ記述項とデータ名-1のデータ記述項は、同じデータ部に書かなければなりません。
7. GLOBAL句を指定したレコード記述項に含まれるデータ記述項に、KEY IS指定付きのOCCURS句を書く場合、データ名-1は大域名でなければなりません。また、そのデータ記述項とデータ名-1のデータ記述項は、同じデータ部に書かなければなりません。

一般規則

書き方1および書き方2に共通する規則

1. OCCURS句を指定したデータ記述項に書いた句は、反復されたそれぞれのデータ項目にも適用されます。ただし、OCCURS句自身は、反復しては適用されません。
2. OCCURS句を指定した集団項目が、SYNCHRONIZED句を指定した2進項目を従属する場合、その集団項目の各反復に対して、必要な遊びバイトがコンパイラによって追加されます。遊びバイトについては、“1.3.5 データの境界調整”を参照してください。
3. OCCURS句を指定したデータ項目の反復回数は、以下のとおりです。

   ー 書き方1では、整数-2の値が一定の反復回数を表します。

   ー 書き方2では、データ名-1のデータ項目の現在の値が、反復回数を表します。

4. KEY IS指定を書いた場合、その反復データは、データ名-2の値に従って、昇順(ASCENDINGを書いたとき)または降順(DESCENDINGを書いたとき)に並んでいなければなりません。昇順および降順は、比較の規則に従って決められます。これらのデータ名は、キーの強さの順に左から右へ列挙します。

5. 内部ブール項目にINDEXED BY指定付きのOCCURS句を指定する場合、内部ブール項目にSYNCHRONIZED句を指定しなければなりません。

書き方2の規則

1. 書き方2は、OCCURS句を指定したデータ項目の反復回数が可変であることを示します。最大の反復回数を、整数-2に指定します。最小の反復回数を、整数-1に指定します。書き方2は、OCCURS句を指定したデータ項目の大きさが可変なのではなく、反復回数が可変であることを意味します。書き方2のOCCURS句を指定したデータ項目を、「可変反復データ項目」といいます。

2. OCCURS句を指定したデータ項目を参照した時点、またはOCCURS句を指定したデータ項目に従属するか従属されるデータ項目を参照した時点で、データ名-1の値は、整数-1以上かつ整数-2以下でなければなりません。データ名-1の値を変化させたとき、整数-2より大きい出現番号を持つデータ項目の内容は、規定されません。

3. 書き方2のOCCURS句を指定したデータ項目を従属する集団項目を参照する場合、OCCURS句を指定したデータ項目の表領域のうち、以下の部分が処理の対象となります。

   - その集団項目がデータ名-1を従属しない場合、処理を開始したときのデータ名-1のデータ項目の値をnとすると、第1番目から第n番目までの表要素までの表領域が対象になります。
   - その集団項目がデータ名-1を従属する場合、その集団項目が送出し側項目として用いられるときは、処理を開始したときのデータ名-1のデータ項目の値をnとすると、第1番目から第n番目までの表要素までの表領域が対象になります。その集団項目が受取り側項目として用いられるときは、整数-2が示す表領域が対象になります。

4. レコード記述項に書き方2のOCCURS句を指定し、かつ関連するファイル記述項または整列併合用ファイル記述項のRECORD句にVARYING指定を書いた場合、そのレコードは可変長です。

5. 書き方2のOCCURS句を指定したデータ項目を含むレコード記述項が、VARYING指定付きかつDEPENDING ON指定なしのRECORD句を書いたファイル記述項または整列併合用ファイル記述項に関連する場合、RELEASE文、REWRITE文またはWRITE文の実行前に、データ名-1に反復回数を設定しなければなりません。

6. SYNCHRONIZED句を省略した内部ブール項目に対して、書き方2のOCCURS句を指定することはできません。

5.4.7 PICTURE句

基本項目の項類、大きさおよび編集の形式を指定します。

使い方の例については、“付録H サンプル集”の“H.27 PICTURE句”を参照してください。

【書き方】

{ PICTURE / PIC } IS 文字列

構文規則

1. PICTURE句は、基本項目にだけ指定することができます。

2. PICTURE句の文字列は、COBOL文字集合に含まれる文字の特定の組合せで構成します。この組合せによって、基本項目の項類を指定します。

3. PICTURE句の文字“A”、“B”、“E”、“N”、“P”、“S”、“V”、“X”、“Z”、“CR”および“DB”に対応する小文字は、PICTURE句の文字列において、それらの大文字と等価です。

4. PICTURE句の文字列は、30文字以下でなければなりません。

5. PICTURE句は、指標データ項目、内部浮動小数点項目、ポインタデータ項目、int型2進整数データ項目またはRENAMES句の左辺を除くすべての基本項目に指定しなければなりません。指標データ項目またはRENAMES句の左辺に、この句を書くことはできません。
6. PICTUREとPICは同義語です。
7. PICTURE句の文字列にゼロ抑制記号の星印(＊)を指定した場合、BLANK WHEN ZERO句を指定することはできません。

一般規則

PICTURE句の文字の組合せと基本項目の項類

PICTURE句の文字の組合せにより、基本項目の項類を指定します。基本項目の項類には、以下の9種類があります。

- 英字
- 数字
- 英数字
- 英数字編集
- 数字編集
- 日本語
- 日本語編集
- ブール
- 外部浮動小数点

1. 項類が英字の基本項目を、「英字項目」といいます。英字項目を指定するときの規則、および英字項目の内容を以下に示します。
   - PICTURE句の文字列は、文字“A”だけで構成します。
   - 英字項目に指定できる用途は、DISPLAYだけです。
   - 英字項目の内容は、1つ以上の英字でなければなりません。
2. 項類が数字の基本項目を、「数字項目」といいます。数字項目を定義するときの規則、および数字項目の内容を以下に示します。
   - PICTURE句の文字列は、文字“9”、“P”、“S”および“V”の組合せで構成します。PICTURE句の文字列で表現できる数字の桁数は、1～18桁です。すなわち、文字“9”と“P”の個数の合計は、1～18でなければなりません。
   - 数字項目の用途は、以下でなければなりません。
     - DISPLAY
     - COMPUTATIONAL
     - BINARY
     - PACKED-DECIMAL
     - COMPUTATIONAL-5
   - 数字項目に符号を指定しなかった場合、数字項目の内容は、1桁以上の数字でなければなりません。数字項目に符号を指定した場合、数字項目の内容は、数字のほかに演算符号を含むことができます。
   - 小数点以下の桁を持たない数字項目を、「整数項目」といいます。

   数字項目には、上記の規則で定義される項目の他にint型2進整数データ項目があります。int型2進整数データ項目については、”5.4.15 USAGE句”を参照してください。

3. 項類が英数字の基本項目を、「英数字項目」といいます。英数字項目を定義するときの規則、および英数字項目の内容を以下に示します。

   ― PICTURE句の文字列は、文字“A”、“X”および“9”の組合せで構成します。PICTURE句の文字列に“X”以外の文字を含む英数字項目は、PICTURE句の文字列に“X”だけを指定した英数字項目と同じように扱われます。ただし、“A”だけまたは“9”だけからなるPICTURE句の文字列を持つデータ項目は、英数字項目ではありません。

   ― 英数字項目に指定できる用途は、DISPLAYだけです。

   ― 英数字項目の内容は、計算機文字集合の1つ以上の文字でなければなりません。

4. 項類が英数字編集の基本項目を、「英数字編集項目」といいます。英数字編集項目を定義するときの規則、および英数字編集項目の内容を以下に示します。

   ― PICTURE句の文字列は、文字“A”、“X”、“9”、“B”、“0”および“/”の組合せで構成します。この文字列には、“A”または“X”のいずれか1つを指定し、かつ、“B”、“0”または“/”のいずれか1つを指定しなければなりません。

   ― 英数字編集項目に指定できる用途は、DISPLAYだけです。

   ― 英数字編集項目の内容は、計算機文字集合の2つ以上の文字でなければなりません。

5. 項類が数字編集の基本項目を、「数字編集項目」といいます。数字編集項目を定義するときの規則、および数字編集項目の内容を以下に示します。

   ― PICTURE句の文字列は、文字“9”、“P”、“V”、“B”、“/”、“Z”、“0”、“,”、“.”、“＊”、“＋”、“－”、“CR”、“DB”および通貨編集用文字の組合せで構成します。この文字列には、“B”、“/”、“Z”、“0”、“,”、“.”、“＊”、“＋”、“－”、“CR”、“DB”または通貨編集用文字のうちのいずれか1つを指定しなければなりません。1つのPICTURE句の文字列に、文字“P”と“.”の両方を指定することはできません。PICTURE句の文字列で表現できる数字の桁数は、1～18桁です。すなわち、文字“9”と“P”の個数の合計は、1～18でなければなりません。

   ― 数字編集項目に指定できる用途は、DISPLAYだけです。

   ― 数字編集項目の各文字位置の内容は、PICTURE句の文字列と矛盾しないものでなければなりません。

6. 項類が日本語の基本項目を、「日本語項目」といいます。日本語項目を定義するときの規則、および日本語項目の内容を以下に示します。

   ― PICTURE句の文字列は、文字“N”だけで構成します。

   ― 日本語項目に指定できる用途は、DISPLAYだけです。

   ― 日本語項目の内容は、1つ以上の日本語文字でなければなりません。

7. 項類が日本語編集の基本項目を、「日本語編集項目」といいます。日本語編集項目を定義するときの規則、および日本語編集項目の内容を以下に示します。

   ― PICTURE句の文字列は、“N”および“B”の組合せで構成します。この文字列には、1つ以上の“N”および以上の“B”を指定しなければなりません。

   ― 日本語編集項目に指定できる用途は、DISPLAYだけです。

   ― 日本語編集項目の内容は、2つ以上の日本語文字でなければなりません。

8. 項類がブールの基本項目を、「ブール項目」といいます。ブール項目を定義するときの規則、およびブール項目の内容を以下に示します。

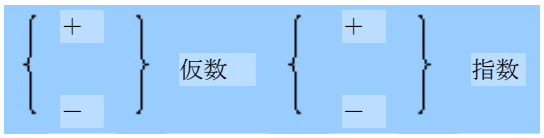
   ― PICTURE句の文字列は、文字“1”だけで構成します。

   ― ブール項目に指定できる用途は、BITまたはDISPLAYだけです。

   ― ブール項目の内容は、ブール文字“0”と“1”の組合せでなければなりません。

9. 項類が外部浮動小数点の基本項目を、「外部浮動小数点項目」といいます。外部浮動小数点項目に指定できる用途は、DISPLAYだけです。外部浮動小数点項目を定義するときの規則、および外部浮動小数点項目の内容を以下に示します。

   ― PICTURE句の文字列の形式を、以下に示します。

   ― 仮数部および指数部の符号は、1文字の正号“＋”または負号“－”でなければなりません。“＋”および“－”は、以下の意味を表します。

   - ＋:データ項目の値が正またはゼロの場合、“＋”を挿入し、データ項目の値が負の場合、“－”を挿入することを表します。“+”は、基本項目の大きさに数えます。
   - －:データ項目の値が正またはゼロの場合、空白を挿入し、データ項目の値が負の場合、“－”を挿入することを表します。“－”は、基本項目の大きさに数えます。

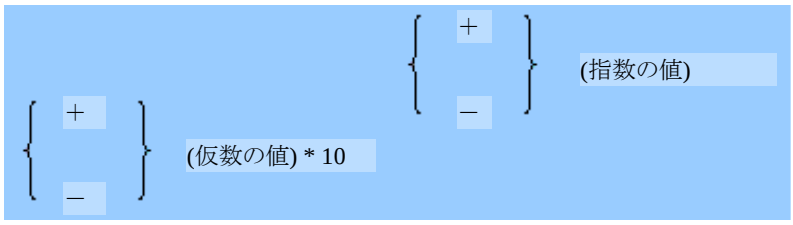
   ― 仮数は、“9”、“V”および“.”からなる文字列でなければなりません。この文字列には、小数点を表す“V”または“.”のいずれか1つを指定しなければなりません。これらの文字は、以下の意味を表します。

   - 9 ： 1個の数字を表します。“9”は基本項目の大きさに数えます。“9”の個数、すなわち仮数の桁数は、1～16でなければなりません。
   - V ： 想定小数点の位置を表します。“V”は基本項目の大きさに数えません。
   - . ： 実小数点の位置を表します。“.”は基本項目の大きさに数えます。

   ― “E”は、指数部が後続することを表します。“E”は基本項目の大きさに数えます。

   ― 指数は、文字列“99”でなければなりません。“99”は基本項目の大きさに数えます。

   ― 外部浮動小数点項目の値を式で表すと、以下のようになります。

   ― 外部浮動小数点項目に、VALUE句およびBLANK WHEN ZERO句を指定することはできません。

   ― 下表に、PICTURE句の文字列と外部浮動小数点項目の値の例を示します。

PICTURE句の文字列	外部データ形式	値
+9.9E+99	-5.4E-79	-5.4×(10の-79乗)
-99.9(5)E-99	12.34567E 00	12.34567×(10の0乗)
+9(8)VE-99	+12345678E-09	+12345678×(10の-9乗)
-V9(5)E+99	-.72000E+76	-0.72×(10の76乗)

PICTURE句の文字列の規則

1. 標準データ形式で数えた基本項目の桁数を、「基本項目の大きさ」といいます。基本項目の大きさは、文字位置を表す文字の個数で決めます。
2. 以下の文字は、同じ文字を繰り返して使うことができます。

   ― “A”、“,”、“X”、“9”、“P”、“Z”、“＊”、“B”、“/”、“0”、“＋”、“－”および通貨編集用文字

   ― “N”および“1”

同じ文字を繰り返す場合、その文字の次に、その文字の反復回数を括弧で囲んで指定します。反復回数は、1以上の符号なし整数でなければなりません。同じ文字を連続して書いたものと、反復回数を括弧で囲んで記述したものは等価です。例えば、“999”は“9(3)”と等価です。

3. 以下の文字は、1つのPICTURE句の文字列中に1つしか書くことができません。

   ― “E”

   ― “S”、“V”、“.”、“CR”および“DB”

4. 基本項目の大きさの最大値については、“付録B システムの定量制限”を参照してください。

PICTURE句の文字の意味

1. “A”は、英字または空白だけを含む1個の文字位置を示します。“A”は、基本項目の大きさに数えます。
2. “B”は、以下のどちらかを示します。

   ― 空白を挿入すべき1個の文字位置

   ― 日本語の空白を挿入すべき1個の日本語文字位置

   “B”は、基本項目の大きさに数えます。
3. “P”は、想定した位取りを示します。“P”は、基本項目の大きさに数えます。“P”は、PICTURE句の数字の桁位置の左端または右端に続けて書かなければなりません。PICTURE句の数字の桁位置の左端に“P”を書いた場合、先頭の“P”の左側に想定小数点があるものとみなされます。PICTURE句の数字の桁位置の右端に“P”を書いた場合、最後の“P”の右側に想定小数点があるものとみなされます。どちらの場合も、想定小数点の位置を示す文字“V”を省略することができます。
4. “S”は、演算符号が存在することを示します。“S”は、PICTURE句の文字列の先頭に書かなければなりません。SIGN句にSEPARATE CHARACTER指定を書いた場合、“S”は基本項目の大きさに数えます。SIGN句にそれ以外の指定を書いた場合またはSIGN句を省略した場合、“S”は基本項目の大きさに数えません。“S”は演算符号の表現形式および演算符号の位置を示すものではありません。
5. “V”は、想定小数点の位置を示します。“V”は、PICTURE句の文字列に1つしか書くことができません。“V”は、文字位置を表すものではないので、基本項目の大きさには数えません。桁位置または桁合わせ位置を表すPICTURE句の文字列(PまたはPの繰返し)の右端に想定小数点がある場合は、“V”を省略することができます。
6. “X”は、1個の文字位置を示します。“X”の文字位置は、計算機文字集合で許される任意の文字を含むことができます。“X”は、基本項目の大きさに数えます。
7. “N”は、1個の日本語文字位置を示します。“N”の日本語文字位置は、任意の日本語文字を含むことができます。“N”は、基本項目の大きさに数えます。
8. “Z”は、先行ゼロ列(有効な数字より上位の桁)を空白に置き換えることを示します。“Z”は、1個の文字位置として基本項目の大きさに数えます。
9. “9”は、数字を含む1個の桁位置を示します。“9”は、基本項目の大きさに数えます。
10. “1”は、ブール文字を含む1個のブール位置を示します。“1”は、基本項目の大きさに数えます。
11. “0”は、数字のゼロを挿入すべき1個の文字位置を示します。“0”は、基本項目の大きさに数えます。
12. “/”は、文字“/”を挿入すべき1個の文字位置を示します。“/”は、基本項目の大きさに数えます。
13. “,”(コンマ)は、文字“,”を挿入すべき1個の文字位置を示します。“,”は、基本項目の大きさに数えます。
14. “.”(小数点)は、文字“.”を挿入すべき1個の文字位置を示します。“.”は、データ項目の桁合わせの基準とする小数点の位置も示します。“.”は、基本項目の大きさに数えます。特殊名段落にDECIMAL-POINT IS COMMA句を書いた場合、プログラムに書いたPICTURE句の文字列および数字定数において、小数点とコンマの機能が入れ替わります。この場合、PICTURE句の“.”に関する規則をコンマに適用し、“,”に関する規則を小数点に適用します。
15. “＋”、“－”、“CR”および“DB”は、正負を表す文字を格納すべき文字位置を示します。これらの文字を、「符号編集用文字」といいます。1つのPICTURE句の文字列には、これらのうちの1種類しか指定できません。“＋”および“－”は、1個の文字位置として、基本項目の大きさに数えます。“CR”および“DB”は、2個の文字位置として、基本項目の大きさに数えます。

16. “＊”は、先行ゼロ列(有効な数字より上位の桁)を“＊”に置き換えることを示します。“＊”は、1個の文字位置として、基本項目の大きさに数えます。

17. 通貨編集用文字は、通貨編集用文字を格納すべき1個の文字位置を示します。「通貨編集用文字」とは、特殊名段落のCURRENCY SIGN句で指定した文字のことです。CURRENCY SIGN句を省略した場合は、通貨記号が通貨編集用文字になります。通貨編集用文字は、1個の文字位置として、基本項目の大きさに数えます。

18. “E”は、外部浮動小数点項目の指数部の開始を示します。“E”は、1個の文字位置として、基本項目の大きさに数えます。

“P”を含むデータ項目の規則

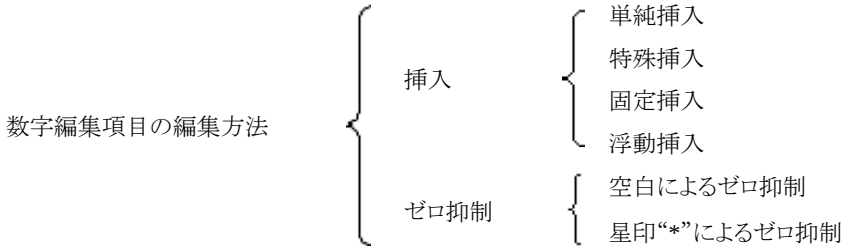
“P”を含むデータ項目を以下の場所に書いた場合、データ項目の値として、実際の文字表現ではなく代数値が用いられます。代数値とは、“P”の位置によって決まる位置に小数点を想定し、“P”を書いた桁位置にゼロを想定した値です。代数値の大きさは、PICTURE句の文字列によって表される桁位置の数、すなわち“P”と“9”の個数の合計です。

- 送出し側作用対象が数字項目の転記で、送出し側作用対象に“P”を含む数字項目を指定した場合
- MOVE文の送出し側作用対象に“P”を含む数字項目を指定した場合
- MOVE文の送出し側作用対象に“P”を含む数字編集項目を指定し、受取り側作用対象に数字項目または数字編集項目を指定した場合
- 両作用対象が数字項目の比較で、少なくともどちらか一方の作用対象に“P”を含む数字項目を指定した場合

上記以外の場所に“P”を含むデータ項目を書いた場合は、“P”を書いた桁位置は無視され、その作用対象の大きさに数えません。

編集項目の編集規則

PICTURE句の文字列では、挿入およびゼロ抑制の2種類の編集方法を指定することができます。数字編集項目のPICTURE句の文字列では、以下の1つ以上の編集方法を指定します。



英数字編集項目のPICTURE句の文字列では、空白、“0”または“/”の単純挿入を指定します。

日本語編集項目のPICTURE句の文字列では、日本語の空白の単純挿入を指定します。

単純挿入

1. 「単純挿入」とは、PICTURE句の文字列で指定した挿入文字の位置に、“,”、空白、“0”または“/”を挿入する方法です。単純挿入を行う場合、PICTURE句の文字列に、挿入文字として“,”、“B”、“0”または“/”を書きます。これらの挿入文字を、「単純挿入文字」といいます。単純挿入文字の“B”は、空白を挿入することを意味します。単純挿入文字は、挿入すべき文字の位置を表します。

2. PICTURE句の文字列の最後に“,”を書く場合、PICTURE句は、そのデータ記述項の最後の句でなければなりません。また、PICTURE句の文字列の直後に、分離符の終止符を書かなければなりません。この結果、データ記述項に“,.”が現れ、DECIMAL-POINT IS COMMA句を書くと、2つの連続した終止符“..”が現れます。

特殊挿入

1. 「特殊挿入」とは、PICTURE句の文字列で指定した挿入文字の位置に、小数点を挿入する方法です。特殊挿入を行う場合、PICTURE句の文字列に、挿入文字として“.”を書きます。この挿入文字は、単純挿入文字の1つとして扱われます。“.”は、挿入文字であるばかりでなく、実際の小数点の位置も表します。1つのPICTURE句の文字列に、“.”と“V”の両方を書くことはできません。

2. PICTURE句の文字列の最後に“.”を書く場合、PICTURE句は、そのデータ記述項の最後の句でなければなりません。また、PICTURE句の文字列の直後に、分離符の終止符を書かなければなりません。この結果、データ記述項に2つの連続した終止符“..”が現れ、DECIMAL-POINT IS COMMA句を書くと“,.”が現れます。

固定挿入

1. 「固定挿入」とは、PICTURE句の文字列で指定した挿入文字の位置に、データ項目の値に従って、挿入文字に対応する文字を挿入する方法です。固定挿入を行う場合、PICTURE句の文字列に、挿入文字として通貨編集用文字および符号編集用文字(“＋”、“－”、“CR”または“DB”)を書きます。これらの挿入文字を、「固定挿入文字」といいます。1つのPICTURE句の文字列には、通貨編集用文字と符号編集用文字をそれぞれ1個ずつしか書くことができません。
2. PICTURE句の文字列での“CR”および“DB”の位置は、基本項目の大きさに数える文字位置の最右端でなければなりません。
3. PICTURE句の文字列での“＋”および“－”の位置は、基本項目の大きさに数える文字位置の左端または右端でなければなりません。
4. 通貨編集用文字の位置は、“＋”または“－”が通貨編集用文字に先行する場合を除いて、基本項目の大きさに数える文字位置の左端でなければなりません。
5. PICTURE句の文字列に書いた通貨編集用文字の位置には、通貨編集用文字が挿入されます。PICTURE句の文字列に書いた符号編集用文字の位置には、データ項目の値に従って、下表に示す文字が挿入されます。

PICTURE 句の文字列に書いた符号編集用文字	編集の結果挿入される文字	
	データ項目の値が正またはゼロの場合	データ項目の値が負の場合
＋	＋	－
－	空白1個	－
CR	空白2個	CR
DB	空白2個	DB

浮動挿入

1. 「浮動挿入」とは、PICTURE句の文字列で指定した連続する挿入文字の位置に、データ項目の値に従って、挿入文字に対応する文字を挿入する方法です。浮動挿入を行う場合、PICTURE句の文字列に、挿入文字として、2つ以上の連続する通貨編集用文字、2つ以上の連続する“＋”、または2つ以上の連続する“－”を書きます。これらの挿入文字を、「浮動挿入文字」といいます。1つのPICTURE句の文字列に、“＋”と“－”の両方を書くことはできません。
2. 浮動挿入文字列の間、または浮動挿入文字列のすぐ右側に、単純挿入文字を書くことができます。これらの単純挿入文字は、浮動挿入文字列の一部とみなされます。
3. 浮動挿入文字として通貨編集用文字を書く場合、浮動挿入文字列のすぐ右側に、固定挿入文字の“CR”または“DB”を書くことができます。
4. 浮動挿入文字列の最左端の文字は、浮動挿入文字を挿入する最左端の文字位置を示します。浮動挿入文字列の最右端の文字は、浮動挿入文字を挿入する最右端の文字位置を示します。浮動挿入文字列の左から2番目の文字は、データ項目の中で数字データを格納することができる最左端の文字位置を示します。
5. PICTURE句の文字列の整数部に対応する数字位置の先頭の一部、または整数部に対応する数字位置の全部を挿入文字で指定した場合、浮動挿入の結果は以下のようになります。
   - データ項目の整数部の値がゼロの場合、小数点のすぐ左側の文字位置に挿入文字1個が挿入されます。挿入文字より左側の文字位置には、空白が挿入されます。
   - データ項目の整数部の値がゼロでない場合、整数部のゼロでない最初の数字のすぐ左側の文字位置に挿入文字1個が挿入されます。挿入文字より左側の文字位置には、空白が挿入されます。
6. PICTURE句の文字列の数字位置を全部挿入文字で指定した場合、浮動挿入の結果は以下のようになります。
   - データ項目の値がゼロの場合、データ項目全体に空白が挿入されます。

ー データ項目の値がゼロでない場合、小数点または整数部のゼロでない最初の数字のうちの左にあるもののすぐ左側の文字位置に、挿入文字1個が挿入されます。挿入文字より左側の文字位置には、空白が挿入されます。

7. PICTURE句の文字列に書いた通貨編集用文字の位置には、通貨編集用文字が挿入されます。PICTURE句の文字列に書いた“＋”または“－”の位置には、データ項目の値に従って、下表に示す文字が挿入されます。

PICTURE 句の文字列に書いた符号編集用文字	編集の結果挿入される文字	
	データ項目の値が正またはゼロの場合	データ項目の値が負の場合
＋	＋	－
－	空白1個	－

8. 数字編集項目を受取り側とする転記で切捨てが起こらないようにするためには、受取り側データ項目のPICTURE句の文字列の文字数は、以下の3つを加えたもの以上でなければなりません。

   ー 送出し側データ項目の文字数

   ー 受取り側データ項目のPICTURE句の文字列に指定した固定挿入文字の文字数

   ー 浮動挿入文字のための1文字

   切捨てが起こった場合、編集のために用いられるデータの値は、切捨て後の値になります。切捨ての方法については、“1.3.4 標準桁よせ規則”を参照してください。

ゼロ抑制

1. 「ゼロ抑制」とは、数字位置の先行ゼロ列(有効な数字より上位の桁)を空白または“＊”(星印)で置き換えることです。ゼロ抑制を行う場合、PICTURE句の文字列に、1つ以上の連続する“Z”、または1つ以上の連続する“＊”を書きます。これらの文字を、「ゼロ抑制文字」といいます。ゼロ抑制文字は、先行ゼロ列を置き換えるべき数字位置を示します。PICTURE句の文字列で“Z”または“＊”を書いた数字位置は、それぞれ空白または“＊”で置き換えられます。1つのPICTURE句の文字列に、“Z”と“＊”の両方を書くことはできません。

2. ゼロ抑制文字列の間、またはゼロ抑制文字列のすぐ右側に、単純挿入文字を指定することができます。これらの単純挿入文字は、ゼロ抑制文字列の一部とみなされます。

3. PICTURE句の文字列の整数部に対応する数字位置の先頭の一部、または整数部に対応する数字位置の全部をゼロ抑制文字で指定した場合、ゼロ抑制の結果は以下のようになります。

   ー データ項目の整数部の値がゼロの場合、小数点のすぐ左側までの先行ゼロ列が、空白または“＊”で置き換えられます。

   ー データ項目の整数部の値がゼロでない場合、整数部のゼロでない最初の数字のすぐ左側までの先行ゼロ列が、空白または“＊”で置き換えられます。

4. PICTURE句の文字列の数字位置を、全部ゼロ抑制文字で指定した場合、ゼロ抑制の結果は以下のようになります。

   ー データ項目の値がゼロの場合は、以下のとおりです。

   - ゼロ抑制文字として“Z”を書いた場合、データ項目全体が空白で置き換えられます。
   - ゼロ抑制文字として“＊”を書き、単純挿入文字“.”を書かなかった場合、データ項目全体が“＊”で置き換えられます。
   - ゼロ抑制文字として“＊”を書き、単純挿入文字“.”を書いた場合、“.”以外の文字位置が“＊”で置き換えられ、“.”の文字位置に小数点が置かれます。

   ー データ項目の値がゼロでない場合、小数点または整数部のゼロでない最初の数字のうちの左にあるもののすぐ左側までの先行ゼロ列が、空白または“＊”で置き換えられます。

浮動挿入とゼロ抑制の組合せ

1つの数字編集項目に対して、浮動挿入とゼロ抑制の両方を指定することはできません。また、2種類のゼロ抑制の両方を指定することもできません。したがって、文字“＋”、“－”、“＊”、“Z”および通貨編集用文字を浮動挿入文字またはゼロ抑制文字として使う場合、このうちの1種類しか書くことはできません。

PICTURE句の文字列の順序規則

PICTURE句の文字列の順序規則を、下表に示します。PICTURE句の文字列に書くことができる文字は、項類によって異なります。項類ごとに、表の規則を適用してください。

先行文字＼後続文字			単純挿入文字					固定挿入文字				ゼロ抑制文字		浮動挿入文字				その他の文字							
			B	0	/	,	.	+ −		CR DB	通貨	Z *		+ −		通貨		9	A X	S	V	P		N	1
								左	右			左	右	左	右	左	右					左	右		
単純挿入文字	B		○	○	○	○	○	○	−	−	○	○	○	○	○	○	○	○	○	−	○	−	○	○	−
	0		○	○	○	○	○	○	−	−	○	○	○	○	○	○	○	○	○	−	○	−	○	−	−
	/		○	○	○	○	○	○	−	−	○	○	○	○	○	○	○	○	○	−	○	−	○	−	−
	,		○	○	○	○	○	○	−	−	○	○	○	○	○	○	○	○	−	−	○	−	○	−	−
	.		○	○	○	○	−	○	−	−	○	○	−	○	−	○	−	○	−	−	−	−	−	−	−
固定挿入文字	+ −	左	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−
		右	○	○	○	○	○	−	−	−	○	○	○	−	−	○	○	○	−	−	○	○	○	−	−
	CR DB		○	○	○	○	○	−	−	−	○	○	○	−	−	○	○	○	−	−	○	○	○	−	−
	通貨		−	−	−	−	−	○	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−
ゼロ抑制文字	Z *	左	○	○	○	○	−	○	−	−	○	○	−	−	−	−	−	−	−	−	−	−	−	−	−
		右	○	○	○	○	○	○	−	−	○	○	○	−	−	−	−	−	−	−	○	−	○	−	−
浮動挿入文字	+ −	左	○	○	○	○	−	−	−	−	○	−	−	○	−	−	−	−	−	−	−	−	−	−	−
		右	○	○	○	○	○	−	−	−	○	−	−	○	○	−	−	−	−	−	○	−	○	−	−
	通貨	左	○	○	○	○	−	○	−	−	−	−	−	−	−	○	−	−	−	−	−	−	−	−	−
		右	○	○	○	○	○	○	−	−	−	−	−	−	−	○	○	−	−	−	○	−	○	−	−
その他の文字	9		○	○	○	○	○	○	−	−	○	○	−	○	−	○	−	○	○	○	○	−	○	−	−
	A X		○	○	○	−	−	−	−	−	−	−	−	−	−	−	−	○	○	−	−	−	−	−	−
	S		−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−
	V		○	○	○	○	−	○	−	−	○	○	−	○	−	○	−	○	−	○	−	○	−	−	−
	P	左	○	○	○	○	−	○	−	−	○	○	−	○	−	○	−	○	−	○	−	○	−	−	−
		右	−	−	−	−	−	○	−	−	○	−	−	−	−	−	−	−	−	○	○	−	○	−	−
	N		○	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	○	−
	1		−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	○

○ ： その列の上段に示した文字をその行の左側に示した文字の前(直前でなくてもよい)に書いてもよいことを示します。
− ： その列の上段に示した文字をその行の左側に示した文字の前に書いてはいけないことを示します。
通貨 ： 通貨編集用文字を示します。
左 ： 小数点の左側に、先行文字または後続文字を書いた場合を示します。
右 ： 小数点の右側に、先行文字または後続文字を書いた場合を示します。

5.4.8 PRINTING POSITION句

印字するときの横方向の印字位置を指定します。

【書き方】

PRINTING POSITION IS 整数-1

[BY 位置決め単位名-1]

構文規則

1. PRINTING POSITION句を日本語項目または日本語編集項目に指定する場合、そのデータ項目またはそのデータ項目を従属する集団項目に、CHARACTER TYPE句を指定しなければなりません。
2. OCCURS句を指定したデータ項目、またはそのデータ項目に従属するデータ項目に、PRINTING POSITION句を指定することはできません。
3. PRINTING POSITION句を指定したデータ項目を含むレコード記述項の中に、DEPENDING ON指定付きのOCCURS句を書くことはできません。
4. レベル番号が01または77以外のREDEFINES句を指定したデータ項目、およびそのデータ項目に従属するデータ項目に、PRINTING POSITION句を指定することはできません。
5. TYPEDEF句を指定したデータ項目、およびそのようなデータ項目に従属するデータ項目に、PRINTING POSITION句を指定することはできません。
6. PRINTING POSITION句を指定したデータ項目を含むレコード記述項の中のどのデータ項目も、RENAMES句の左辺および右辺に指定することはできません。
7. 位置決め単位名-1は、特殊名段落のPOSITIONING UNIT句で、位置決め単位を示す定数に対応付けなければなりません。

一般規則

1. 整数-1には、印字するデータ項目の印字行での絶対的な印字位置を指定します。
2. 位置決め単位名-1には、印字位置の単位を指定します。位置決め単位名-1を省略した場合、印字位置の単位は10CPIになります。
3. 整数-1には、印字行の最左端の印字位置を1として、印字するデータ項目の印字開始位置を指定します。
4. PRINTING POSITION句を集団項目に指定した場合、整数-1は集団項目の先頭の印字位置を示します。
5. 整数-1で示される印字位置は、印字するデータ項目を含むレコード記述項の中で、印字するデータ項目の直前に定義したデータ項目の右端の印字位置よりも右側でなければなりません。
6. PRINTING POSITION句を省略した場合、データ項目は以下の印字位置に印字されます。
   - レコード記述項の先頭のデータ項目は、印字行の1桁目から印字されます。
   - レコード記述項の2番目以降のデータ項目は、直前に定義したデータ項目の直後から印字されます。

5.4.9 REDEFINES句

同じ記憶領域に異なるデータ項目を定義します。

使い方の例については、“付録H サンプル集”の“H.28 REDEFINES句”を参照してください。

【書き方】

レベル番号 [データ名-1 / FILLER] [REDEFINES データ名-2]

注意

レベル番号、データ名-1およびFILLERは、書き方を明確にするために示してあるだけで、REDEFINES句の一部ではありません。

構文規則

1. REDEFINES句は、レベル番号、データ名-1またはFILLERの直後に書かなければなりません。
2. データ名-2とREDEFINES句の左辺のレベル番号は、同じでなければなりません。レベル番号は、66または88であってはいけません。
3. ファイル節のレコード記述項を2つ以上書くと、レコード記述項の領域が暗に再定義されます。ファイル節のレベル番号が01のデータ記述項に、REDEFINES句を書くことはできません。
4. データ名-2のデータ記述項に、OCCURS句を書くことはできません。データ名-2のデータ記述項は、OCCURS句を書いたデータ記述項に従属することができます。データ名-2のデータ記述項、REDEFINES句を書いたデータ記述項、およびそれらのデータ記述項に従属するデータ記述項に、DEPENDING ON指定付きのOCCURS句を書くことはできません。
5. レベル番号が01以外のデータ記述項にREDEFINES句を書く場合、またはデータ名-2が外部データレコードのレコード名の場合、データ名-2のデータ項目の文字位置の個数は、REDEFINES句の左辺によって参照されるデータ項目の文字位置の個数より大きいか等しくなければなりません。レベル番号が01のデータ記述項にREDEFINES句を書き、かつ、データ名-2に外部データレコードのレコード名でないデータ名を指定する場合は、このような制限はありません。
6. データ名-2のデータ記述項とREDEFINES句を書いたデータ記述項の間には、データ名-2のレベル番号より小さいかまたはREDEFINES句の左辺のレベル番号より小さいレベル番号を指定したデータ記述項を書くことはできません。
7. REDEFINES句を含む記述項は、再定義される文字位置の記述項の後に書かなければならず、この間に新しい文字位置を定義するデータ記述項を書くことはできません。
8. 同じ文字位置を重複して再定義することができます。同じ文字位置を重複して再定義する場合は、データ名-2は、その文字位置を最初に定義したデータ記述項のデータ名でなければなりません。

   ただし、このコンパイラでは、データ名-2は、その文字位置を最初に定義した記述項のデータ名である必要はありません。
9. REDEFINES句を書いたデータ記述項に、VALUE句を書くことはできません。ただし、そのデータ記述項に関連する条件名記述項には、VALUE句を書くことができます。
10. データ名-2のデータ項目は、REDEFINES句を指定したデータ項目に従属することができます。
11. データ名-2に、SYNCHRONIZED句を省略した内部ブール項目を指定する場合、再定義するデータ項目は、内部ブール項目でなければなりません。
12. データ名-2のデータ記述項に、CHARACTER TYPE句を記述してはなりません。また、データ名-2のデータ記述項は、CHARACTER TYPE句を書いたデータ記述項に従属してはいけません。
13. データ名-2のデータ記述項は、型宣言のものであってはなりません。
14. データ名-2のデータ記述項は、TYPE句を指定したものであってはなりません。
15. データ名-1およびデータ名-2は、強く型付けされた項目またはそれに従属する項目であってはなりません。
16. [Linux]では、REDEFINES句はint型2進整数データ項目には指定できません。

17. [Linux]では、データ名-2のデータ記述項は、int型2進整数データ項目であってはなりません。

一般規則

1. REDEFINES句を指定したデータ項目には、データ名-2の記憶領域の先頭から始まる、REDEFINES句を指定したデータ項目の大きさ分の記憶領域が割り当てられます。
2. 同じ文字位置を2つ以上のデータ記述項によって定義した場合、どのデータ記述項のデータ名も、その文字位置を参照するために用いることができます。
3. REDEFINES句を指定したデータ項目、またはそのデータ項目に従属する最初の基本項目にSYNCHRONIZED句を指定した場合、データ名-2は、SYNCHRONIZED句で指定した固有の境界に合わせて定義しなければなりません。たとえば、以下の場合、Aは半語境界から始まらなければなりません。

```
02  A PICTURE X(2).
02  B REDEFINES A PICTURE S9(4) BINARY SYNCHRONIZED.
```

4. REDEFINES句を指定した集団項目に従属する最初の項目がTYPE句を持ち、そのTYPE句の参照する型が集団項目として定義されている場合、データ名-2は、8バイト境界に合わせて定義しなければなりません。

5.4.10 RENAMES句

いくつかの基本項目をまとめたものに名前を付けます。

使い方の例については、“付録H サンプル集”の“H.21 RENAMES句”を参照してください。

【書き方】

66　データ名-1　RENAMES

データ名-2 [{ THROUGH / THRU } データ名-3] .

注意

【書き方】中のレベル番号およびデータ名-1は、書き方を明確にするために示してあるだけで、RENAMES句の一部ではありません。

構文規則

1. RENAMES句を書いたデータ記述項を、「RENAMES記述項」といいます。RENAMES記述項は、1つのレコード記述項の中にいくつでも書くことができます。
2. データ名-2およびデータ名-3は、同じレコード記述項中の基本項目または集団項目の名前でなければなりません。データ名-2およびデータ名-3は、同じ名前であってはいけません。データ名-2およびデータ名-3は、レベル番号が77、88、01または66のデータ記述項で定義したデータ名であってはいけません。
3. RENAMES記述項は、レコード記述項の最後のデータ記述項の直後にまとめて書かなければなりません。
4. データ名-1を修飾語として使用することはできません。
5. データ名-1を修飾できるのは、RENAMES記述項を含むレコード記述項中の01レベルのデータ記述項、ファイル記述項または整列併合用ファイル記述項の名前だけです。
6. データ名-2およびデータ名-3は、OCCURS句を指定したデータ項目、またはそのようなデータ項目に従属するデータ項目であってはいけません。
7. データ名-2およびデータ名-3は、修飾することができます。

8. データ名-2またはデータ名-3がRENAMES記述項を含むレコード記述項の中で一意でない場合、データ名-2またはデータ名-3を修飾しなければなりません。
9. データ名-2からデータ名-3までの範囲内に、可変反復データ項目を定義することはできません。
10. THROUGHとTHRUは同義語です。
11. データ名-3のデータ項目は、データ名-2のデータ項目に従属してはいけません。
12. データ名-3の領域の左端は、データ名-2の領域の左端より左にあってはいけません。データ名-3の領域の右端は、データ名-2の領域の右端より右になければなりません。
13. データ名-3を書く場合、データ名-2およびデータ名-3は内部ブール項目であってはいけません。
14. データ名-2は、OCCURS句の書き方2のデータ記述項の後の、それに従属しないデータ記述項であってはなりません。
15. データ名-2およびデータ名-3は、TYPEDEF句を指定した項目、およびその従属項目であってはなりません。
16. データ名-2およびデータ名-3は、TYPE句を指定した項目であってはなりません。
17. データ名-2からデータ名-3の範囲に、TYPE句を指定したデータ項目を定義することはできません。

一般規則

1. データ名-3を書くと、データ名-1は集団項目になります。この集団項目は、以下の範囲のデータ項目を従属します。
   - 集団項目に従属する最初の基本項目は、データ名-2が基本項目の場合はデータ名-2、データ名-2が集団項目の場合はデータ名-2に従属する最初の基本項目です。
   - 集団項目に従属する最後の基本項目は、データ名-3が基本項目の場合はデータ名-3、データ名-3が集団項目の場合はデータ名-3に従属する最後の基本項目です。
2. データ名-3を省略した場合、データ名-2のデータ属性がすべてデータ名-1のデータ属性になります。
3. データ名-3を書いた場合、データ名-1をFROM指定に指定したWRITE文の実行では、データ名-2、データ名-3およびその間の基本項目に指定したCHARACTER TYPE句およびPRINTING POSITION句の記述は無視されます。

5.4.11 SIGN句

演算符号の位置と表現形式を指定します。

使い方の例については、“付録H サンプル集”の“H.29 SIGN句”を参照してください。

【書き方】

[SIGN IS] { LEADING / TRAILING } [SEPARATE CHARACTER]

構文規則

1. SIGN句は、PICTURE句の文字列に文字“S”を指定した数字項目、またはそのような数字項目を少なくとも1つ含む集団項目にだけ指定することができます。
2. SIGN句が適用される数字項目の用途は、表示用でなければなりません。

一般規則

1. SIGN句は、数字項目の符号の位置と表現形式を指定します。SIGN句を基本項目に指定した場合、SIGN句はその基本項目に適用されます。SIGN句を集団項目に指定した場合、SIGN句は集団項目に従属するすべての符号

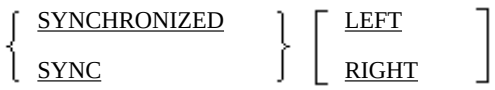
付き数字項目に対して適用されます。PICTURE句の文字列の“S”で符号の存在を指定し、符号の表示形式と位置はSIGN句で指定します。

2. SIGN句を指定した集団項目に従属する集団項目にSIGN句を指定した場合、従属する集団項目に対しては、そこで指定したSIGN句が適用されます。
3. SIGN句を指定した集団項目に従属する数字項目にSIGN句を指定した場合、数字項目に対しては、そこで指定したSIGN句が適用されます。
4. PICTURE句の文字列に“S”を指定した数字項目にSIGN句を指定しなかった場合、SIGN句にTRAILINGだけを書いたものとみなされます。
5. SEPARATE CHARACTERを省略した場合の符号の規則を、以下に示します。
   - 演算符号は、LEADINGを書いた場合は左端の桁位置、TRAILINGを書いた場合は右端の桁位置に付きます。
   - PICTURE句の文字列の“S”は、データ項目の大きさには数えません。
6. SEPARATE CHARACTERを書いた場合の符号の規則を、以下に示します。
   - 演算符号は、LEADINGを書いた場合は左端の桁位置、TRAILINGを書いた場合は右端の桁位置に付きます。演算符号の文字位置は、桁位置とは別の文字位置を占有します。
   - PICTURE句の文字列の中の文字“S”は、標準データ形式におけるデータ項目の大きさに数えます。
   - 正と負の演算符号は、標準データ形式の“＋”と“－”です。
   - 演算符号の文字位置に“＋”または“－”を設定していない場合の実行結果は規定されません。
7. SIGN句を指定した数字項目を計算や比較で使う場合、変換が行われることがあります。この変換は、自動的に行われます。SIGN句を指定したデータ項目の表現形式については、“5.4.15 USAGE句”を参照してください。

5.4.12 SYNCHRONIZED句

計算機の記憶装置の固有の境界に従って、基本項目を配置することを指定します。

【書き方】



構文規則

1. SYNCHRONIZED句は、基本項目にだけ指定することができます。ただし、[Win32][WinIPF][Winx64][Solaris][LinuxIPF][Linux64][.NET]では、SYNCHRONIZED句を集団項目に指定することもできます。
2. SYNCHRONIZEDはSYNCと同義語です。

一般規則

1. [Win32][WinIPF][Winx64][Solaris][LinuxIPF][Linux64][.NET]では、SYNCHRONIZED句を集団項目に指定した場合、SYNCHRONIZED句は、集団項目に従属するすべての基本項目に適用されます。
2. SYNCHRONIZED句は、USAGE句にBINARY、COMPUTATIONAL、COMPUTATIONAL-1、COMPUTATIONAL-2、COMPUTATIONAL-5,INDEXまたはBITを指定した基本項目にだけ有効です。その他の基本項目に指定しても無視されます。
3. SYNCHRONIZED句を指定した基本項目は、下表に示す境界に合わせて割り付けられます。

USAGE 句の指定	桁数	固有の境界
BINARY	1 ～ 4桁	2バイト(半語)境界
COMPUTATIONAL	5 ～ 18桁	4バイト(語)境界
COMPUTATIONAL-5	1 ～ 4桁	2バイト(半語)境界
	5 ～ 18桁	4バイト(語)境界
COMPUTATIONAL-1	―	4バイト(語)境界
COMPUTATIONAL-2	―	8バイト(倍語)境界
INDEX	―	4バイト(語)境界
BIT	―	1バイト境界

4. SYNCHRONIZED句を指定したデータ項目は、必要なら以下を挿入することにより、固有の境界に合わせて割り付けられます。遊びバイトおよび遊びビットについては、“1.3.5 データの境界調整”を参照してください。

   ― 遊びバイト

   ― 遊びビット

5. SYNCHRONIZED句を指定すると、SYNCHRONIZED句を省略した場合に比べてプログラムの実行効率が向上します。

6. LEFT指定およびRIGHT指定は、注釈とみなされます。

7. REDEFINES句を指定した基本項目、またはREDEFINES句を指定した集団項目に従属する最初の基本項目にSYNCHRONIZED句を指定する場合、REDEFINES句の右辺に指定したデータ項目を、SYNCHRONIZED句で指定した固有の境界に合わせておくことは、利用者の責任です。

8. コンパイラは、ファイル節、作業場所節、連絡節および定数節のレベル番号01のデータ項目、および連絡節のレベル番号77のデータ項目は、8バイト(倍語)境界から始まるものとみなします。利用者は、以下の方法でこのことを保証しなければなりません。

   a. ファイル節のレコードについては、レベル番号01のデータ項目が8バイト境界から始まるようにするために、必要なレコード間遊びバイトを定義しなければなりません。

   b. 作業場所節および定数節のレコードについては、この割付けは自動的に行われるので、利用者は考慮する必要はありません。

   c. 連絡節のレコードおよびレベル番号77のデータ項目については、呼ぶプログラムの中のこれらに対応するデータ項目を、8バイト境界から始まるようにしなければなりません。

5.4.13 TYPE句

データ記述項のTYPE句は、記述項のデータ記述が型宣言によって指定されたことを示します。

使い方の例については、“付録H サンプル集”の“H.31 TYPE句”を参照してください。

【書き方】

TYPE 型名-1

注意

TYPE句は、[Win32][WinIPF][Winx64][Solaris][Linux][LinuxIPF][Linux64][.NET]固有の機能です。

構文規則

1. 従属するデータ項目を含んでいる型名-1の記述は、この記述項の左辺を参照するTYPE句を含んでいてはなりません。
2. 記述項の左辺が従属している集団項目は、CHARACTER TYPE句、PRINTING POSITION句、USAGE句、SIGN句を含んでいてはなりません。
3. TYPE句を指定するデータ記述項に続いて、従属するデータ記述項を記述することはできません。
4. TYPE句を指定したデータ項目は、アドレス可変位置にあってはなりません。
5. 型名-1がSTRONG指定で宣言されている場合、TYPE句を指定したデータ項目は、レベル番号01のデータ項目であるか、STRONG指定の型宣言に従属する項目でなければなりません。
6. 型名-1がSTRONG指定で宣言されている場合、TYPEDEF句と同時に指定できません。

一般規則

1. TYPE句は、レベル番号、名前、GLOBALおよびEXTERNAL属性を除いて、型名-1で識別されるデータ記述をTYPE句の位置に記述したことと同じ効果があります。
2. 型名-1が集団項目である場合、型名-1に従属する項目のレベル番号は、以下の規則に従って調整されます。
   - TYPE句を指定した項目は、型名-1に従属する項目と同じ名前、記述、階層の従属項目を持つ集団項目となります。
   - 集団項目に従属する項目のレベル番号は、レベル番号の階層が49を超えない範囲で、型名-1の階層構造を保存するように調整されます。
   - TYPE句を指定した項目は、レベル番号01の項目と同じ8バイト境界に位置付けられます。
3. 記述項の左辺のデータ記述にVALUE句が指定された場合、このVALUE句の指定が受け入れられ、型名-1の記述に指定されたVALUE句は無視されます。

5.4.14 TYPEDEF句

TYPEDEF句は、これが型宣言であることを指定します。

使い方の例については、“付録H サンプル集”の“H.30 TYPEDEF句”を参照してください。

【書き方】

IS TYPEDEF [STRONG]

TYPEDEF句は、[Win32][WinIPF][Winx64][Solaris][Linux][LinuxIPF][Linux64][.NET]固有の機能です。
ただし、[.NET]ではSTRONGは指定できません。

構文規則

1. TYPEDEF句はSQL宣言節に記述することはできません。
2. 基本項目にはSTRONGは指定できません。
3. TYPEDEF句はSTRONG指定の型を参照するTYPE句と同時に指定できません。
4. STRONGを指定する場合、従属する項目にDEPENDING ON指定のOCCURS句を含んではなりません。

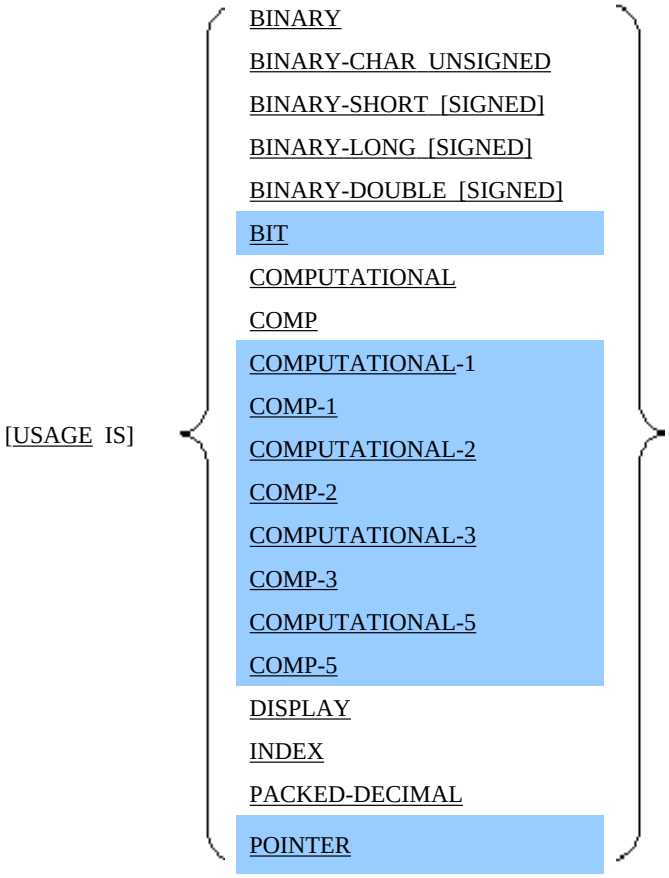
一般規則

1. TYPEDEF句が指定された場合、データ記述項は型宣言になります。記述項に指定されたデータ名は型名とみなされます。また、従属するデータ記述項はこの型の型宣言の一部です。これらの従属するデータ記述項のデータ名は、型名を使用して定義される集団項目の従属項目としてだけ参照できます。
   そのような集団項目が1つ以上存在するならば、集団項目の名前を用いた修飾が必要です。また、そのような集団項目が1つも存在しない場合、これらの従属するデータ記述項のデータ名は参照することができず、この名前により参照されるいかなるリソースも存在しません。
2. 型宣言は、それと関連付けられる領域を持ちません。
3. GLOBAL句は、型名の範囲に適応されます。その他のデータ記述項の句指定および従属するデータ記述は、型名を用いて定義されたデータ記述に記述されたものとみなされます。

5.4.15 USAGE句

データ項目の用途を指定します。

【書き方】



注意

BINARY-CHAR、BINARY-SHORT、BINARY-LONG、BINARY-DOUBLEは、[Win32][WinIPF][Winx64][Solaris][Linux][LinuxIPF][Linux64][.NET]固有の機能です。

構文規則

1. USAGE句は、レベル番号が66または88以外のデータ記述項に書くことができます。
2. USAGE句を集団項目のデータ記述項に書く場合、従属する基本項目または集団項目のデータ記述項にもUSAGE句を書くことができます。その場合、両方のデータ記述項に同じUSAGE句を書かなければなりません。
3. USAGE句で以下を指定した基本項目、およびUSAGE句で以下を指定した集団項目に従属する基本項目は、数字項目でなければなりません。すなわち、PICTURE句の文字列を“P”、“S”、“V”および“9”だけで構成しなければなりません。
   - BINARY
   - COMPUTATIONALまたはCOMP
   - COMPUTATIONAL-5またはCOMP-5
   - PACKED-DECIMAL
   - COMPUTATIONAL-3またはCOMP-3
4. 以下の2つの組の語は、それぞれ同義語です。どちらを書いてもかまいません。
   - COMPUTATIONALとCOMP
   - COMPUTATIONAL-1とCOMP-1
   - COMPUTATIONAL-2とCOMP-2
   - COMPUTATIONAL-3とCOMP-3
   - COMPUTATIONAL-5とCOMP-5
5. USAGE IS INDEX句を指定した基本項目、およびUSAGE IS INDEX句を指定した集団項目に従属する基本項目は、以下の場所にだけ書くことができます。
   - SEARCH文
   - SET文
   - DISPLAY文
   - 比較条件
   - 手続き部の見出しのUSING指定
   - 手続き部のENTRY文のUSING指定
   - CALL文のUSING指定
6. USAGE IS INDEX句を書いたデータ記述項、およびそのデータ記述項に従属するデータ記述項に、以下の句を書くことはできません。
   - BLANK WHEN ZERO句
   - CHARACTER TYPE句
   - JUSTIFIED句
   - PICTURE句
   - PRINTING POSITION句
   - VALUE句
7. USAGE IS INDEX句を指定した基本項目に、条件名記述項を関係付けてはいけません。
8. 定数節のデータ項目にUSAGE IS INDEX句を指定することはできません。
9. USAGE句にBITを指定した基本項目、またはUSAGE句にBITを指定した集団項目に従属する基本項目は、ブール項目でなければなりません。すなわち、PICTURE句の文字列を“1”だけで構成しなければなりません。

10. USAGE句にCOMPUTATIONAL-1、COMP-1、COMPUTATIONAL-2またはCOMP-2を書いたデータ記述項、およびそのデータ記述項に従属するデータ記述項に、以下の句を書くことはできません。

    ― BLANK WHEN ZERO句

    ― CHARACTER TYPE句

    ― JUSTIFIED句

    ― PICTURE句

    ― PRINTING POSITION句

    ― VALUE句

11. USAGE IS POINTER句は、以下のデータ記述項にだけ書くことができます。

    ― 基底場所節

    ― 作業場所節

    ― 定数節

    ― 連絡節

12. USAGE IS POINTER句を書いたデータ記述項、およびそのデータ記述項に従属するデータ記述項に、以下の句を書くことはできません。

    ― BLANK WHEN ZERO句

    ― CHARACTER TYPE句

    ― JUSTIFIED句

    ― PICTURE句

    ― PRINTING POSITION句

    ― SIGN句

13. ポインタデータ項目は、データ部ではBASED ON句のデータ名にだけ書くことができます。

14. ポインタデータ項目は、手続き部では、以下の場所にだけ書くことができます。

    ― 手続き部の見出しのUSING指定、RETURNING指定

    ― ENTRY文のUSING指定

    ― CALL文のUSING指定、RETURNING指定

    ― DISPLAY文

    ― MOVE文

    ― IF文およびEVALUATE文の比較条件

15. [Linux]では、USAGE IS BINARY-CHAR、BINARY-SHORT、BINARY-LONG、BINARY-DOUBLEの各句は、レベル番号が01または77の基本項目にだけ指定できます。

16. [Linux]では、USAGE IS BINARY-CHAR、BINARY-SHORT、BINARY-LONG、BINARY-DOUBLEの各句は、作業場所節、定数節および連絡節にだけ書くことができます。また、以上の句が指定されたデータ項目は手続き部でのみ参照できます。

17. [Linux]では、USAGE句にBINARY-CHAR、BINARY-SHORT、BINARY-LONGまたはBINARY-DOUBLEを指定したデータ記述項は、条件変数であってはなりません。

18. USAGE句にBINARY-CHAR、BINARY-SHORT、BINARY-LONGまたはBINARY-DOUBLEのどれかを指定したデータ記述項に以下の句を書くことはできません。

    また、[Win32][WinIPF][Winx64][Solaris][LinuxIPF][Linux64][.NET]では、上記のデータ記述項に従属するデータ記述項にも、以下の句を書くことはできません。

    ― BLANK WHEN ZERO句

― JUSTIFIED句

― PICTURE句

[Linux]の場合、以下の句も書くことができません。

― CHARACTER TYPE句

― PRINTING POSITION句

― REDEFINES句

一般規則

1. USAGE句を集団項目に指定した場合、USAGE句は、集団項目に従属するすべての基本項目に適用されます。
2. USAGE句は、データ項目の用途を指定します。USAGE句の記述によって記憶領域内のデータ項目の表現形式が決められます。
3. 数字項目の内部表現は、USAGE句の記述によって決められます。下表に、数字項目の内部表現を示します。

USAGE 句	SIGN句	PICTURE句	値	内部表現	備考
DISPLAY(外部10進)	なし	9(4)	6789	36373839	3 :ゾーンビット 4 :正の演算符号 5 :負の演算符号
		S9(4)	+6789	36373849	
			-6789	36373859	
	LEADING	S9(4)	+6789	46373839	
			-6789	56373839	
	TRAILING	S9(4)	+6789	36373849	
			-6789	36373859	
	LEADING SEPARATE	S9(4)	+6789	2B36373839	2B:標準データ形式の+ 2D:標準データ形式の-
			-6789	2D36373839	
	TRAILING SEPARATE	S9(4)	+6789	363738392B	
			-6789	363738392D	
BINARY, COMP(2進)	―	9(4)	1234	04D2	最左端ビットが演算符号を表します。 0 :正、1 :負 負の値は2の補数で表現されます。
		S9(4)	+1234	04D2	
			-1234	FB2E	
[DS][HP][Solaris] COMP-5(2進)	―	9(4)	1234	04D2	最左端ビットが演算符号を表します。 0 :正、1 :負 負の値は2の補数で表現されます。
		S9(4)	+1234	04D2	
			-1234	FB2E	
[Win16][Win32][WinIPF][Winx64][Linux][LinuxIPF][Linux64][.NET] COMP-5(2進)	―	9(4)	1234	D204	最右端バイトの最左端ビットが演算符号を表します。 0 :正、1 :負 負の値は2の補数で表現されます。
		S9(4)	+1234	D204	
			-1234	2EFB	
PACKED-DECIMAL	―	9(4)	1234	01234F	F :絶対値を表す演算符号

USAGE 句	SIGN句	PICTURE句	値	内部表現	備考
（内部10進）		S9(4)	+1234	01234C	C：正の演算符号
			-1234	01234D	D：負の演算符号

－：指定できない組合せ

4. ブール項目の内部表現は、USAGE句の記述によって決められます。下表に、ブール項目の内部表現を示します。

USAGE 句	PICTURE句	値	内部表現	備考
DISPLAY（外部ブール）	1(8)	11100101	3131313030313031	1文字の“0”または“1”で表現されます。
BIT（内部ブール）	1(8)	11100101	E5	1ビットの0または1で表現されます。

5. 基本項目または基本項目を従属する集団項目にUSAGE句を指定しなかった場合、USAGE IS DISPLAY句を指定したものとみなされます。USAGE IS DISPLAY句が有効なデータ項目を、「用途が表示用のデータ項目」といいます。

6. USAGE IS DISPLAY句は、データ項目を記憶装置内で標準データ形式で表現すること、およびデータ項目を文字境界に桁よせすることを表します。USAGE IS DISPLAY句を明にまたは暗に指定したデータ項目の内容は、文字位置すなわちバイトを単位として扱われます。標準データ形式のデータ項目が占める記憶領域の大きさを、「文字位置の個数」といいます。文字位置の個数は、バイト数と等価です。

7. USAGE IS DISPLAY句は、以下に指定することができます。

   － 英字項目、英数字項目、英数字編集項目、数字編集項目、数字項目

   － 日本語項目、日本語編集項目、ブール項目、外部浮動小数点項目

   また、USAGE IS DISPLAY句を集団項目に指定した場合、集団項目に従属する基本項目のうち、これらの基本項目にUSAGE IS DISPLAY句が適用されます。

8. USAGE IS DISPLAY句を明にまたは暗に指定した数字項目を、「外部10進項目」といいます。外部10進項目の各数字の桁位置は、文字位置すなわちバイトを単位として表現されます。

9. 外部10進項目のPICTURE句の文字列に文字“S”を指定した場合、SIGN句の記述に従って、演算符号が以下のように表現されます。SIGN句を省略した場合は、SIGN IS TRAILING句を指定した場合と同じです。

   － SIGN IS LEADING句を指定した場合、先頭の桁位置の上位4ビットに、演算符号が含まれます。

   － SIGN IS TRAILING句を指定した場合、最後の桁位置の上位4ビットに、演算符号が含まれます。

   － SIGN IS LEADING SEPARATE CHARACTER句を指定した場合、先頭の桁位置の直前の文字位置に、標準データ形式の“＋”または“－”が置かれます。

   － SIGN IS TRAILING SEPARATE CHARACTER句を指定した場合、最後の桁位置の直後の文字位置に、標準データ形式の“＋”または“－”が置かれます。

10. USAGE IS DISPLAY句を明にまたは暗に指定したブール項目を、「外部ブール項目」といいます。外部ブール項目の各ブール位置には、標準データ形式の“1”または“0”が含まれます。

11. 以下を明にまたは暗に指定した数字項目を、「2進項目」といいます。

    － USAGE IS COMPUTATIONAL句

    － USAGE IS COMPUTATIONAL-5句

    － USAGE IS BINARY句

    USAGE IS COMPUTATIONAL句とUSAGE IS BINARY句は同義です。

[DS][HP][Solaris]では、USAGE IS COMPUTATIONAL-5句も上記の2つと同義となりますが、[Win16][Win32][WinIPF][Winx64][Linux][LinuxIPF][Linux64][.NET]ではUSAGE IS COMPUTATIONAL-5句だけ内部表現が異なります。

2進項目に割り付けられる記憶領域の大きさは、PICTURE句で指定した桁数によって、以下のように決まります。

- 1～4桁の場合:2バイト
- 5～9桁の場合:4バイト
- 10～18桁の場合:8バイト

12. USAGE IS INDEX句を明にまたは暗に指定した基本項目を、「指標データ項目」といいます。指標データ項目は、表要素の出現番号に対応する値を持ちます。指標データ項目には、SET文などにより、指標名に対応する値がそのまま格納されます。
指標データ項目の大きさは4バイトです。ただし、[WinIPF][Winx64][LinuxIPF][Linux64]では、指標データ項目の大きさは8バイトです。
13. 集団項目にUSAGE IS INDEX句を指定した場合、集団項目に従属する基本項目は、すべて指標データ項目になります。しかし、USAGE IS INDEX句を指定した集団項目そのものは、指標データ項目ではありません。
14. MOVE文または入出力文に、指標データ項目を従属する集団項目を指定した場合、指標データ項目の変換は行われません。
15. USAGE IS PACKED-DECIMAL句を明にまたは暗に指定した数字項目を、「内部10進項目」といいます。内部10進項目には、最後のバイト以外の各バイトに2桁の数字が含まれ、最後のバイトの下位4ビットに演算符号が含まれます。
16. USAGE IS BIT句を明にまたは暗に指定したブール項目を、「内部ブール項目」といいます。内部ブール項目の1個のブール文字は、1ビットの“1”または“0”で表現されます。
17. USAGE IS COMPUTATIONAL-1句またはUSAGE IS COMP-1句を明にまたは暗に指定した基本項目を、「単精度内部浮動小数点項目」といいます。単精度内部浮動小数点項目の長さは、4バイトです。
18. USAGE IS COMPUTATIONAL-2句またはUSAGE IS COMP-2句を明にまたは暗に指定した基本項目を、「倍精度内部浮動小数点項目」といいます。倍精度内部浮動小数点項目の長さは、8バイトです。
19. USAGE IS COMPUTATIONAL-1句、USAGE IS COMP-1句、USAGE IS COMPUTATIONAL-2句またはUSAGE IS COMP-2句を明にまたは暗に指定した基本項目を、「内部浮動小数点項目」といいます。
20. 浮動小数点項目の内部データ形式は、以下のとおりです。
    - 仮数の符号は、最左端ビットです。
    - 指数は、単精度では最左端ビットに続く8ビットを占めます。倍精度では最左端ビットに続く11ビットを占めます。
    - 仮数は、単精度では指数に続く23ビットを占めます。倍精度では指数に続く52ビットを占めます。
21. USAGE IS COMPUTATIONAL-3句またはUSAGE IS COMP-3句は、USAGE IS PACKED-DECIMAL句と同義です。
22. USAGE IS POINTER句を明にまたは暗に指定した基本項目を、「ポインタデータ項目」といいます。ポインタデータ項目は、記憶領域のアドレスを示す値を保持することができます。
ポインタデータ項目の大きさは4バイトです。ただし、[WinIPF][Winx64][LinuxIPF][Linux64]では、ポインタデータ項目の大きさは8バイトです。
23. USAGE句にBINARY-CHAR、BINARY-SHORT、BINARY-LONGおよびBINARY-DOUBLEのどれかが指定されたデータ記述項を「int型2進整数データ項目」といいます。

    また、[Win32][WinIPF][Winx64][Solaris][LinuxIPF][Linux64][.NET]では、上記のデータ記述項に従属する基本項目も「int型2進整数データ項目」といいます。

    int型2進整数データ項目は、数字項目です。また、整数項目および2進項目でもあります。

    int型2進整数データ項目が占める記憶領域全体は、数値データとして有効になります。

下表にint型2進整数データ項目に割り付けられる記憶領域の大きさと保持できる値の範囲および内部表現を示します。

表5.1 int型2進整数データ項目の大きさと値の範囲

USAGE句種別	項目の大きさ(バイト)	値の範囲
BINARY-CHAR UNSIGNED	1	0 ≦ n ＜ 256 [2**8]
BINARY-SHORT SIGNED	2	-32768 [-2**15] ≦ n ＜ 32768 [2**15]
BINARY-LONG SIGNED	4	-2147483648 [-2**31] ≦ n ＜ 2147483648 [2**31]
BINARY-DOUBLE SIGNED	8	-9223372036854775808 [-2**63] ≦ n ＜ 9223372036854775808 [2**63]

表5.2 [Solaris]int型2進整数データ項目の内部表現

USAGE 句	値	内部表現	備考
BINARY-CHAR UNSIGHNED	123	7B	全ビットで正数値を表します。
BINARY-SHORT SIGNED	+1234	04D2	最左端ビットが演算符号を表します。 0:正、1:負 負の値は2の補数で表現されます。
	-1234	FB2E	
BINARY-LONG SIGNED	+1234	000004D2	
	-1234	FFFFFB2E	
BINARY-DOUBLE SIGNED	+1234	00000000000004D2	
	-1234	FFFFFFFFFFFFFB2E	

表5.3 [Win32][Linux][LinuxIPF][Linux64][.NET][WinIPF][Winx64]int型2進整数データ項目の内部表現

USAGE 句	値	内部表現	備考
BINARY-CHAR UNSIGHNED	123	7B	全ビットで正数値を表します。
BINARY-SHORT SIGNED	+1234	D204	最右端バイトの最左端ビットが演算符号を表します。 0:正、1:負 負の値は2の補数で表現されます。
	-1234	2EFB	
BINARY-LONG SIGNED	+1234	D2040000	
	-1234	2EFBFFFF	
BINARY-DOUBLE SIGNED	+1234	D204000000000000	
	-1234	2EFBFFFFFFFFFFFF	

5.4.16 VALUE句

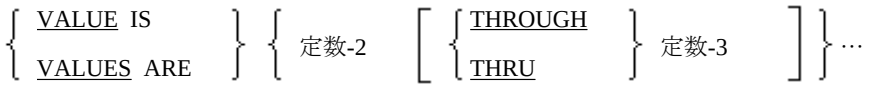
データ項目に初期値を与えます。または、条件名の値を指定します。

使い方の例については、“付録H サンプル集”の“H.22 VALUE句”を参照してください。

【書き方1】データ項目に初期値を与える

VALUE IS 定数-1

【書き方2】条件名の値を指定する



構文規則

1. THROUGHとTHRUは同義語です。
2. 数字項目にVALUE句を指定する場合、または数字項目の条件変数の条件名にVALUE句を指定する場合、定数-1～定数-3は、数字定数でなければなりません。このとき、数字定数の値は、数字項目のPICTURE句で示される値の範囲に入るものでなければなりません。
3. 符号付き数字項目、または符号付き数字項目を条件変数とする条件名にVALUE句を指定する場合、定数-1～定数-3は、符号付き数字定数でなければなりません。
4. 定数-1～定数-3に以下を指定する場合、その値はPICTURE句に指定した大きさを超えてはいけません。
   - 文字定数
   - 日本語定数またはブール定数
5. PICTURE句の文字列に“P”を指定したデータ項目のVALUE句に指定できる値の例を、下表に示します。

PICTURE句の文字列	VALUE句に指定できる値
999PP	0、100、200、…、99900
PP999	0、.00001、.00002、…、.00999

6. 以下のデータ記述項に、VALUE句を書くことはできません。ただし、以下のデータ記述項に関係付けた条件名記述項には、VALUE句を書くことができます。
   - EXTERNAL句を書いたデータ記述項、またはそのデータ記述項に従属するデータ記述項
   - REDEFINES句を書いたデータ記述項、またはそのデータ記述項に従属するデータ記述項
7. 条件変数がブール項目である条件名に、THROUGH指定を指定することはできません。

一般規則

書き方1および書き方2に共通する規則

1. VALUE句は、VALUE句を書いたデータ記述項およびこのデータ記述項を従属するデータ記述項に書いた他の句と矛盾するものであってはいけません。
2. 数字項目にVALUE句を指定する場合、VALUE句の定数は数字定数でなければなりません。

   作業場所節または定数節の数字項目の場合、VALUE句の定数は、標準の桁よせの規則に従って格納されます。
3. 英字項目、英数字項目、英数字編集項目、数字編集項目または集団項目にVALUE句を指定する場合、VALUE句の定数は文字定数でなければなりません。
4. 日本語項目または日本語編集項目にVALUE句を指定する場合、VALUE句の定数は日本語定数でなければなりません。
5. ブール項目にVALUE句を指定する場合、VALUE句の定数はブール定数でなければなりません。
6. 内部浮動小数点項目にVALUE句を指定する場合、VALUE句の定数は、浮動小数点定数、数字定数または表意定数ZEROでなければなりません。
7. ポインタデータ項目にVALUE句を指定する場合、VALUE句の定数は表意定数ZEROでなければなりません。このときの表意定数ZEROの値は、数値0です。

8. 英数字編集項目、数字編集項目または日本語編集項目にVALUE句を指定する場合、VALUE句の定数は、編集の済んだ形式で指定しなければなりません。
9. VALUE句を指定したデータ項目にBLANK WHEN ZERO句またはJUSTIFIED句を指定しても、それらの句の機能は無視されます。

書き方1の規則

1. VALUE句をどのデータ項目に指定できるかは、データ項目を定義する節によって、以下のように異なります。
   - ファイル節の場合、条件名記述項の中にだけVALUE句を書くことができます。ファイル節のデータ項目の初期値は、規定されません。
   - 連絡節の場合、条件名記述項の中にだけVALUE句を書くことができます。
   - 作業場所節の場合、条件名記述項とデータ記述項にVALUE句を書くことができます。作業場所節のデータ記述項のVALUE句は、そのプログラムが初期状態に置かれたときにだけ効果があります。VALUE句を指定したデータ項目は、VALUE句に指定した定数の値によって初期化されます。VALUE句を省略したデータ項目の初期値は、規定されません。
   - 定数節の場合、VALUE句を指定してデータ項目の値を与えなければなりません。
2. 集団項目にVALUE句を指定する場合、VALUE句の定数は、表意定数または文字定数でなければなりません。集団項目の領域には、それに従属する個々の基本項目や集団項目に関係なく初期値が格納されます。
3. VALUE句を指定した集団項目に従属する個々のデータ項目に、VALUE句を指定することはできません。
4. JUSTIFIED句、SYNCHRONIZED句、またはUSAGE IS DISPLAY以外のUSAGE句を指定したデータ項目を従属する集団項目には、VALUE句を指定することはできません。
5. 可変反復データ項目、可変反復データ項目を従属する集団項目、または可変反復データ項目に従属するデータ項目にVALUE句を指定した場合、可変反復データ項目の大きさが、可変反復データ項目の最大の大きさ(OCCURS句のTO 整数-2 TIMESの整数-2の値)であるとみなされて、初期値が格納されます。このとき、可変反復データ項目のDEPENDING ON指定に指定したデータ項目にVALUE句を指定していても、その値は可変反復データ項目の初期化には影響を与えません。
6. OCCURS句を指定したデータ項目、またはそのデータ項目に従属するデータ項目にVALUE句を指定した場合、VALUE句に指定した値が、関連するデータ項目の各繰り返しに格納されます。

書き方2の規則

1. 書き方2のVALUE句は、条件名記述項にだけ書くことができます。
2. 条件名記述項には、条件名および書き方2のVALUE句を書かなければなりません。条件名の性質は、条件変数によって決まります。
3. THRU指定を書く場合、
   - 条件変数が、項類が英字項目、英数字項目、英数字編集項目、日本語項目、日本語編集項目または数字編集項目の基本項目であるか、集団項目であるなら、定数-2は各文字位置の文字を文字の大小順序(collating sequence)に従って比較した結果、定数-3より小さくなければなりません。
   - 条件変数が上記以外の項類の基本項目なら、定数-2は代数的に定数-3より小さな値でなければなりません。

5.4.17 BASED ON句

基底場所節のデータ項目に、暗黙のポインタ修飾子を指定します。

使い方の例については、“付録H サンプル集”の“H.32 BASED ON句”を参照してください。

【書き方】

BASED ON データ名-1

構文規則

1. BASED ON句は、レベル番号が01または77の、基底場所節のデータ記述項にだけ書くことができます。
2. BASED ON句を書いたデータ記述項に、REDEFINES句を書いてはなりません。
3. BASED ON句を書いたデータ記述項に、強い型を参するTYPE句を書いてはなりません。
4. データ名-1は、以下のいずれかで定義したポインタデータ項目でなければなりません。
   - 作業場所節
   - 連絡節
5. データ名-1は、修飾することができます。

一般規則

1. BASED ON句を指定したデータ項目またはそれに従属するデータ項目を、明にポインタ付けしないで参照する場合、データ名-1には、参照可能な領域のアドレスを示す値を設定しておかなければなりません。
2. BASED ON句を指定したデータ項目をREDEFINES句の右辺に書いた場合、REDEFINES句の左辺のデータ項目には、同じBASED ON句を指定したものとみなされます。同様に、BASED ON句を指定したデータ項目をRENAMES句の右辺に書いた場合、RENAMES句の左辺のデータ項目には、同じBASED ON句を指定したものとみなされます。

5.5 画面データ記述項

画面データ記述項では、画面項目を定義します。画面データ記述項では、画面項目を画面上に表示するときの配置と表示属性、画面項目の大きさ、入出力属性などを定義します。画面データ記述項は、画面節に書くことができます。

画面項目には、基本画面項目と集団画面項目の2種類があります。「基本画面項目」とは、画面上のそれぞれの独立した領域のことです。「集団画面項目」とは、いくつかの基本画面項目をまとめたものです。基本画面項目は、さらに入出力属性によって、「定数項目」、「入力項目」、「出力項目」、および「更新項目」の4種類があります。

注意

画面データ記述項は[WinIPF][Linux][LinuxIPF][Linux64][.NET]では使用できません。

【書き方1】集団画面項目を定義する

```
レベル番号
  [ データ名-1 ]
  [ FILLER     ]
    [AUTO句]
    [BACKGROUND-COLOR句]
    [BLANK  SCREEN句]
    [FOREGROUND-COLOR句]
    [FULL句]
    [REQUIRED句]
    [SECURE句]
    [SIGN句]
    [USAGE句].
```

【書き方2】定数項目を定義する

レベル番号

[データ名-1]
[FILLER]

[BACKGROUND-COLOR句]
[BELL句]
[BLANK LINE句]
[BLANK SCREEN句]
[BLINK句]
[COLUMN NUMBER句]
[ERASE句]
[FOREGROUND-COLOR句]
[HIGHLIGHT句]
[LOWLIGHT句]
[LINE NUMBER句]
[REVERSE-VIDEO句]
[UNDERLINE句]
VALUE句.

【書き方3】入力項目、出力項目または更新項目を定義する

レベル番号

[データ名-1]
[FILLER]

[AUTO句]
[BACKGROUND-COLOR句]
[BELL句]
[BLANK LINE句]
[BLANK SCREEN句]
[BLANK WHEN ZERO句]
[BLINK句]
[COLUMN NUMBER句]
[ERASE句]
[FOREGROUND-COLOR句]
[FULL句]
[HIGHLIGHT句]
[LOWLIGHT句]
[JUSTIFIED句]
[LINE NUMBER句]
[PICTURE句]

[REQUIRED句]

[REVERSE-VIDEO句]

[SECURE句]

[SIGN句]

[UNDERLINE句]

[USAGE句].

構文規則

1. 画面データ記述項は、画面節にだけ書くことができます。
2. レベル番号は、01～49のいずれかでなければなりません。
3. データ名-1またはFILLERを書く場合、レベル番号の直後に書かなければなりません。その他の句は、任意の順序で書くことができます。
4. データ名-1およびFILLERを省略した場合、FILLERを書いたものとみなされます。
5. レベル番号が01の場合、データ名-1を書かなければなりません。
6. 集団画面項目は、集団画面項目または基本画面項目を従属することができます。集団画面項目は、書き方1を使って定義しなければなりません。基本画面項目は、書き方2または書き方3を使って定義しなければなりません。
7. 書き方2および書き方3では、BELL句、BLANK LINE句、BLANK SCREEN句、COLUMN NUMBER句、LINE NUMBER句のうちの少なくとも1つを書かなければなりません。
8. 集団画面項目に指定した句を、集団画面項目に従属する画面項目に指定することができます。その場合、階層構造中で最下位レベルの画面項目に指定した句が有効になります。
9. 画面項目は、スクリーン操作機能のACCEPT文またはDISPLAY文にだけ書くことができます。

一般規則

1. 書き方3の入力項目、出力項目、または更新項目の区別は、PICTURE句で指定します。
2. 集団画面項目にSECURE句を指定した場合、SECURE句は集団画面項目に従属するすべての入力項目に適用されます。
3. 集団画面項目に以下の句を指定した場合、その句は集団画面項目に従属するすべての入力項目および更新項目に適用されます。
   - AUTO句
   - BACKGROUND-COLOR句
   - BLANK SCREEN句
   - FOREGROUND-COLOR句
   - FULL句
   - REQUIRED句
4. LINE NUMBER句とCOLUMN NUMBER句で、画面上で領域が重なるような指定をしてはいけません。また、物理画面の制限を超える指定をしてはいけません。

5.5.1 AUTO句

入力および更新項目へカーソルの移動の自動化を行います。

【書き方】

AUTO

構文規則

AUTO句は、定数項目以外の基本画面項目または集団画面項目に指定することができます。

一般規則

1. 集団画面項目にAUTO句を指定した場合、AUTO句は、集団画面項目に従属するすべての入力項目および更新項目に適用されます。
2. 集団画面項目に対するACCEPT文の実行中、AUTO句が有効な入力項目または更新項目に対して最後の文字を入力すると、カーソルは自動的に次の入力項目または更新項目に移動します。
3. AUTO句は、DISPLAY文を実行するときは意味を持ちません。

5.5.2 BACKGROUND-COLOR句

画面項目の背景色を指定します。

【書き方】

BACKGROUND-COLOR IS 整数-1

構文規則

1. BACKGROUND-COLOR句は、任意の画面項目に指定することができます。
2. 整数-1の値は、0～7でなければなりません。

一般規則

1. 集団画面項目にBACKGROUND-COLOR句を指定した場合、BACKGROUND-COLOR句は、集団画面項目に従属するすべての基本画面項目に適用されます。
2. BACKGROUND-COLOR句は、カラー画面で実行する場合だけ意味を持ちます。
3. 整数-1には、画面項目の背景色を示す値を指定します。整数-1の値に対応する背景色を、下表に示します。

整数-1	背景色
0	黒
1	青
2	緑
3	シアン
4	赤
5	マゼンダ
6	茶 または 黄
7	白

4. BACKGROUND-COLOR句を省略した場合、背景色は黒になります。
5. BACKGROUND-COLOR句はDISPLAY文およびACCEPT文を実行するときに意味を持ちます。BACKGROUND-COLOR句が有効な画面項目は、整数-1に指定した色で表示されます。
6. BLANK SCREEN句とBACKGROUND-COLOR句の両方が有効な画面項目をDISPLAY文で表示すると、背景色の省略値がBACKGROUND-COLOR句で指定した色に変わります。

5.5.3 BELL句

オーディオトーンを鳴らすことを指定します。

【書き方】

BELL

構文規則

BELL句は、基本画面項目にだけ指定することができます。

一般規則

BELL句を指定した画面項目がDISPLAY文の実行によって表示されるとき、オーディオトーンが鳴ります。

5.5.4 BLANK LINE句

画面項目を表示する前に画面行を消去することを指定します。

【書き方】

BLANK LINE

構文規則

BLANK LINE句は、基本画面項目にだけ指定することができます。

一般規則

1. BLANK LINE句を指定した画面項目がDISPLAY文の実行によって表示されるとき、画面項目が表示される前に、BLANK LINE句を指定した画面項目が表示される行の最左端から行の最右端までが消去されます。
2. BLANK LINE句は、ACCEPT文を実行するときは意味を持ちません。

5.5.5 BLANK SCREEN句

画面項目を表示する前に画面全体を消去することを指定します。

【書き方】

BLANK SCREEN

構文規則

BLANK SCREEN句は、任意の画面項目に指定することができます。

一般規則

1. 集団画面項目にBLANK SCREEN句を指定した場合、BLANK SCREEN句は、集団画面項目に従属するすべての基本画面項目に適用されます。
2. BLANK SCREEN句が有効な画面項目がDISPLAY文によって表示されるとき、画面項目が表示される前に画面全体が消去されます。このとき、カーソルは第1行の第1列に設定されます。
3. BLANK SCREEN句とBACKGROUND-COLOR句を組み合わせて使うと、背景色の省略値を変更することができます。背景色の省略値については、“5.5.2 BACKGROUND-COLOR句”を参照してください。
4. BLANK SCREEN句とFOREGROUND-COLOR句を組み合わせて使用すると前景色の省略値を変更することができます。前景色の省略値については、“5.5.10 FOREGROUND-COLOR句”を参照してください。
5. BLANK SCREEN句は、ACCEPT文を実行するときは意味を持ちません。

5.5.6 BLANK WHEN ZERO句

画面項目の値がゼロのとき、空白を表示することを指定します。

【書き方】

BLANK WHEN { ZERO / ZEROS / ZEROES }

構文規則

1. BLANK WHEN ZERO句は、項類が数字または数字編集の基本画面項目にだけ指定することができます。
2. 数字画面項目にBLANK WHEN ZERO句を指定する場合、数字画面項目の用途は表示用でなければなりません。
3. ZERO、ZEROS、およびZEROESは同義語です。

一般規則

1. BLANK WHEN ZERO句は、画面項目の値がゼロのときに、その値を空白に置き換えて表示することを指定します。
2. 数字画面項目にBLANK WHEN ZERO句を指定した場合、その項目の項類は数字編集であるとみなされます。
3. 入力項目にBLANK WHEN ZERO句を指定しても、その指定は無視されます。
4. BLANK WHEN ZERO句は、ACCEPT文を実行するときは意味を持ちません。

5.5.7 BLINK句

画面項目の内容を点滅させることを指定します。

【書き方】

BLINK

構文規則

BLINK句は、基本画面項目にだけ指定することができます。

一般規則

1. BLINK句を指定した画面項目がDISPLAY文の実行によって表示されるとき、その画面項目に対応する画面上の領域が点滅します。
2. BLINK句は、ACCEPT文を実行するときには意味を持ちません。

5.5.8 COLUMN NUMBER句

画面項目を配置する列を指定します。

【書き方】

COLUMN NUMBER IS [PLUS] { 一意名-1 / 整数-1 }

構文規則

1. COLUMN NUMBER句は、基本画面項目にだけ指定することができます。
2. 一意名-1は、符号なし整数項目でなければなりません。

一般規則

1. COLUMN NUMBER句は、画面項目を表示するときまたは画面項目を入力するとき、画面項目の表示行上の文字位置を指定します。
2. PLUSを省略する場合、一意名-1または整数-1に、画面項目を配置する領域の列番号を設定します。一意名-1および整数-1に設定した値の意味は、ACCEPT文またはDISPLAY文の記述によって異なります。一意名-1および整数-1の値をnとすると、画面項目は以下の位置に配置されます。
   - DISPLAY文またはACCEPT文でAT COLUMN NUMBER指定を省略した場合、その文で操作する画面の第1列目は、物理画面の第1列目であるとみなされます。したがって、この句を指定したその画面項目は、物理画面の第n列目に配置されます。
   - DISPLAY文またはACCEPT文にAT COLUMN NUMBER指定を書いた場合、AT COLUMN NUMBER指定に指定した列番号の値をxとすると、その文で操作する画面の第1列目は、物理画面上の第x列目であるとみなされます。したがって、この句を指定した画面項目は、物理画面の第n＋x列目に配置されます。
3. PLUSを書く場合、一意名-1または整数-1に、この句を指定した画面項目に先行する画面項目の、右端からの相対列番号を設定します。設定した値は、先行する画面項目をACCEPT文またはDISPLAY文で操作するかどうかに関係なく、先行する画面項目の右端からの相対列番号として使われます。
4. PLUS付きのCOLUMN NUMBER句は、画面項目の最初に指定することはできません。
5. COLUMN NUMBER句を省略した画面項目には、以下の句を指定したものとみなされます。
   - その画面項目にLINE NUMBER句を指定した場合、“COLUMN NUMBER 1”を指定したものとみなされます。
   - その画面項目にLINE NUMBER句を指定しなかった場合、その画面項目に先行する画面項目が存在するときは、“COLUMN NUMBER PLUS 1”を指定したものとみなされます。

5.5.9 ERASE句

画面項目を表示する前に行の一部または画面の一部を消去することを指定します。

【書き方】

```
ERASE    { EOL }
         { EOS }
```

構文規則

ERASE句は、基本画面項目にだけ指定することができます。

一般規則

1. ERASE句を指定した画面項目がDISPLAY文の実行によって表示されるとき、画面項目が表示される前に、画面上の以下の領域が消去されます。
   - ERASE EOL句を指定した場合、その画面項目が表示される行の、その画面項目が表示される列からその行の最右端までが消去されます。
   - ERASE EOS句を指定した場合、その画面項目が表示される行の、その画面項目が表示される列から画面の終わりまでが消去されます。
2. DISPLAY文に以下の画面項目を指定した場合、その画面項目に対する画面上の領域だけが、DISPLAY文によって変更されます。画面上のその他の領域は、DISPLAY文を実行する前の状態のままです。
   - DISPLAY文に集団画面項目を指定し、集団画面項目にBLANK句(BLANK SCREEN句またはBLANK LINE句)とERASE句の両方が有効な基本画面項目が従属しない場合
   - DISPLAY文に、BLANK句(BLANK SCREEN句またはBLANK LINE句)とERASE句の両方が有効でない基本画面項目を指定した場合
3. ERASE句は、ACCEPT文を実行するときは意味を持ちません。

5.5.10 FOREGROUND-COLOR句

画面項目の前景色を指定します。

【書き方】

```
FOREGROUND-COLOR  IS  整数-1
```

構文規則

1. FOREGROUND-COLOR句は、任意の画面項目に指定することができます。
2. 整数-1の値は、0～7でなければなりません。

一般規則

1. 集団画面項目にFOREGROUND-COLOR句を指定した場合、FOREGROUND-COLOR句は、集団画面項目に従属するすべての基本画面項目に適用されます。
2. FOREGROUND-COLOR句は、カラー画面で実行する場合だけ意味を持ちます。
3. 整数-1には、画面項目の前景色を示す値を指定します。整数-1の値に対応する前景色を、下表に示します。

整数-1	背景色
0	黒
1	青
2	緑
3	シアン
4	赤
5	マゼンダ
6	茶 または 黄
7	白

4. FOREGROUND-COLOR句を省略した場合、前景色は白になります。
5. FOREGROUND-COLOR句は、DISPLAY文およびACCEPT文を実行するときに意味を持ちます。
   FOREGROUND-COLOR句が有効な画面項目は、整数-1に指定した色で表示されます。
6. BLANK SCREEN句とFOREGROUND-COLOR句の両方が有効な画面項目をDISPLAY文で表示すると、前景色の省略値がFOREGROUND-COLOR句で指定した色に変わります。

5.5.11 FULL句

画面項目のデータの入力を促すことを指定します。

【書き方】

FULL

構文規則

1. FULL句は、定数項目以外の基本画面項目または集団画面項目に指定することができます。
2. FULL句を基本画面項目に指定する場合、JUSTIFIED句を同時に指定することはできません。

一般規則

1. FULL句を集団画面項目に指定した場合、FULL句は、集団画面項目に従属するすべての入力項目および更新項目に適用されます。
2. ACCEPT文の実行によって、FULL句が有効な画面項目の入力操作を行うとき、その画面項目に以下のデータを入力するまで、入力キーの押下は無視され、その画面項目の先頭にカーソルが位置付けられます。
   - その画面項目が英数字画面項目または英数字編集画面項目の場合、画面項目の領域の最初と最後の文字位置に空白以外の文字を入力するまで、または画面項目の領域全体に空白を入力するまで。
   - その画面項目が日本語画面項目または日本語編集画面項目の場合、画面項目の領域の最初と最後の日本語文字位置に空白以外の日本語文字を入力するまで、または画面項目の領域全体に日本語の空白を入力するまで。
   - その画面項目が数字画面項目または数字編集画面項目の場合、画面項目のすべての数字桁位置に数字を入力するまで、または画面項目に値ゼロを入力するまで。ただし、ゼロ抑制を指定した数字編集画面項目の場合、有効な数字桁よりも上位のゼロは入力する必要はありません。
3. 入力操作を完了させるためにファンクションキーを押した場合、FULL句の規則は適用されません。
4. FULL句は、DISPLAY文を実行するときには意味を持ちません。

5.5.12 HIGHLIGHT句

画面項目を高輝度で表示することを指定します。

【書き方】

HIGHLIGHT

構文規則

HIGHLIGHT句は、基本画面項目にだけ指定することができます。

一般規則

1. HIGHLIGHT句は、DISPLAY文およびACCEPT文を実行するときに意味を持ちます。HIGHLIGHT句を指定した画面項目は、高輝度で表示されます。
2. HIGHLIGHT句を指定した画面項目にBACKGROUND-COLOR句が有効な場合、その画面項目の背景色も、高輝度で表示されます。同様に、HIGHLIGHT句を指定した画面項目にFOREGROUND-COLOR句が有効な場合、その画面項目の前景色も、高輝度で表示されます。

5.5.13 JUSTIFIED句

画面項目の右端に合わせて表示することを指定します。

【書き方】

{ JUSTIFIED / JUST } RIGHT

構文規則

1. JUSTIFIED句は、項類が英字、英数字または日本語の基本画面項目にだけ指定することができます。
2. JUSTIFIEDとJUSTは同義語です。

一般規則

1. JUSTIFIED句は、データ項目から画面項目へ転記するときの、桁よせの方法を指定します。画面項目にJUSTIFIED句を指定した場合、送出し側のデータが画面項目の右端に合わせて格納されます。送出し側データ項目と画面項目の大きさが異なる場合の転記の規則は以下のとおりです。
   - ー 送出し側データ項目の大きさが画面項目の大きさより大きい場合、送出し側データ項目の左端の文字が切り捨てられます。
   - ー 送出し側データ項目の大きさが画面項目の大きさより小さい場合、受取り側データ項目の左端の余りの部分に空白が埋められます。
2. JUSTIFIED句を省略した場合、データを基本項目に格納するときの標準桁よせの規則が適用されます。
3. 入力項目にJUSTIFIED句を指定しても、その指定は無視されます。
4. JUSTIFIED句は、ACCEPT文を実行するときは意味を持ちません。

5.5.14 LINE NUMBER句

画面項目を配置する行を指定します。

【書き方】

LINE NUMBER IS [PLUS] { 一意名-1 / 整数-1 }

構文規則

1. LINE NUMBER句は、基本画面項目にだけ指定することができます。
2. 一意名-1は、符号なし整数項目でなければなりません。

一般規則

1. LINE NUMBER句は、画面項目を表示するときまたは画面項目を入力するときの、画面項目の表示行位置を指定します。
2. PLUSを省略する場合、一意名-1または整数-1に、画面項目を配置する領域の行番号を設定します。一意名-1および整数-1に設定した値の意味は、ACCEPT文またはDISPLAY文の記述によって異なります。一意名-1および整数-1の値をnとすると、画面項目は以下の位置に配置されます。
   - DISPLAY文またはACCEPT文でAT LINE NUMBER指定を省略した場合、その文で操作する画面の第1行目は、物理画面の第1行目であるとみなされます。したがって、この句を指定した画面項目は、物理画面の第n行目に配置されます。
   - DISPLAY文またはACCEPT文にAT LINE NUMBER指定を書いた場合、AT LINE NUMBER指定に指定した列番号の値をxとすると、その文で操作する画面の第1行目は、物理画面上の第x行目であるとみなされます。したがって、この句を指定した画面項目は、物理画面の第n＋x行目に配置されます。
3. PLUSを書く場合、一意名-1または整数-1に、この句を指定した画面項目に先行する画面項目の行位置からの相対行番号を設定します。設定した値は、先行する画面項目をACCEPT文またはDISPLAY文で操作するかどうかに関係なく、先行する画面項目の行位置からの相対行番号として使われます。
4. PLUS付きのLINE NUMBER句は、画面項目の最初に指定することはできません。
5. LINE NUMBER句を省略した画面項目は、以下の位置に配置されます。
   - その画面項目に先行する画面項目が存在しない場合、物理画面の第1行に配置されます。
   - その画面項目に先行する画面項目が存在する場合、先行する画面項目と同じ行位置に配置されます。

5.5.15 LOWLIGHT句

画面項目を低輝度で表示することを指定します。

【書き方】

LOWLIGHT

構文規則

LOWLIGHT句は、基本画面項目にだけ指定することができます。

一般規則

1. LOWLIGHT句は、DISPLAY文およびACCEPT文を実行するときに意味を持ちます。LOWLIGHT句を指定した画面項目は、低輝度で表示されます。
2. 輝度のレベルが2つしかないシステムでは、通常の輝度と低輝度が同じ意味を持ちます。

5.5.16 PICTURE句

基本画面項目の項類および編集の形式を指定し、画面上の領域とデータのやりとりを行うためのデータ項目を、基本画面項目に対応付けます。

【書き方】

```
{ PICTURE } IS 文字列-1
{ PIC     }

{ USING 一意名-1                                                    }
{ FROM  { 一意名-2 } [TO 一意名-3] }
{       { 定数-1   }               }
```

構文規則

1. PICTURE句は、定数項目以外の基本画面項目にだけ指定することができます。PICTURE句とVALUE句を同時に指定することはできません。
2. 一意名-1、一意名-2および一意名-3は、以下のいずれかで定義したデータ項目でなければなりません。

   ー ファイル節

   ー 作業場所節

   ー 連絡節

   ー 基底場所節

   一意名-2には、定数節で定義したデータ項目を指定することもできます。
3. PICTURE句には、USING指定だけ、FROM指定だけ、TO指定だけ、およびFROM指定とTO指定の両方のうちの、いずれかを書かなければなりません。
4. PICTURE句、USING指定、FROM指定およびTO指定は、任意の順で書くことができます。また、PICTURE句とUSING指定の間、PICTURE句とFROM指定の間、またはPICTURE句とTO指定の間に、別の句を書くこともできます。
5. PICTUREとPICは同義語です。

一般規則

1. 文字列-1の組合せにより、基本画面項目の項類を指定します。基本画面項目の項類には、英字、数字、英数字、英数字編集、数字編集、日本語および日本語編集の7種類があります。それぞれの項類を持つ基本画面項目を、「英字画面項目」、「数字画面項目」、「英数字画面項目」、「英数字編集画面項目」、「数字編集画面項目」、「日本語画面項目」および「日本語編集画面項目」といいます。文字列-1の規則は、データ記述項のPICTURE句の文字列の規則と同じです。“5.4.7 PICTURE句”を参照してください。ただし、数字画面項目を定義する場合、その項目の用途は表示用で、文字列-1に“S”を指定しなければなりません。
2. データをデータ項目から画面項目へ転記し、画面上に表示する場合、FROM指定を書きます。FROM指定だけを指定した画面項目を「出力項目」といいます。

3. 画面項目によって画面からデータを入力し、別のデータ項目に格納する場合、TO指定を書きます。TO指定だけを指定した項目を「入力項目」といいます。
4. データをデータ項目から画面項目へ転記し、画面上で更新して同じデータ項目に格納する場合、USING指定を書きます。USING指定を指定した画面項目を、「更新項目」といいます。
5. データをデータ項目から画面項目へ転記し、画面上で編集して別のデータ項目に格納する場合、FROM指定とTO指定の両方を書きます。FROM指定とTO指定の両方に同じデータ項目を指定した場合は、USING指定と同様に扱われます。
6. 一意名-1、一意名-2、一意名-3および定数-1の文字位置の個数は、関連する画面項目の文字位置の個数と同じである必要はありません。
7. 画面項目と、一意名-1、一意名-2、一意名-3または定数-1との間の転記は、転記の規則に従って行われます。
8. FROM指定は、DISPLAY文を実行するときだけ意味を持ちます。
9. TO指定は、ACCEPT文を実行するときだけ意味を持ちます。
10. USING指定は、ACCEPT文またはDISPLAY文を実行するとき意味を持ちます。

5.5.17 REQUIRED句

画面項目への入力が必須であることを指定します。

【書き方】

REQUIRED

構文規則

REQUIRED句は、定数項目以外の基本画面項目または集団画面項目に指定することができます。

一般規則

1. REQUIRED句を集団画面項目に指定した場合、REQUIRED句は、集団画面項目に従属するすべての入力項目と更新項目に適用されます。
2. ACCEPT文の実行によって、REQUIRED句が有効な画面項目の入力操作が行われるとき、その画面項目に以下のデータを入力するまで、入力キーの押下は無視され、その画面項目の先頭にカーソルが位置付けられます。
   - その画面項目が英数字画面項目または英数字編集画面項目の場合、空白以外の1文字以上の文字を入力するまで
   - その画面項目が日本語画面項目または日本語編集画面項目の場合、空白以外の1文字以上の日本語文字を入力するまで
   - その画面項目が数字画面項目または数字編集画面項目の場合、ゼロ以外の値を入力するまで
3. 入力操作を完了させるためにファンクションキーを押した場合、REQUIRED句の規則は適用されません。
4. REQUIRED句は、DISPLAY文を実行するときには意味を持ちません。

5.5.18 REVERSE-VIDEO句

画面項目の前景色と背景色を反転することを指定します。

【書き方】

REVERSE-VIDEO

構文規則

REVERSE-VIDEO句は、基本画面項目にだけ指定することができます。

一般規則

REVERSE-VIDEO句は、DISPLAY文およびACCEPT文を実行するときに意味を持ちます。REVERSE-VIDEO句を指定した画面項目は、前景色と背景色が反転されて表示されます。

5.5.19 SECURE句

入力項目の表示を非表示状態にすることを指定します。

【書き方】

SECURE

構文規則

SECURE句は、入力項目または集団画面項目に指定することができます。

一般規則

1. SECURE句を集団画面項目に指定した場合、SECURE句は、集団画面項目に従属するすべての入力項目に適用されます。
2. ACCEPT文の実行によって、SECURE句が有効な画面項目の入力操作を行うとき、その画面項目に対して入力した文字は、画面上に表示されません。このとき、カーソルは、その画面項目の先頭に位置付けられたままです。

5.5.20 SIGN句

画面項目の演算符号の位置と表現形式を指定します。

【書き方】

[SIGN IS] { LEADING / TRAILING } [SEPARATE CHARACTER]

構文規則

SIGN句は、定数項目以外の基本画面項目または集団画面項目に指定することができます。その他の構文規則は、データ記述項のSIGN句の構文規則と同じです。“5.4.11 SIGN句”を参照してください。

一般規則

一般規則は、データ記述項のSIGN句の一般規則と同じです。“5.4.11 SIGN句”を参照してください。

5.5.21 UNDERLINE句

画面項目を下線付きで表示することを指定します。

【書き方】

UNDERLINE

構文規則

UNDERLINE句は、基本画面項目にだけ指定することができます。

一般規則

UNDERLINE句は、DISPLAY文およびACCEPT文を実行するときに意味を持ちます。UNDERLINE句を指定した画面項目は、下線付きで表示されます。

5.5.22 USAGE句

画面項目の用途を指定します。

【書き方】

[USAGE IS] DISPLAY

構文規則

1. USAGE句は、定数項目以外の基本画面項目または集団画面項目に指定することができます。
2. USAGE句を集団画面項目に指定した場合、集団画面項目に従属する基本画面項目または集団画面項目にも、USAGE句を指定することができます。

一般規則

1. 集団画面項目にUSAGE句を指定した場合、USAGE句は、集団画面項目に従属するすべての基本項目に適用されます。USAGE句が有効な基本画面項目を、「用途が表示用の画面項目」といいます。
2. USAGE句は、画面上のデータ項目の表現形式を指定します。USAGE IS DISPLAY句は、データ項目の表現形式が標準データ形式であることを示します。
3. 集団画面項目にUSAGE句を指定しなかった場合、集団画面項目に従属するすべての基本画面項目の用途は表示用であるとみなされます。また、USAGE句を省略した基本画面項目の用途は、表示用であるとみなされます。
4. その他の一般規則は、データ記述項のUSAGE句の一般規則と同じです。“5.4.15 USAGE句”を参照してください。

5.5.23 VALUE句

定数項目の値を指定します。

【書き方】

VALUE IS 定数-1

構文規則

1. VALUE句は、定数項目にだけ指定することができます。VALUE句とPICTURE句を同時に指定することはできません。
2. 定数-1は、文字定数、16進文字定数または日本語定数でなければなりません。

一般規則

定数-1に、定数項目の値、すなわち画面に表示する値を指定します。

5.6 報告書記述項

報告書記述項では、報告書の名前、報告書の各ページの体裁および制御データ項目を定義します。報告書記述項は、報告書節にだけ書くことができます。

注意

[WinIPF][Winx64][LinuxIPF][Linux64][.NET]では、報告書記述項を記述できません。

【書き方】

```
RD  報告書名-1
    [CODE句]
    [CONTROL句]
    [PAGE句].
```

構文規則

1. 報告書記述項の最初にレベル指示語RDを書き、その後に報告書名-1を書かなければなりません。
2. 報告書名-1は、ただ1つのREPORT句に書かなければなりません。
3. 報告書名-1の後に書く句の順序は任意です。
4. 行カウンタ、ページカウンタ、および報告書節で定義するデータ名に対する最高位の修飾語は、報告書名-1です。

特殊レジスタ

報告書記述項を書くと、特殊レジスタPAGE-COUNTERおよびLINE-COUNTERが自動的に作成されます。特殊レジスタPAGE-COUNTERを「ページカウンタ」といい、LINE-COUNTERを「行カウンタ」といいます。

ページカウンタの規則

1. ページカウンタは、以下の場所だけに書くことができます。
   - 報告書節の報告書集団記述項のSOURCE句
   - 手続き部で、整数値を持つデータ項目が書けるところ
2. 1つのプログラムに2つ以上の報告書記述項を書いた場合、ページカウンタを以下のように書かなければなりません。
   - 手続き部に書く場合、ページカウンタは報告書名で修飾しなければなりません。

ー SOURCE句では、ページカウンタを修飾しないで書くことができます。修飾しなかった場合、SOURCE句を書いた報告書集団に関連する報告書名で、暗に修飾されます。別の報告書のページカウンタを参照する場合、報告書記述項に書いた報告書名で、明に修飾しなければなりません。

3. ページカウンタには、報告書作成管理システムによって、以下のように値が設定されます。

   ー INITIATE文を実行すると、ページカウンタに1が設定されます。

   ー 報告書作成管理システムが改ページをするごとに、ページカウンタの値が1ずつ加算されます。

4. 手続き部の文によって、ページカウンタの値を変更することもできます。

行カウンタの規則

1. 行カウンタは、以下の場所だけに書くことができます。

   ー 報告書節の報告書集団記述項のSOURCE句

   ー 手続き部で、整数値を持つデータ項目が書けるところ

2. 行カウンタに値を設定することはできません。行カウンタの値の設定は、報告書管理システムが行います。

3. 1つのプログラムに2つ以上の報告書記述項を書いた場合、行カウンタを以下のように書かなければなりません。

   ー 手続き部に書く場合、行カウンタは報告書名で修飾しなければなりません。

   ー SOURCE句では、行カウンタを修飾しないで書くことができます。修飾しなかった場合、その句を書いた報告書集団に関連する報告書記述項の報告書名で、暗に修飾されます。別の報告書の行カウンタを参照する場合、その報告書名で明に修飾しなければなりません。

4. 行カウンタには、報告書作成管理システムによって、以下のように値が設定されます。

   ー INITIATE文を実行すると、行カウンタに0が設定されます。

   ー 報告書作成管理システムが改ページをするごとに、行カウンタに0が設定されます。

5. 印字項目を持たない報告集団を処理しても、または、SUPPRESS文で印字の抑制される集団を処理しても、行カウンタの値は変更されません。

6. 各印字行が表示されるときに、行カウンタの値はその印字行が表示される行位置を表します。報告集団の表示後の行カウンタの値は、各報告集団の表示規則によって決まります。報告集団の表示規則については、“5.8 報告集団の表示規則”を参照してください。

5.6.1 CODE句

報告書の印字行に付ける定数を指定します。

【書き方】

CODE 定数-1

構文規則

1. 定数-1は、2文字の文字定数でなければなりません。

2. あるファイル中のどれかの報告書にCODE句を指定したときには、そのファイル中の他のすべての報告書に対しても指定しなければなりません。

一般規則

1. CODE句を指定すると、定数-1の2文字が報告書の各レコードの左端に自動的に付きます。

2. この機能を使うと、数種の報告書を同時に1つの出力装置に記録しておき、後でこれらを選択して個々の報告書を作成することができます。報告書を直接ラインプリンタ装置に書き出すときには、この句を書くことはできません。

5.6.2 CONTROL句

報告書の制御階層を指定します。

【書き方】

```
{ CONTROL IS    } { {データ名-1} …          }
{ CONTROLS ARE  } { FINAL [データ名-1] …    }
```

構文規則

1. データ名-1は、報告書節で定義されたものであってはなりません。データ名-1は、修飾することもできます。
2. データ名-1の繰返しは、同じデータ項目であってはいけません。
3. データ名-1には、可変反復データ項目が属していてはいけません。

一般規則

1. データ名-1およびFINALは、制御階層を指定します。FINALを最も高い制御とし、データ名-1を大制御データ項目、データ名-1の次の繰返しを中制御データ項目などとします。データ名-1の最後の繰返しは、小制御データ項目とします。
2. データ名-1に指定したデータ項目を「制御データ項目」といいます。制御データ項目の値が変化することを「制御切れ」といいます。
   ある報告書に対して、GENERATE文が最初に実行されると、報告書作成管理システムは、その報告書に関するすべての制御データ項目の値を保存します。その報告書に対する以後のGENERATE文を実行するとき、報告書作成管理システムは、制御データ項目の値の変化を検査します。制御データ項目の値が1つでも変化していると、制御切れが起こります。制御切れは、値の変化が記録された最も高いレベルと結びつけられます。
3. 報告書作成管理システムは、制御データ項目の内容と、同じ報告書に対する直前のGENERATE文の実行のときに保存されている制御データ項目の前の内容とを比較することにより、制御切れを検査します。報告書作成管理システムは、以下に示す不等比較検査を行います。
   a. 制御データ項目が数字項目のとき、検査は2つの数字作用対象の比較です。
   b. 制御データ項目が指標データ項目のとき、検査は2つの指標データ項目の比較です。
   c. 制御データ項目が日本語項目または日本語編集項目のとき、検査は2つの日本語作用対象の比較です。
   d. 制御データ項目が前記a.、b.またはc.のいずれでもないとき、検査は2つの文字作用対象の比較です。
4. 報告書で最も包括的な制御集団に対応する制御データ項目がないとき、FINALを指定します。

5.6.3 PAGE句

ページの長さと報告集団が表示される縦の範囲を指定します。

【書き方】

```
PAGE [ LIMIT IS   ] 整数-1 [ LINE  ]
     [ LIMITS ARE ]        [ LINES ]
  [HEADING 整数-2] [FIRST DETAIL 整数-3]
```

[LAST DETAIL 整数-4] [FOOTING 整数-5]

構文規則

1. HEADING(頭書き限界)指定、FIRST DETAIL(明細報告集団の上限)指定、LAST DETAIL(明細報告集団の下限)指定およびFOOTING(脚書き限界)指定を書く順序は任意です。
2. 整数-1は、有効数字で3桁を超えてはいけません。
3. 整数-2は、1以上でなければなりません。
4. 整数-3は、整数-2以上でなければなりません。
5. 整数-4は、整数-3以上でなければなりません。
6. 整数-5は、整数-4以上でなければなりません。
7. 整数-1は、整数-5以上でなければなりません。
8. PAGE句を書いたとき、報告集団が表示されるページの縦の範囲を以下に示します。
   - 報告書頭書き報告集団が、1ページに単独で表示されるときには、報告書頭書き報告集団は、整数-2から整数-1までの行位置の範囲に表示されます。報告書頭書き報告集団が、1ページに単独で表示されないときは、報告書頭書き報告集団は、整数-2から整数-3の1行上までの行位置の範囲に表示されます。
   - ページ頭書き報告集団が指定されていれば、ページ頭書き報告集団は、整数-2から整数-3の1行上までの行位置の範囲に表示されます。
   - 制御頭書き報告集団または明細報告集団が指定されていれば、制御頭書き報告集団または明細報告集団は、整数-3から整数-4までの行位置の範囲に表示されます。
   - 制御脚書き報告集団が指定されていれば、制御脚書き報告集団は、整数-3から整数-5までの行位置の範囲に表示されます。
   - ページ脚書き報告集団が指定されていれば、ページ脚書き報告集団は、整数-5に1を加えた値から、整数-1までの行位置の範囲に表示されます。
   - 報告書脚書き報告集団が、1ページに単独で表示されるときは、報告書脚書き報告集団は、整数-2から整数-1までの行位置の範囲に表示されます。報告書脚書き報告集団が、1ページに単独で表示されないときは、報告書脚書き報告集団は、整数-5に1を加えた値から整数-1までの行位置の範囲に表示されます。
9. すべての報告集団は、1つのページ内に納まるように記述しなければなりません。報告書作成管理システムは、複数行からなる報告集団をページの境界にまたがって表示することはありません。

一般規則

1. 報告書のページの縦の体裁は、PAGE句で指定された整数の値で決定されます。各整数の意味は、以下のとおりです。
   - 整数-1は、各ページの有効な行数を指定して、報告書の1ページの大きさを決めます。
   - HEADINGの整数-2は、報告書頭書き報告集団またはページ頭書き報告集団を表示できる最初の行位置を指定します。
   - FIRST DETAILの整数-3は、本体集団を表示できる最初の行位置を指定します。報告書頭書き報告集団(NEXT GROUP NEXT PAGEのない)は、整数-3で指定された行位置またはその行位置を超えて表示されることはありません。
   - LAST DETAILの整数-4は、制御頭書き報告集団または明細報告集団を表示できる最後の行位置を指定します。
   - FOOTINGの整数-5は、制御脚書き報告集団を表示できる最後の行位置を指定します。報告書脚書き報告集団(LINE 整数-1 NEXT PAGEのない)およびページ脚書き報告集団は、整数-5で指定された行位置より後ろに表示されます。

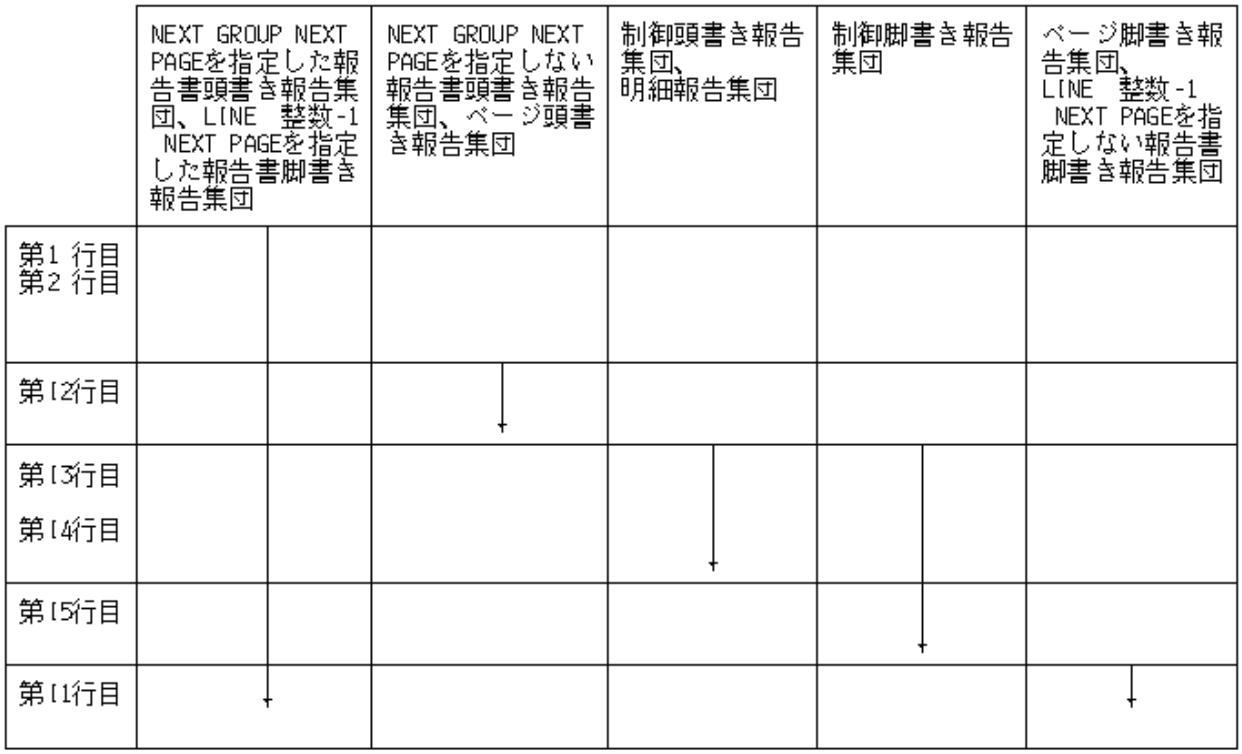
2. PAGE句が指定され、かつその中の指定が省略されたときは、以下に示す値が仮定されます。
   - HEADING指定を省略すると、整数-2は1とみなします。
   - FIRST DETAIL指定を省略すると、整数-3は整数-2と同じとみなします。
   - LAST DETAIL指定とFOOTING指定の両方を省略すると、整数-4および整数-5も、整数-1と同じとみなします。
   - FOOTING指定が指定され、LAST DETAIL指定が省略されると、整数-4は整数-5と同じとみなします。
   - LAST DETAIL指定が指定されFOOTING指定が省略されると、整数-5は整数-4と同じとみなします。
3. PAGE句を省略すると、報告書は長さの決まらない1つのページからなります。
4. 報告集団の表示規則は、“5.8 報告集団の表示規則”を参照してください。

ページ領域

PAGE句で設定されるページ領域を、下表に示します。

領域に表示される報告集団	領域の最初の行位置	領域の最後の行位置
NEXT GROUP NEXT PAGEを指定した報告書頭書き報告集団 **LINE 整数-1 NEXT PAGEを指定した報告書脚書き報告集団**	整数-2	整数-1
NEXT GROUP NEXT PAGEを指定しない報告書頭書き報告集団 **ページ頭書き報告集団**	整数-2	(整数-3)－1
制御頭書き報告集団 **明細報告集団**	整数-3	整数-4
制御脚書き報告集団	整数-3	整数-5
ページ脚書き報告集団 **LINE 整数-1 NEXT PAGEを指定しない報告書脚書き報告集団**	(整数-5)＋1	整数-1

これを図示すると以下のようになります。



[1]: PAGE句の整数-1の値
[2]: HEADING 指定の整数-2の値
[3]: FIRST DETAIL指定の整数-3の値
[4]: LAST DETAIL 指定の整数-4の値
[5]: FOOTING 指定の整数-5の値

5.7 報告集団記述項

報告集団記述項では、報告書に従属する報告集団の性質、および報告集団内の個々の項目の性質を定義します。報告集団記述項は、報告書節にだけ書くことができます。

報告集団記述項は、報告書記述項に続いて書きます。報告集団記述項は、1～3段階の階層で構成します。

注意

[WinIPF][Winx64][LinuxIPF][Linux64][.NET]では、報告書集団記述項を記述できません。

【書き方1】1番目の階層の報告集団記述項

01 [データ名-1]
[LINE NUMBER句]
[NEXT GROUP句]
TYPE句
[USAGE句].

【書き方2】2番目の階層の報告集団記述項

レベル番号 [データ名-1]

[LINE NUMBER句]
[USAGE句].

【書き方3】3番目の階層(印字項目)の報告集団記述項

レベル番号 [データ名-1]
[BLANK WHEN ZERO句]
[COLUMN NUMBER句]
[GROUP INDICATE句]
[JUSTIFIED句]
[LINE NUMBER句]
PICTURE句
[SIGN句]
[USAGE句]
{ SOURCE句 / SUM句 / VALUE句 } .

構文規則

1. 報告集団記述項の先頭には、データ名-1を書く場合、データ名-1はレベル番号の直後に書かなければなりません。データ名-1以外の句の順序は、任意です。
2. 書き方2のレベル番号は02～48、書き方3のレベル番号は02～49の整数でなければなりません。
3. 報告集団記述項は、1～3の階層で構成します。階層の付け方の規則を、以下に示します。
   - 報告集団記述項の最初の記述項は、書き方1でなければなりません。
   - 書き方1の記述項のすぐ下位に、書き方2または書き方3の記述項を書くことができます。
   - 書き方2の記述項を書いた場合、すぐ下位に少なくとも1つの書き方3の記述項を書かなければなりません。
   - 書き方3の記述項は、基本項目を定義するものでなければなりません。
4. 書き方1のデータ名-1は、以下の場合にだけ書きます。
   - 明細報告集団をGENERATE文で参照する場合
   - 明細報告集団をSUM句のUPON指定で参照する場合
   - 報告集団をUSE BEFORE REPORTING完結文で参照する場合
   - 制御脚書き報告集団の名前を、合計カウンタの修飾に使う場合

   データ名-1を書いた場合、データ名-1はGENERATE文、SUM句のUPON指定、USE BEFORE REPORTING文および合計カウンタの修飾語でだけ、参照することができます。
5. 書き方2の記述項には、少なくとも1つの句を書かなければなりません。
6. 書き方2でデータ名-1を書いた場合、データ名-1は合計カウンタを修飾するためにだけ使うことができます。
7. 報告書節でのUSAGE句は、印字項目の用途を宣言するためにだけ書きます。
   - 書き方3の記述項にUSAGE句を書く場合、その記述項は印字項目を定義するものでなければなりません。
   - 書き方1または書き方2の記述項にUSAGE句を書く場合、それに従属する少なくとも1つの記述項で、印字項目を定義しなければなりません。

8. LINE NUMBER句を指定した記述項は、LINE NUMBER句を指定した記述項を従属してはいけません。
9. 書き方3の記述項で定義した項目を「印字項目」といいます。書き方3では、以下の規則に従わなければなりません。
   - GROUP INDICATE句は、明細報告集団にだけ書くことができます。
   - SUM句は、制御脚書き報告集団にだけ書くことができます。
   - COLUMN NUMBER句を書き、LINE NUMBER句を省略した記述項は、LINE NUMBER句を書いた記述項に従属しなければなりません。

     PRINTING POSITION句を書き、LINE NUMBER句を省略した記述項は、LINE NUMBER句を書いた記述項に従属しなければなりません。
   - データ名-1は、どの記述項にも書くことができます。ただし、その記述項が合計カウンタを定義する場合にだけ参照することができます。
   - VALUE句を指定した記述項には、COLUMN NUMBER句またはPRINTING POSITION句のどちらかを書かなければなりません。
   - 1つの報告集団に、COLUMN NUMBER句を書いた記述項とPRINTING POSITION句を書いた記述項を混在させてはいけません。
10. 書き方3の記述項で許される句の組合せには、下表に示す5種類があります。

句	句の組合せ				
	その1	その2	その3	その4	その5
PICTURE 句	必	必	必	必	必
COLUMN NUMBER 句	―	必	可	可	必
PRINTING POSITION 句	―	必	可	可	必
SOURCE句	―	―	必	必	―
SUM 句	必	必	―	―	―
VALUE 句	―	―	―	―	必
JUST句	―	―	可	―	可
BLANK WHEN ZERO 句	―	可	―	可	―
GROUP INDICATE句	―	―	可	可	可
USAGE 句	―	可	可	可	可
SIGN句	可	可	可	可	可
LINE句	可	可	可	可	可

必 ： その句を必ず書かなければならないことを示します。
可 ： その句を書いてもかまいませんが、必ずしも必要でないことを示します。
― ： その句を書いてはならないことを示します。

一般規則

1. 書き方1は、報告集団記述項です。報告集団は、この記述項と、それに従属する記述項全体によって定義されます。
2. 報告集団記述項のBLANK WHEN ZERO句、JUSTIFIED句およびPICTURE句は、それぞれ、データ記述項のBLANK WHEN ZERO句、JUSTIFIED句およびPICTURE句と同じです。これらの句の規則については、“5.4 データ記述項”を参照してください。
3. レベル番号が01の報告集団記述項を2つ以上書いた場合、それらには異なる領域が割り当てられます。

5.7.1 COLUMN NUMBER句

印字行の上での印字位置を指定します。

【書き方】

COLUMN NUMBER IS 整数-1

構文規則

1. COLUMN NUMBER句は、報告集団中の基本項目にだけ書くことができます。COLUMN NUMBER句を書くときには、LINE NUMBER句を含む記述項またはそれに従属する記述項に書かなければなりません。
2. 1つの印字行の中で印字項目は、その行の左のものから右のものへ順に書かなければなりません。各印字項目の印字位置は、重ならないようにしなければなりません。

一般規則

1. COLUMN NUMBER句は、SOURCE句またはVALUE句の目的語またはSUM句で定義される合計カウンタを、印字行に表示することを示します。COLUMN NUMBER句がないと、その項目は印字行に表示されません。
2. 整数-1は、印字項目の最左端の印字位置を示します。
3. 報告書作成管理システムは、印字項目によって占められていない印字行のすべての印字位置に空白を補います。
4. 印字行の最左端の印字位置は、1とします。

5.7.2 GROUP INDICATE句

制御切れまたは改ページの明細報告集団の表示を指示します。

【書き方】

GROUP INDICATE

構文規則

GROUP INDICATE句は、印字項目を定義する明細報告集団にだけ書くことができます。

一般規則

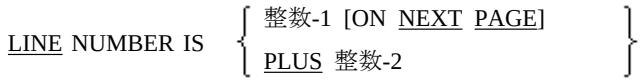
1. GROUP INDICATE句は、制御または改ページの後で、最初に出てくる明細報告集団のときだけ、この印字項目を表示することを指示します。
2. GROUP INDICATE句を指定すると、以下の場合を除いて、SOURCE句やVALUE句は無視され、空白が補われます。
   - ー その報告書の明細報告集団を最初に表示するとき
   - ー 改ページの後で明細報告集団を最初に表示するとき
   - ー 制御切れの後で明細報告集団を最初に表示するとき

3. 報告書記述項にPAGE句およびCONTROL句がない場合、GROUP INDICATE句を指定された印字項目が表示されるのは、INITIATE文の実行後、最初にこの明細報告集団が表示されるときです。その後、SOURCE句またはVALUE句をもつその項目には、空白が補われます。

5.7.3 LINE NUMBER句

報告集団の縦の位置を指定します。

【書き方】



構文規則

1. 整数-1および整数-2は、有効数字で3桁を超えてはなりません。整数-1および整数-2は、この報告集団の型に対してPAGE句で定義したページの縦の範囲の外に、印字項目を表示しようとするものであってはいけません。
   整数-2には、ゼロを指定することもできます。
2. LINE NUMBER句を含む記述項は、LINE NUMBER句を含む記述項に従属してはいけません。
3. 1つの報告集団記述項において、絶対的なLINE NUMBER句は、相対的なLINE NUMBER句よりも前になければなりません。
4. 1つの報告集団記述項において、一連の絶対的なLINE NUMBER句の整数は、昇順でなければなりません。連続している必要はありません。
5. PAGE句を省略したとき、その報告書の報告集団記述項では、相対的なLINE NUMBER句だけを指定することができます。
6. 1つの報告集団記述項にNEXT PAGE指定は一度だけ書くことができます。このとき、最初のLINE NUMBER句に書かなければなりません。
7. NEXT PAGE指定を伴うLINE NUMBER句は、本体報告集団と報告書脚書き報告集団の記述項にだけ書くことができます。
8. 印字項目を定義する記述項は、LINE NUMBER句を含むか、LINE NUMBER句を含む記述項に従属していなければなりません。
9. ページ脚書き報告集団の最初のLINE NUMBER句は、絶対的なLINE NUMBER句でなければなりません。

一般規則

1. 報告集団の印字行の行位置を指定するために、LINE NUMBER句を書かなければなりません。
2. 報告書作成管理システムは、そのLINE NUMBER句によって指定された印字行を表示する前に、そのLINE NUMBER句によって指定された縦の位置付けを行います。
3. 整数-1には、絶対的な行位置を指定します。絶対的な行位置は、印字行が表示される行位置を示します。
4. 整数-2には、相対的な行位置を指定します。相対的なLINE NUMBER句が報告集団記述項の最初のLINE NUMBER句でないとき、印字行が表示される行位置は、その報告集団中の直前の印字行を表示した行位置と相対的なLINE NUMBER句の整数-2との和の位置になります。整数-2がゼロのときは、印字行は前の印字行と同じ行に表示されます。
   相対的なLINE NUMBER句が、その報告集団記述項の最初のLINE NUMBER句のとき、印字行が表示される行位置は、“5.8 報告集団の表示規則”に従って決まります。
5. NEXT PAGE指定を書くと、その報告集団は、新しいページのその句で指定された行位置に表示されます。

5.7.4 NEXT GROUP句

報告集団の最後の行を表示した後に行う行送りを指示します。

【書き方】

```
NEXT GROUP IS { 整数-1
              | PLUS 整数-2
              | NEXT PAGE }
```

構文規則

1. LINE NUMBER句が指定されていない報告集団記述項に、NEXT GROUP句を指定することはできません。
2. 整数-1と整数-2は、有効数字で3桁を超えてはいけません。
3. 報告書記述項でPAGE句を省略したとき、その報告書の報告集団記述項では、相対的なNEXT GROUP句だけを指定することができます。
4. NEXT GROUP句のNEXT PAGE指定は、ページ脚書き報告集団に指定することはできません。
5. NEXT GROUP句は、報告書脚書き報告集団またはページ頭書き報告集団に指定することはできません。

一般規則

1. NEXT GROUP句によって指定される行送りは、その句が指定された報告集団の表示の後に行われます。
2. 報告書作成管理システムは、NEXT GROUP句で指定される行送りの情報を、TYPE句、PAGE句の情報および行カウンタの値と組み合わせて、行カウンタの新しい値を決定します。
3. 制御切れが検出された最高のレベル以外の制御脚書き報告集団に書かれたNEXT GROUP句は、無視されます。
4. 本体集団のNEXT GROUP句は、次の本体集団が表示される位置に影響を及ぼします。
   報告書頭書き報告集団のNEXT GROUP句は、ページ頭書き報告集団が表示される位置に影響を及ぼします。
   ページ脚書き報告集団のNEXT GROUP句は、報告書脚書き報告集団が表される位置に影響を及ぼします。

5.7.5 SIGN句

演算符号の位置と表現形式を指定します。

【書き方】

```
[SIGN IS] { LEADING  } SEPARATE CHARACTER
          { TRAILING }
```

構文規則

1. SIGN句は、PICTURE句の文字列に文字“S”を含んでいる数字項目にだけ指定することができます。
2. SIGN句が適用される数字項目の用途は、暗にまたは明に表示用でなければなりません。
3. 報告集団記述項にSIGN句を含めるときは、SEPARATE CHARACTER指定を書かなければなりません。

一般規則

1. SIGN句は、数字項目の符号の位置と表現形式を指定します。数字項目に書いたSIGN句は、その項目に対して適用されます。この場合、PICTURE句の文字列の“S”は符号の存在を示すだけで、表現形式や位置を示すものではありません。
2. PICTURE句の文字列に“S”のある数字項目で、SIGN句のない場合は、SIGN句にTRAILINGだけを書いたときと同じに扱われます。
3. 報告集団記述項中のSIGN句にはSEPARATE CHARACTER指定を書かなければならないので、符号は以下によります。
   - 演算符号は、LEADINGのとき左端、TRAILINGのとき右端の文字位置に付きます。この文字位置は、数字位置とは別とします。
   - PICTURE句の文字列の中の文字“S”は、標準データ形式におけるこの項目の大きさに数えます。
   - 正と負の演算符号は、標準データ形式のそれぞれ“＋”と“－”とします。
     SEPARATE CHARACTERを書いた場合、実行時に、“＋”または“－”がデータの中にないと、その実行結果は規定されません。
4. PICTURE句の文字列に“S”のある数字項目は、すべて符号付き数字項目となります。SIGN句のある項目に対して計算や比較のために変換が必要なときには、この変換は自動的に行われます。
5. SIGN句を指定したときのデータ項目の表現形式については、“5.4.11 SIGN句”および“5.4.15 USAGE句”を参照してください。

5.7.6 SOURCE句

印字項目に転記される値の送出し側データ項目を指定します。

【書き方】

SOURCE IS 一意名-1

構文規則

1. 一意名-1は、データ部のどの節で定義されたものであってもかまいませんが、報告書節の項目の場合には、以下のものだけが許されます。
   - ページカウンタ
   - 行カウンタ
   - そのSOURCE句と同じ報告書中の合計カウンタ
2. 一意名-1は、報告書作成管理システムが印字項目へ転記するために暗に行うMOVE文の送出し側データ項目を指定します。それは、MOVE文の送出し側項目に関する規則に従って定義しなければなりません。転記の規則については“6.3.5 転記の規則”を参照してください。
3. 一意名-1を部分参照する場合、部分参照子の最左端位置および長さは、データ名または定数でなければなりません。
4. 一意名-1を添字付けする場合、添字に演算子(＋または－)を書くことはできません。

一般規則

報告書作成管理システムは報告集団を表示する直前に、その報告集団の印字行を作成します。このとき、SOURCE句で指定された暗黙のMOVE文を実行します。報告集団を表示する時点については、“5.7.8 TYPE句”を参照してください。

5.7.7 SUM句

合計カウンタを作り、合計されるデータ項目の名前を指定します。

【書き方】

```
{SUM {一意名-1} … [UPON {データ名-1} … ]} …
  [ RESET ON { データ名-2 } ]
             { FINAL      }
```

構文規則

1. SUM句が現れる報告書集団記述項の主体であるデータ項目は、英字として定義されていてはなりません。一意名-1は、数字項目を参照しなければなりません。一意名-1が報告書節にある場合には、合計カウンタを参照しなければなりません。
   UPON指定を省略し、かつSUM句中の一意名が合計カウンタである場合には、その一意名はこのSUM句を含んでいる同じ制御脚書き報告集団か、または同じ報告書中のこれより低い制御階層に含まれる制御脚書き報告集団の、どちらかで定義されていなければなりません。
   UPON指定を記述した場合には、SUM句中の一意名は合計カウンタであってはなりません。
2. データ名-1は、SUM句が書かれた制御脚書き報告集団と同じ報告書中の明細報告集団の名前でなければなりません。データ名-1は、報告書名で修飾することができます。
3. SUM句は、制御脚書き報告集団にだけ書くことができます。
4. データ名-2は、報告書のCONTROL句に書いたデータ名の1つでなければなりません。データ名-2は、RESET指定が書かれた制御脚書き報告集団の制御階層より低いレベルの制御データ名であってはいけません。RESET指定にFINALを書く場合には、この報告書のCONTROL句にもFINALを書いておかなければなりません。
5. 合計カウンタに対して許される最も高いレベルの修飾語は、報告書名とします。

一般規則

1. SUM句は合計カウンタを設定します。合計カウンタは、コンパイラが生成する演算符号付きの数字データ項目です。合計カウンタの大きさと小数点位置は、SUM句が指定された報告集団記述項によって指定されたデータ項目の項類に依存します。それらは、以下のように決められます。
   - 関連するデータ項目が数字ならば、合計カウンタの大きさと小数点位置はそのデータ項目のそれと等しくなります。
   - 関連するデータ項目が数字編集ならば、合計カウンタの大きさはそのデータ項目桁位置の数に等しく、小数点位置は、その関連するデータ項目のそれと等しくなります。
   - 関連するデータ項目が英数字または英数字編集ならば、合計カウンタの大きさは、編集文字を除いたそのデータ項目の大きさまたは18文字のいずれか小さい方に等しく、合計カウンタは整数です。
2. 意名-1のデータ項目の値が、実行時に、報告書作成管理システムによって合計カウンタに加算されます。この加算は、ADD文の規則に従います。
3. 報告書の1つの基本項目の記述項中にSUM句を2つ以上書いても、設定されるカウンタはただ1つだけです。
4. SUM句を印字項目の記述項に書くと、その合計カウンタが源データ項目となり、報告書作成管理システムは、その合計カウンタの値を表示するために、MOVE文の規則に従って、印字項目に転記します。
5. SUM句を含む基本項目の記述項に対して、データ名を書くことができます。それは、合計カウンタの名前であって印字項目の名前ではありません。
6. 合計カウンタの値は、手続き部の文によって変更することができます。

7. GENERATE文およびTERMINATE文の実行中に、報告書作成管理システムは、合計カウンタに一意名-1のデータ項目の値を加えます。合計カウンタへの加算には、縦の集計、横の集計および繰上げの3種類があります。縦の集計は、GENERATE文の実行中に制御切れの処理をした後、明細報告集団を表示する前に行われます。
横の集計と繰上げは、制御脚書き報告集団の処理中に行われます。

8. UPON指定を書くと、縦の集計をデータ名-1の各繰返しの明細報告集団に対して選択的に行うことができます。

9. 報告書作成管理システムは、各加数(SUM句中の一意名-1の各繰返し)をその性質に従って合計カウンタに加算します。合計カウンタの加算の方法を以下に示します。

   ー 加数が同じ制御脚書き報告集団中の合計カウンタであるとき、合計カウンタとその加数との加算を、「横の集計」といいます。
   横の集計は、制御切れによって、制御脚書き報告集団が処理されるときに行われます。横の集計は、合計カウンタが制御脚書き報告集団中に定義されている順に行われます。
   すなわち、制御脚書き報告集団の最初の合計カウンタに対して、すべての加算が行われ、次に制御脚書き報告集団の2番目の合計カウンタに対して、すべての加算が行われます。この手続きは、すべての横の集計が終わるまで繰り返されます。
   加数の1つがこのSUM句が書かれたデータ記述項で定義された合計カウンタであるとき、加算を行う直前でのその合計カウンタの値が加算に使われます。

   ー 加数がより低いレベルの制御脚書き報告集団の合計カウンタであるとき、合計カウンタとその加数との加算を、「繰上げ」といいます。これは、制御切れによって、低いレベルの制御脚書き報告集団が処理されるときに行われます。

   ー 加数が合計カウンタでないとき、合計カウンタとその加数との加算を、「縦の集計」といいます。SUM句中にUPON指定を書くと、その加数は、指定された明細報告集団に対するGENERATE文が実行されるたびに加算されます。SUM句中にUPON指定を書かなければ、その加数は、SUM句が書かれた報告書に対する、データ名を指定したすべてのGENERATE文が実行されるたびに加算されます。

10. 加数として同じ一意名を2回以上指定すると、その加数は指定された回数だけ合計カウンタに加算されます。UPON指定に同じデータ名を2回以上指定すると、その明細報告集団に対するデータ名を指定したGENERATE文が実行されるときに、その回数だけ加算が繰り返されます。

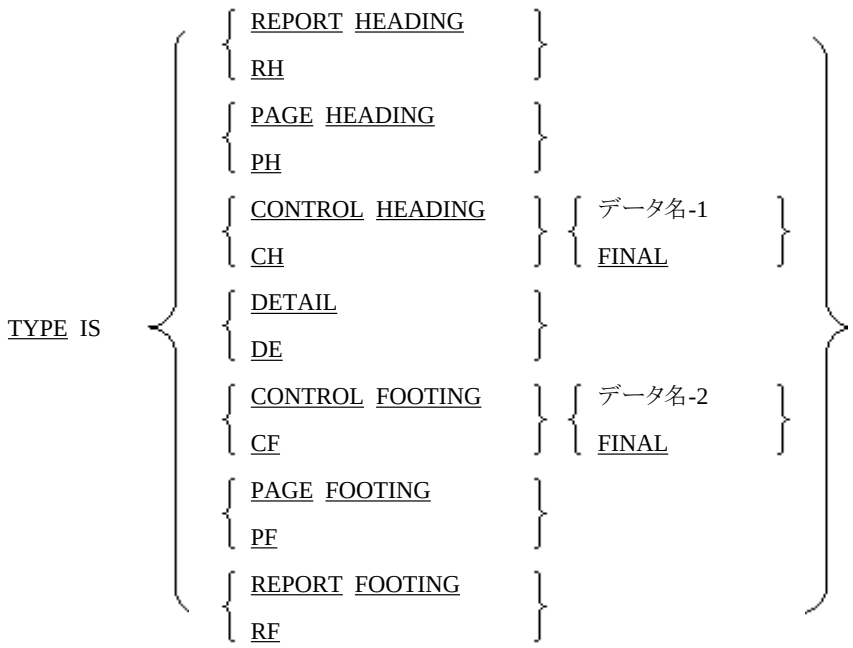
11. 報告書名を指定したGENERATE文が実行されるときに行われる縦の集計については、“6.4.21 GENERATE文（報告書作成）”を参照してください。

12. RESET指定を省略すると、報告書作成管理システムはその合計カウンタを含む制御脚書き報告集団を処理するときに、その合計カウンタに0を設定します。RESET指定を書くと、報告書作成管理システムはRESET指定が指定されたレベルの制御脚書き報告集団を処理した後に、合計カウンタに0を設定します。報告集団を処理する時点については、“5.7.8 TYPE句”参照してください。
    INITIATE文を実行すると、報告書作成管理システムは、その報告書の合計カウンタのすべてに0を設定します。

5.7.8 TYPE句

報告集団の種類と表示すべき時点を指定します。

【書き方】



構文規則

1. 以下に示す語はそれぞれが同義語です。
   - REPORT HEADINGとRH
   - PAGE HEADINGとPH
   - CONTROL HEADINGとCH
   - DETAILとDE
   - CONTROL FOOTINGとCF
   - PAGE FOOTINGとPF
   - REPORT FOOTINGとRF
2. 報告書頭書き報告集団、ページ頭書き報告集団、FINALを指定した制御頭書き報告集団、FINALを指定した制御脚書き報告集団、ページ脚書き報告集団および報告書脚書き報告集団は、1つの報告書にそれぞれ1つ書くことができます。
3. ページ頭書き報告集団とページ脚書き報告集団は、その報告書記述項にPAGE句を指定したときにだけ、書くことができます。
4. データ名-1、データ名-2およびFINALは、報告書記述項のCONTROL句にも指定しておかなければなりません。報告書記述項のCONTROL句に指定した各データ名またはFINALのそれぞれに対し、1つの制御頭書き報告集団と1つの制御脚書き報告集団だけ書くことができます。また、その両方を省略することもできます。
5. ページ頭書き報告集団およびページ脚書き報告集団中のSOURCE句および関連するUSE手続きでは、制御データ名を参照することはできません。
6. 制御脚書き報告集団、ページ頭書き報告集団、ページ脚書き報告集団および報告書脚書き報告集団中のSOURCE句および関連するUSE手続きでは、以下のデータ項目を参照することはできません。
   - 制御データ項目を含む集団項目
   - 制御データ項目に従属するデータ項目
   - 制御データ項目の一部または全部を再定義または再命名したデータ項目

7. 手続き部に報告書名を指定したGENERATE文を書くときは、その報告書記述項に、2つ以上の明細報告集団を指定することはできません。この報告書に対してデータ名を指定したGENERATE文を書かないときは、明細報告集団を省略することができます。

8. 1つの報告書の記述の中には、少なくとも1つの本体集団の記述がなければなりません。

一般規則

1. 明細報告集団は、GENERATE文で直接に作成を指示され、報告書作成管理システムが処理します。明細報告集団以外の報告集団は、報告書作成管理システムが自動的に処理します。

2. REPORT HEADING指定は、報告書の最初の報告集団として1つの報告書で一度だけ報告書作成管理システムが処理する報告集団に指定します。報告書の最初の報告集団を「報告書頭書き報告集団」といいます。報告書頭書き報告集団は、その報告書に対する最初のGENERATE文の実行中に処理されます。

3. PAGE HEADING指定は、以下に示す場合を除き、報告書の各ページの最初の報告集団として、報告書作成管理システムが処理する報告集団に指定します。報告書のページの最初の報告集団を「ページ頭書き報告集団」といいます。

   ー 報告書頭書き報告集団または報告書脚書き報告集団だけで1ページを構成し、ページ頭書き報告集団が処理されない場合

   ー 単独では1ページを構成しないような報告書頭書き報告集団が既にそのページに表示されていて、ページ頭書き報告集団が、ページの2番目の報告集団として処理される場合

4. CONTROL HEADING指定は、制御データ名に対する制御集団の始めに報告書作成管理システムが処理する報告集団に指定します。制御集団の最初の報告集団を「制御頭書き報告集団」といいます。
   またFINALを指定すると、報告書に対する最初のGENERATE文の実行中に処理されます。報告書作成管理システムは、GENERATE文の実行中に制御切れを検出すると、その最も高いレベル以下のすべての制御頭書き報告集団を処理します。

5. DETAIL指定は、対応するGENERATE文を実行するとき、報告書作成管理システムが処理する報告集団に指定します。DETAIL指定のTYPE句を指定した報告集団を「明細報告集団」といいます。明細報告集団、制御頭書き報告集団および制御脚書き報告集団を総称して「本体集団」といいます。

6. CONTROL FOOTING指定は、制御データ名に対する制御集団の終わりに報告書作成管理システムが処理する報告集団に指定します。制御集団の最後の報告集団を「制御脚書き報告集団」といいます。
   またFINALを指定すると、報告書の最後の本体集団として、報告書でただ一度だけ処理されます。報告書作成管理システムは、GENERATE文の実行中に制御切れを検出すると、その最も高いレベル以下のすべての制御脚書き報告集団を処理します。報告書に対してGENERATE文が少なくとも1回実行されていれば、TERMINATE文の実行中にすべての制御脚書き報告集団が表示されます。

7. PAGE FOOTING指定は、以下に示す場合を除き、報告書の各ページの最後の報告集団として、報告書作成管理システムが処理する報告集団に指定します。報告書のページの最後の報告集団を「ページ脚書き報告集団」といいます。

   ー 報告書頭書き報告集団または報告書脚書き報告集団だけで1ページを構成し、ページ脚書き報告集団が処理されない場合

   ー 単独では1ページを構成しないような報告書脚書き報告集団がそのページの最後に表示され、ページ脚書き報告集団はページの最後から2番目の報告集団として処理される場合

8. REPORT FOOTING指定は、報告書の最後の報告集団として1つの報告書で一度だけ報告書作成管理システムが処理する報告集団に指定します。報告書の最後の報告集団を「報告書脚書き報告集団」といいます。報告書に対してGENERATE文が少なくとも1回実行されていれば、TERMINATE文の実行中に報告書脚書き報告集団が処理されます。

9. 報告書頭書き報告集団、ページ頭書き報告集団、制御頭書き報告集団、ページ脚書き報告集団および報告書脚書き報告集団を報告書作成管理システムは以下の順序で処理します。

   ー 報告集団のデータ名を参照しているUSE BEFORE REPORTING手続きがあれば、USE手続きを実行します。

   ー SUPPRESS文が実行されたか、または報告集団が印字用でなければ、報告集団に対する処理を終了します。

ー SUPPRESS文が実行されず、報告集団が印字用であれば、印字行を作成し、報告集団の表示規則に従って、報告集団を表示します。

10. 制御脚書き報告集団を、報告書作成管理システムは以下の順序で処理します。

    a. 合計カウンタを横に集計します。すなわち、この報告集団中で定義された合計カウンタであって、同じ報告集団中のSUM句の作用対象でもある合計カウンタをすべて、SUM句の主体である合計カウンタに加えます。

    b. 合計カウンタを繰り上げて集計します。すなわち、この報告集団中で定義された合計カウンタであって、これよりもレベルの高い制御脚書き報告集団中のSUM句の作用対象でもあるものをすべて、そのレベルの高い制御脚書き報告集団中の合計カウンタに加えます。

    c. この報告集団のデータ名を参照しているUSE BEFORE REPORTING手続きがあれば、USE手続きを実行します。

    d. SUPPRESS文が実行されたか、報告集団が印字用でなければ、次にf.の処理を行います。

    e. SUPPRESS文が実行されず、報告集団が印字用であれば、印字行を作成し、制御脚書き報告集団の表示規則に従って、報告集団を表示します。

    f. 制御階層中のこのレベルを処理するときに、必要により合計カウンタに0を設定します。

    制御切れが起こったとき、報告書作成管理システムは、GENERATE文の規則に従って最も低いレベルから始めてレベルが高くなる順に、制御切れが検出されたうちで最も高いレベルの制御脚書き報告集団までを作成表示します。このとき、制御脚書き報告集団が制御データ項目に対して定義されていなくても、報告書記述のRESET指定にその制御データ名が指定されていれば、報告書作成管理システムはf.の処理を行います。

11. データ名を指定したGENERATE文に対応して、報告書作成管理システムが行う明細報告集団の処理はa.～e.によります。
    報告書の記述中に明細報告集団をただ1つだけ書いた場合に、報告書名を指定したGENERATE文に対応して、報告書作成管理システムが行う明細に関する処理はa.～d.によります。これは、データ名を指定したGENERATE文が実行されたかのように処理します。
    報告書の記述中に明細報告集団を1つも書かない場合に、報告書名を指定したGENERATE文に対応して、報告書作成管理システムが行う明細に関する処理はa.によります。これは、報告書の記述中に明細報告集団がただ1つだけ書かれていて、データ名を指定したGENERATE文が実行されたかのように処理します。

    a. 明細報告書集団に指定されているすべての縦の集計を行います。

    b. この報告集団のデータ名を参照しているUSE BEFORE REPORTING手続きがあれば、USE手続きを実行します。

    c. SUPPRESS文が実行されたか、報告集団が印字用でなければ、報告集団に対する処理を終了します。

    d. 明細報告集団の処理が報告書名を指定したGENERATE文による場合、報告集団に対する処理を終了します。

    e. c.でもd.でもなければ、報告書作成管理システムは印字行を作成し、明細報告集団の表示規則に従って、報告集団を表示します。

12. 一般規則9.、10.および11.に従って、制御頭書き報告集団、制御脚書き報告集団または明細報告集団を処理しているとき、報告書作成管理システムは本体集団の表示を決定した後、その処理を一時中止し、実際に本体集団を表示する前に、改ページの処理とそれに伴うページ脚書き報告集団およびページ頭書き報告集団の処理を実行することがあります。

13. 制御データ項目を参照したときの値は、以下のとおりです。ただし、報告書作成管理システムが、制御切れの検出に使った制御データ項目の保存された値を、古い値といいます。制御データ項目については、“5.6.2 CONTROL句”を参照してください。

    a. 制御脚書き報告集団の制御切れの処理中に、関連するUSE手続きまたはSOURCE句で制御データ項目を参照すると、古い値が使われます。

    b. TERMINATE文が実行されたとき、報告書作成管理システムは、最も高いレベルの制御切れが検出されたとみなします。そして、制御脚書き報告集団および報告書脚書き報告集団中のSOURCE句または関連するUSE手続きで制御データ項目を参照すると、古い値が使われます。

c. 報告集団および関連するUSE手続きでデータ項目を参照すると、a.およびb.以外はすべて、その報告集団が処理されるときにそのデータ項目に含まれている現在の値が使われます。

5.7.9 USAGE句

記憶装置内でのデータ項目の表現形式を指定します。

【書き方】

[USAGE IS] DISPLAY

構文規則

1. USAGE句は、どのデータ記述項にも書くことができます。
2. USAGE句を集団項目のデータ記述項に書いたとき、従属する基本項目または集団項目のデータ記述項にもUSAGE句を書くことができます。
3. 報告集団記述項のUSAGE句には、USAGE IS DISPLAYだけを指定することができます。

一般規則

1. USAGE句を集団項目のレベルに書くと、その中のすべての基本項目に適用されます。
2. USAGE句は、記憶装置内でのデータ項目の表現形式を指定します。
3. USAGE IS DISPLAY句は、データ項目の表現形式が標準データ形式であることを示します。
4. 基本項目またはそれを含む集団項目にUSAGE句を指定しなかった場合、用途は表示用(DISPLAY)とみなされます。

5.7.10 VALUE句

印字項目の値を指定します。

【書き方】

VALUE IS 定数-1

構文規則

1. 符号付き数字定数は、符号付き数字項目を記述するPICTURE句の文字列に対応していなければなりません。
2. VALUE句中の数字定数は、PICTURE句で示される値の範囲内に入るものでなければならず、ゼロでない数字の切捨てを引き起こすものであってはなりません。VALUE句の定数が文字定数のときは、PICTURE句で示されている大きさを超えてはなりません。

一般規則

1. VALUE句は、この項目やこれを含む集団項目のデータ記述に書いた他の句と矛盾するものであってはなりません。さらに以下の規則を適用します。
   - 項類が数字の場合、VALUE句の定数-1は数字定数でなければなりません。
   - 項類が英字、英数字、英数字編集または数字編集の場合、VALUE句の定数-1は文字定数でなければなりません。この定数はデータ項目が英数字と記述されているものとして扱われます。

— 項類が日本語または日本語編集の場合、VALUE句の定数-1は日本語定数でなければなりません。この定数はデータ項目が日本語と記述されているものとして扱われます。

— 項類がブールの場合、VALUE句の定数-1はブール定数でなければなりません。

— PICTURE句の中にデータ項目の大きさに数えるような編集用文字が入っていても、このデータ項目に初期値を与えるときは無視されます。したがって、編集項目の定数は、編集の済んだ形にしておくべきです。

— 項目に初期値を与えるとき、BLANK WHEN ZERO句およびJUSTIFIED句の影響は受けません。

2. 報告書節では、VALUE句を含む基本記述項がGROUP INDICATE句を含んでいなければ、印字項目には報告集団が表示されるたびに指定された値が与えられます。しかし、GROUP INDICATE句を含むときは、指定された値は本節の“5.7.2 GROUP INDICATE句”に示した条件が成り立つときだけ表示されます。

5.8 報告集団の表示規則

この節では、以下のことを説明します。

- 報告集団のそれぞれの型に対するLINE NUMBER句とNEXT GROUP句の可能な組合せ
- LINE NUMBER句およびNEXT GROUP句を使用する場合の必要条件
- LINE NUMBER句およびNEXT GROUP句に対する報告書作成管理システムの処理

5.8.1 表示規則表の見かた

報告書頭書き報告集団、ページ頭書き報告集団、ページ脚書き報告集団および報告書脚書き報告集団に対しては、個々に表示規則表があります。明細報告集団、制御頭書き報告集団および制御脚書き報告集団は、本体集団の表示規則表でまとめて扱います。

表示規則表の第1欄と第2欄は、指定された報告集団の型に対するLINE NUMBER句とNEXT GROUP句すべての可能な組合せを示します。したがって、LINE NUMBER句とNEXT GROUP句との1つの組合せに対しては、表示規則表のその列を左から右へと読みます。

適用規則の欄の見かた

表示規則表の適用規則の欄は、2つに分かれます。最初の部分は、報告書記述がPAGE句を含んでいる場合に適用し、2番目の部分は、PAGE句が省略されている場合に適用します。“ー”は、適用される規則がないことを示します。適用規則の欄にある各規則の名前と目的は、以下のとおりです。

上限と下限の規則

これらの規則は、指定された報告集団が表示されるページ内の縦の範囲を規定します。PAGE句がない場合には、印字される報告書は、縦に分割されません。したがって、PAGE句のない報告書記述に対し、上限と下限の規則は表の中に現れません。

改ページの規則

改ページの規則は、本体集団にだけ適用します。したがって、改ページの規則は、本体集団の表示規則表にだけ現れます。実行時に報告書作成管理システムは、報告書の現在のページに、指定された本体集団を表示できるかどうかを、改ページの規則を適用して決めます。報告書記述項にPAGE句がない場合には、本体集団に対しても改ページの規則は適用されません。

印字開始行位置の規則

印字開始行位置の規則は、報告書作成管理システムが、報告集団の最初の行を報告書媒体上のどこに表示すべきかを規定します。表示規則表では、報告書作成管理システムが、報告集団の(もしあれば)2番目以降の印字行を報告書媒体上のどこに表示すべきかは規定されません。これは、LINE NUMBER句の一般規則で定めます。

次の報告集団の規則

次の報告集団の規則は、NEXT GROUP句の使い方に関して規定します。

行カウンタの最終値設定の規則

行カウンタの最終値設定の規則は、報告集団を表示した後、報告書作成管理システムが行カウンタに設定する最終の値を規定します。

LINE NUMBER句の表記法

表示規則表の第1欄は、1つの報告集団中のLINE NUMBER句の順序を示します。記号の意味は、以下のとおりです。

- “絶対”は、NEXT PAGE指定なしのLINE NUMBER句を書くことを示します。NEXT PAGE指定なしのLINE NUMBER句では、絶対的な行位置を指定します。NEXT PAGE指定なしのLINE NUMBER句は、2つ以上続けて書くことができます。その後に、PLUS指定のLINE NUMBER句を1つ以上続けて書くこともできます。
- “PLUS”は、PLUS指定のLINE NUMBER句を書くことを示します。PLUS指定のLINE NUMBER句では、相対的な行位置を指定します。PLUS指定のLINE NUMBER句は、2つ以上続けて書くことができます。
- “NEXT PAGE”は、NEXT PAGE指定付きのLINE NUMBER句を書くことを示します。NEXT PAGE指定付きのLINE NUMBER句では、ページ替えをした後の絶対的な行位置を指定します。この句の後に、NEXT PAGE指定なしのLINE NUMBER句またはPLUS指定のLINE NUMBER句を1つ以上続けて書くこともできます。
- “省略”は、LINE NUMBER句を省略することを示します。

NEXT GROUP句の表記法

表示規則表の第2欄は、1つの報告集団中のNEXT GROUP句の書き方を示します。1つの報告集団中に、NEXT GROUP句は、1つだけ書くことができます。記号の意味は、以下のとおりです。

- “絶対”は、整数-1指定のNEXT GROUP句を書くことを示します。
- “PLUS”は、PLUS指定のNEXT GROUP句を書くことを示します。
- “NEXT PAGE”は、NEXT PAGE指定のNEXT GROUP句を書くことを示します。
- “省略”は、NEXT GROUP句を省略することを示します。
- “禁止”は、NEXT GROUP句を書いてはいけないことを示します。

次の報告集団用整数退避項目

次の報告集団用整数退避項目は、報告書作成管理システムだけが扱えるデータ項目です。NEXT GROUP句で現在のページに適用できない絶対的な行位置が指定されると、報告書作成管理システムは、その値を次の報告集団用整数退避項目に設定します。

改ページ処理を行った後、報告書作成管理システムは、次の報告集団用整数退避項目に設定された値を使って、次の本体集団の位置付けを行います。

5.8.2 報告書頭書き報告集団の表示規則

報告書頭書き報告集団のLINE NUMBER句とNEXT GROUP句の可能なすべての組合せおよびその表示規則を、下表に示します。

LINE NUMBER句とNEXT GROUP句の組み合わせ		適用規則						
		PAGE句あり				PAGE句なし		
LINE NUMBER句	NEXT GROUP句	上限	下限	印字開始行位置	次の報告集団	行カウンタの最終値設定	印字位置開始行位置	行カウンタの最終値設定
絶対	**絶対**	(1)	(2)－a	(3)－a	(4)－a	(5)－a	無効な組合わせ	
絶対	**PLUS**	(1)	(2)－a	(3)－a	(4)－b	(5)－b	無効な組合わせ	
絶対	**NEXT PAGE**	(1)	(2)－b	(3)－a	(4)－c	(5)－c	無効な組合わせ	
絶対	**省略**	(1)	(2)－a	(3)－a	---	(5)－d	無効な組合わせ	
PLUS	**絶対**	(1)	(2)－a	(3)－b	(4)－a	(5)－a	無効な組合わせ	
PLUS	**PLUS**	(1)	(2)－a	(3)－b	(4)－b	(5)－b	(3)－d	(5)－b

LINE NUMBER句とNEXT GROUP句の組み合わせ		適用規則						
		PAGE句あり				PAGE句なし		
LINE NUMBER句	NEXT GROUP句	上限	下限	印字開始行位置	次の報告集団	行カウンタの最終値設定	印字位置開始行位置	行カウンタの最終値設定
PLUS	**NEXT PAGE**	(1)	(2)－b	(3)－b	(4)－c	(5)－c	無効な組合わせ	
PLUS	**省略**	(1)	(2)－a	(3)－b	---	(5)－d	(3)－d	(5)－d
省略	**禁止**	---	---	(3)－c	---	(5)－e	(3)－c	(5)－e

上表の見かたについては、“5.8.1 表示規則表の見かた”を参照してください。

上表の括弧内の番号の意味は、以下のとおりです。

(1) 上限の規則

報告書頭書き報告集団を表示できる最初の行位置は、PAGE句中のHEADING指定で指定します。

(2) 下限の規則

a. 報告書頭書き報告集団を表示できる最後の行位置は、PAGE句中のFIRST DETAIL指定の整数-3から1を引いた値とします。

b. 報告書頭書き報告集団を表示できる最後の行位置は、PAGE句の整数-1で指定します。

(3) 印字開始行位置の規則

a. 報告書頭書き報告集団の最初の印字行は、そのLINE NUMBER句の整数で指定された行位置に表示します。

b. 報告書頭書き報告集団の最初の印字行は、最初のLINE NUMBER句の整数と、PAGE句中のHEADING指定の整数-2から1を引いた値の和で示される行位置に表示します。

c. 報告書頭書き報告集団は、表示しません。

d. 報告書頭書き報告集団の最初の印字行は、行カウンタの値(この場合は0)に最初のLINE NUMBER句の整数を加えて得られる行位置に表示します。

(4) 次の報告集団の規則

a. NEXT GROUP句の整数は、報告書頭書き報告集団の最後の印字行を表示する行位置より大きく、PAGE句中のFIRST DETAIL指定の整数-3が示す行位置より小さくなければなりません。

b. NEXT GROUP句の整数と、報告書頭書き報告集団の最後の印字行を表示する行位置との和は、PAGE句中のFIRST DETAIL指定の整数-3より小さくなければなりません。

c. NEXT GROUP句のNEXT PAGE指定は、報告書頭書き報告集団を報告書の最初のページに単独で表示することを意味します。報告書作成管理システムは、報告書が最初のページに位置付けられている間は、他の報告集団があってもそれらを処理しません。

(5) 行カウンタの最終値設定の規則

a. 報告書作成管理システムは、最終値として行カウンタにNEXT GROUP句の整数を設定します。

b. 報告書作成管理システムは、最終値として行カウンタに、NEXT GROUP句の整数と、報告書頭書き報告集団の最後の印字行を表示した行位置との和を設定します。

c. 報告書作成管理システムは、最終値として行カウンタに0を設定します。

d. 報告書作成管理システムは、最終値として行カウンタに、報告書頭書き報告集団の最後の印字行を表示した行位置を設定します。

e. 行カウンタは、印字項目を含まない報告集団の処理によって影響されることはありません。

5.8.3 ページ頭書き報告集団の表示規則

ページ頭書き報告集団のLINE NUMBER句とNEXT GROUP句の可能なすべての組合せおよびその表示規則を、下表に示します。

LINE NUMBER句とNEXT GROUP句の組み合わせ		適用規則				
		PAGE句あり (*1)				
LINE NUMBER句	NEXT GROUP句	上限	下限	印字開始行位置	次の報告集団	行カウンタの最終値設定
絶対	**禁止**	(1)	(2)	(3)－a	---	(4)－a
PLUS	**禁止**	(1)	(2)	(3)－b	---	(4)－a
省略	**禁止**	---	---	(3)－c	---	(4)－b

*1 ： 報告書記述項にPAGE句がない場合は、ページ頭書き報告集団を定義してはいけません。

上表の見かたについては、“5.8.1 表示規則表の見かた”を参照してください。

上表の括弧内の番号の意味は、以下のとおりです。

(1) 上限の規則

ページ頭書き報告集団を表示すべきページに、既に報告書頭書き報告集団が表示されていれば、ページ頭書き報告集団を表示できる最初の行位置は、報告書頭書き報告集団で設定された行カウンタの最終値に1を加えた値とします。
その他の場合には、ページ頭書き報告集団を表示できる最初の行位置は、PAGE句中のHEADING指定で指定します。

(2) 下限の規則

ページ頭書き報告集団を表示できる最終の行位置は、PAGE句中のFIRST DETAIL指定の整数-3から1を引いた値とします。

(3) 印字開始行位置の規則

a. ページ頭書き報告集団の最初の印字行は、その集団のLINE NUMBER句の整数で指定された行位置に表示します。

b. ページ頭書き報告集団を表示しようとするページに、既に報告書頭書き報告集団が表示されていれば、報告書頭書き報告集団によって設定された行カウンタの最終値とページ頭書き報告集団の最初のLINE NUMBER句の整数との和で示される行位置に、ページ頭書き報告集団の最初の印字行を表示します。
その他の場合には、ページ頭書き報告集団の最初のLINE NUMBER句の整数と、PAGE句中のHEADING指定の整数-2から1を引いた値の和で示される位置に、ページ頭書き報告集団の最初の印字行を表示します。

c. ページ頭書き報告集団は、表示されません。

(4) 行カウンタの最終値設定の規則

a. 報告書作成管理システムは、最終値として行カウンタに、ページ頭書き報告集団の最後の印字行を表示した行位置を設定します。

b. 行カウンタは、印字項目は含まない報告集団の処理によって影響されることはありません。

5.8.4 本体集団の表示規則

制御頭書き報告集団、明細報告集団および制御脚書き報告集団のLINE NUMBER句とNEXT GROUP句の可能なすべての組合せおよびその表示規則を、下表に示します。

LINE NUMBER句とNEXT GROUP句の組み合わせ		適用規則							
		PAGE句あり						PAGE句なし	
LINE NUMBER句	NEXT GROUP句	上限	下限	改ページ	印字開始行位置	次の報告集団	行カウンタの最終値設定	印字位置開始行位置	行カウンタの最終値設定
絶対	**絶対**	(1)	(2)	(3)－a	(4)－a	(5)	(6)－a	無効な組合わせ	
絶対	**PLUS**	(1)	(2)	(3)－a	(4)－a	---	(6)－b	無効な組合わせ	
絶対	**NEXT PAGE**	(1)	(2)	(3)－a	(4)－a	---	(6)－c	無効な組合わせ	
絶対	**省略**	(1)	(2)	(3)－a	(4)－a	---	(6)－d	無効な組合わせ	
PLUS	**絶対**	(1)	(2)	(3)－b	(4)－b	(5)	(6)－a	無効な組合わせ	
PLUS	**PLUS**	(1)	(2)	(3)－b	(4)－b	---	(6)－b	(4)－d	(6)－f
PLUS	**NEXT PAGE**	(1)	(2)	(3)－b	(4)－b	---	(6)－c	無効な組合わせ	
PLUS	**省略**	(1)	(2)	(3)－b	(4)－b	---	(6)－d	(4)－d	(6)－d
NEXT PAGE	**絶対**	(1)	(2)	(3)－c	(4)－a	(5)	(6)－a	無効な組合わせ	
NEXT PAGE	**PLUS**	(1)	(2)	(3)－c	(4)－a	---	(6)－b	無効な組合わせ	
NEXT PAGE	**NEXT PAGE**	(1)	(2)	(3)－c	(4)－a	---	(6)－c	無効な組合わせ	
NEXT PAGE	**省略**	(1)	(2)	(3)－c	(4)－a	---	(6)－d	無効な組合わせ	
省略	**禁止**	---	---	---	(4)－c	---	(6)－a	(4)－c	(6)－e

上表の見かたについては、“5.8.1 表示規則表の見かた”を参照してください。

上表の括弧内の番号の意味は、以下のとおりです。

(1) 上限の規則

本体集団を表示できる最初の行位置は、PAGE句中のFIRST DETAIL指定で指定します。

(2) 下限の規則

制御頭書き報告集団または明細報告集団を表示できる最後の行位置は、PAGE句中のLAST DETAIL指定で指定します。

制御脚書き報告集団を表示できる最後の行位置は、PAGE句中のFOOTING指定で指定します。

(3) 改ページの規則

a. 行カウンタの値が、最初の絶対的なLINE NUMBER句の整数より小さければ、本体集団を現在のページに表示します。
その他の場合には、報告書作成管理システムは改ページの処理を行います。ページ頭書き報告集団が定義されていれば、その処理の後、報告書作成管理システムは、前のページの最後の本体集団を表示したときに、次の報告集団用整数退避項目に値が設定されたかどうか調べます(6.a.を参照してください)。次の報告集団用整数退避項目に値が設定されていなければ、本体集団を現在のページに表示します。次の報告集団用整数退避項目に値が設定されていれば、報告書作成管理システムは、次の報告集団用整数退避項目の値を行カウンタに転記し、次の報告集団用整数退避項目に0を設定し、3.a.を再び適用します。

b. 報告書の現在のページに、すでに本体集団が表示されていれば、報告書作成管理システムは、行カウンタの内容とその報告集団のすべてのLINE NUMBER句の整数とを加え、試験的な合計を計算します。この合計がその本体集団の下限の整数より大きくなければ、その報告集団を現在のページに表示します。
この合計が本体集団の下限の整数より大きければ、報告書作成管理システムは改ページの処理を行います。

ページ頭書き報告集団が定義されていれば、その処理の後、報告書作成管理システムは3.b.を再び適用します。報告書の現在のページに、本体集団がまだ表示されていなければ、報告書作成管理システムは、前のページの最後の本体集団を表示したときに、次の報告集団用整数退避項目に値が設定されたかどうか調べます(6.a.を参照してください)。
次の報告集団用整数退避項目に値が設定されていなければ、その本体集団を報告書の現在のページに表示します。次の報告集団用整数退避項目に値が設定されていれば、報告書作成管理システムは、次の報告集団用整数退避項目の値を行カウンタに転記し、次の報告集団用整数退避項目に0を設定し、試験的な合計を計算します。
この合計は、行カウンタの値に1およびその本体集団の最初を除くすべてのLINE NUMBER句の整数を加えて計算されます。この合計がその本体集団の下限の整数より大きくなければ、その本体集団を現在のページに表示します。この合計がその本体集団の下限の整数より大きければ、報告書作成管理システムは改ページの処理を行います。ページ頭書き報告集団が定義されていれば、その処理の後、本体集団をそのページに表示します。

c. 報告書の現在のページに、すでに本体集団が表示されていれば、報告書作成管理システムは改ページの処理を行います。ページ頭書き報告集団が定義されていれば、その処理の後、3.c.を再び適用します。報告書の現在のページに、本体集団がまだ表示されていなければ、報告書作成管理システムは、前のページの最後の本体集団を表示したときに、次の報告集団用整数退避項目に値が設定されたかどうかを調べます(6.a.を参照してください)。次の報告集団用整数退避項目に値が設定されていなければ、その本体集団を報告書の現在のページに表示します。次の報告集団用整数退避項目に値が設定されていれば、報告書作成管理システムは、次の報告集団用整数退避項目の値を行カウンタに転記し、次の報告集団用整数退避項目に0を設定します。そして、行カウンタの値が最初の絶対的なLINE NUMBER句の整数より小さければ、その本体集団を現在のページに表示します。そうでなければ、報告書作成管理システムは改ページの処理を行います。ページ頭書き報告集団が定義されていれば、その処理の後、そのページに本体集団を表示します。

(4) 印字開始行位置の規則

a. 本体集団の最初の印字行は、その集団のLINE NUMBER句の整数で指定された行位置に表示されます。

b. 行カウンタの値がPAGE句中のFIRST DETAIL指定で指定された行位置以上であり、かつ、現在のページにまだどの本体集団も表示されていなければ、現在の本体集団の最初の印字行は、行カウンタの値で示される行の次の行に表示されます。
行カウンタの値がPAGE句中のFIRST DETAIL指定で指定された行位置以上であり、かつ、現在のページにすでに本体集団が表示されていれば、現在の本体集団の最初の印字行は、行カウンタの値と現在の本体集団の最初のLINE NUMBER句の整数とを加えた値で示される行に表示されます。行カウンタの値がPAGE句中のFIRST DETAIL指定で指定された行位置より小さければ、その本体集団の最初の印字行は、FIRST DETAIL指定で指定された行に表示されます。

c. 本体集団は、表示されません。

d. 行カウンタの値と最初のLINE NUMBER句の整数との和が、最初の印字行が表示される行位置を指定します。

(5) 次の報告集団の規則

絶対的なNEXT GROUP句の整数は、PAGE句中のFIRST DETAIL指定で指定する行位置以上、PAGE句中のFOOTING指定で指定する行位置以下でなければなりません。

(6) 行カウンタの最終値設定の規則

a. 直前に表示された本体集団が制御脚書き報告集団のとき、それが制御切れに対応する制御脚書き報告集団の中でレベルが最高のものでなければ、行カウンタの最終値は、その制御脚書き報告集団の最後の印字行を表示した行位置とします。
上記以外のすべての場合、報告書作成管理システムは、本体集団の最後の印字行を表示した行位置とNEXT GROUP指定の整数とを比較します。前者が後者より小さければ、報告書作成管理システムは、最終値として行カウンタに、NEXT GROUP指定の整数を設定します。前者が後者以上ならば、報告書作成管理システムは、最終値として行カウンタに、PAGE句中のFOOTING指定によって指定された行位置を設定し、さらにNEXT GROUP句の整数を次の報告集団用整数退避項目に設定します。

b. 直前に表示された本体集団が制御脚書き報告集団のとき、それが制御切れに対応する制御脚書き報告集団の中でレベルが最高のものでなければ、行カウンタの最終値は、その制御脚書き報告集団の最後の印字行を表示した行位置とします。
上記以外のすべての場合、報告書作成管理システムは、NEXT GROUP句の整数と本体集団の最後の印字

行を表示した行位置とを加えて、試験的な合計を計算します。合計がPAGE句中のFOOTING指定で指定された行位置より小さければ、報告書作成管理システムは、最終値として行カウンタにその合計を設定します。合計がPAGE句中のFOOTING指定で指定された行位置以上であれば、報告書作成管理システムは、最終値として行カウンタに、PAGE句中のFOOTING指定で指定された行位置を設定します。

c. 直前に表示された本体集団が制御脚書き報告集団のとき、それが制御切れに対応する制御脚書き報告集団の中でレベルが最高のものでなければ、行カウンタの最終値は、その制御脚書き報告集団の最後の印字行を表示した行位置とします。
上記以外のすべての場合、報告書作成管理システムは、最終値として行カウンタに、PAGE句中のFOOTING指定で指定された行位置を設定します。

d. 行カウンタの最終値は、本体集団の最後の印字行を表示した行位置とします。

e. 行カウンタは、印字項目を含まない本体集団の処理によって影響されることはありません。

f. 直前に表示された本体集団が制御脚書き報告集団のとき、それが制御切れに対応する制御脚書き報告集団の中でレベルが最高のものでなければ、行カウンタの最終値は、その制御脚書き報告集団の最後の印字行を表示した行位置とします。

上記以外のすべての場合、報告書作成管理システムは、最終値として行カウンタに、最後の印字行を表示した行位置とNEXT GROUP句の整数との和を設定します。

5.8.5 ページ脚書き報告集団の表示規則

ページ脚書き報告集団のLINE NUMBER句とNEXT GROUP句の可能なすべての組合せおよびその表示規則を、下表に示します。

LINE NUMBER句とNEXT GROUP句の組み合わせ		適用規則				
		PAGE句あり (*1)				
LINE NUMBER句	NEXT GROUP句	上限	下限	印字開始行位置	次の報告集団	行カウンタの最終値設定
絶対	**絶対**	(1)	(2)	(3)－a	(4)－a	(5)－a
絶対	**PLUS**	(1)	(2)	(3)－a	(4)－b	(5)－b
絶対	**省略**	(1)	(2)	(3)－a	---	(5)－c
省略	**禁止**	---	---	(3)－b	---	(5)－d

*1 ： 報告書記述項にPAGE句がない場合は、ページ頭書き報告集団を定義してはいけません。

上表の見かたについては、“5.8.1 表示規則表の見かた”を参照してください。

上表の括弧内の番号の意味は、以下のとおりです。

(1) 上限の規則

ページ脚書き報告集団を表示できる最初の行位置は、PAGE句中のFOOTING指定の整数-5に1を加えた値とします。

(2) 下限の規則

ページ脚書き報告集団を表示できる最後の行位置は、PAGE句の整数-1で指定します。

(3) 印字開始行位置の規則

a. ページ脚書き報告集団の最初の印字行は、その集団のLINE NUMBER句の整数で指定された行位置に表示します。

b. ページ脚書き報告集団は、表示されません。

(4) 次の報告集団の規則

a. NEXT GROUP句の整数は、ページ脚書き報告集団の最後の印字行を表示する行位置より大きく、PAGE句の整数-1で指定する行位置以下でなければなりません。

b. NEXT GROUP句の整数とページ脚書き報告集団の最後の印字行を表示する行位置との和は、PAGE句の整数-1で指定する行位置以下でなければなりません。

(5) 行カウンタの最終値設定の規則

a. 報告書作成管理システムは、最終値として行カウンタに、NEXT GROUP句の整数を設定します。

b. 報告書作成管理システムは、最終値として行カウンタに、NEXT GROUP句の整数とページ脚書き報告集団の最後の印字行を表示した行位置との和を設定します。

c. 報告書作成管理システムは、最終値として行カウンタに、ページ脚書き報告集団の最後の印字行を表示した行位置を設定します。

d. 行カウンタは、印字項目を含まない報告集団の処理によって影響されることはありません。

5.8.6 報告書脚書き報告集団の表示規則

報告書脚書き報告集団のLINE NUMBER句とNEXT GROUP句の可能なすべての組合せ、およびその表示規則を、下表に示します。

LINE NUMBER句とNEXT GROUP句の組み合わせ		適用規則						
		PAGE句あり					PAGE句なし	
LINE NUMBER句	NEXT GROUP句	上限	下限	印字開始行位置	次の報告集団	行カウンタの最終値設定	印字位置開始行位置	行カウンタの最終値設定
絶対	**禁止**	(1)－a	(2)	(3)－a	---	(4)－a	無効な組合わせ	
PLUS	**禁止**	(1)－a	(2)	(3)－b	---	(4)－a	(3)－d	(4)－b
NEXT PAGE	**禁止**	(1)－a	(2)	(3)－c	---	(4)－a	無効な組合わせ	
省略	**禁止**	---	---	(3)－e	---	(4)－b	(3)－e	(4)－b

上表の見かたについては、“5.8.1 表示規則表の見かた”を参照してください。上表の括弧内の番号の意味は、以下のとおりです。

(1) 上限の規則

a. 報告書の現在のページに、すでにページ脚書き報告集団が表示されていれば、報告書脚書き報告集団を表示できる最初の行位置は、ページ脚書き報告集団で設定された行カウンタの最終値に1を加えた値とします。

その他の場合には、報告書脚書き報告集団を表示できる最初の行位置は、PAGE句の整数-5に1を加えた値とします。

b. 報告書脚書き報告集団を表示できる最初の行位置は、PAGE句中のHEADING指定で指定します。

(2) 下限の規則

報告書脚書き報告集団を表示できる最後の行位置は、PAGE句の整数-1で指定します。

(3) 印字開始行位置の規則

a. 報告書脚書き報告集団の最初の印字行は、その集団のLINE NUMBER句の整数で指定された行位置に表示します。

b. 報告書の現在のページに、すでにページ脚書き報告集団が表示されていれば、ページ脚書き報告集団によって決まる行カウンタの最終値と、報告書脚書き報告集団の最初のLINE NUMBER句の整数との和が、報告書脚書き報告集団の最初の印字行を表示する行位置を示します。そのページにページ脚書き報告集団が表示されていなければ、報告書脚書き報告集団の最初のLINE NUMBER句の整数と、PAGE句中のFOOTING指定の整数-5との和が、報告書脚書き報告集団の最初の印字行を表示する行位置を示します。

c. LINE NUMBER句にNEXT PAGE指定を書くと、報告書脚書き報告集団だけで単独のページを構成します。報告書脚書き報告集団の最初の印字行は、その集団のLINE NUMBER句の整数で指定された行位置に表示します。

d. 行カウンタの値と最初のLINE NUMBER句の整数との和が、最初の印字行の表示される行位置を示します。

e. 報告書脚書き報告集団は、表示されません。

(4) **行カウンタの最終値設定の規則**

a. 報告書作成管理システムは、最終値として行カウンタに、報告書脚書き報告集団の最後の印字行を表示した行位置を設定します。

b. 行カウンタは、印字項目を含まない報告集団の処理によって影響されることはありません。

第6章 手続き部

手続き部では、実行用プログラムの処理手順を書きます。手続き部は、データ部の後に書きます。実行用プログラムの処理手順を書く必要がない場合、手続き部を省略することができます。

6.1 手続き部の構成（PROCEDURE DIVISION）

手続き部は、宣言部分と手続き部分で構成します。

宣言部分は、DECLARATIVESで始まり、END DECLARATIVESで終わります。

手続き部分は、1つの段落、連続したいくつかの段落の組、1つの節、または連続したいくつかの節の組で構成します。節に属するすべての段落は、節の中にまとめて書きます。

【書き方1】宣言部分を書かない場合（その1）

PROCEDURE DIVISION.
{節名 SECTION.
[段落名.
[完結文] …] … } …

手続き部分

【書き方2】宣言部分を書かない場合（その2）

PROCEDURE DIVISION.
[段落名.
[完結文] …] …

手続き部分

【書き方3】宣言部分を書く場合

PROCEDURE DIVISION.
DECLARATIVES.
{節名 SECTION.
USE文. (*1)
[段落名.
[完結文] …] … } …

宣言部分

END DECLARATIVES.
{節名 SECTION.
[段落名.
[完結文] …] … } …

手続き部分

*1 ： USE文は、USE BEFORE REPORTING文も含みます。

注意

このコンパイラでは、手続き部分の節名と段落名の両方を省略することができます。

節

節は、節の見出し(“節名 SECTION.”)といくつかの段落で構成します。段落は、省略することもできます。節の範囲は、節の見出しから以下のいずれかの地点までの部分です。

- 次の節の直前まで
- 手続き部の終わりまで
- 宣言部分の場合、END DECLARATIVESまで

段落

段落は、段落名といくつかの完結文で構成します。段落名の後には、分離符の終止符を書かなければなりません。完結文は、省略することもできます。段落の範囲は、段落名から以下のいずれかの地点までの部分です。

- 次の段落名の直前まで
- 次の節名の直前まで
- 手続き部の終わりまで
- 宣言部分の場合、END DECLARATIVESまで

文

完結文は、いくつかの文で構成し、分離符の終止符で終わります。手続き部に書いた文は、原則として、プログラムに書いた順に実行されます。

文には、条件文、無条件文および翻訳指示文の3種類があります。

「条件文」は、次に行う動作が条件の真理値によって決まる文です。

「無条件文」は、次に行う動作が翻訳時に決まる文です。

「翻訳指示文」は、翻訳時の動作を指示する文です。翻訳指示文は、実行時には意味を持ちません。翻訳指示文は、COPY文、REPLACE文およびUSE文です。翻訳指示文のCOPY文およびREPLACE文は、手続き部以外にも書くことができます。COPY文およびREPLACE文は、“第7章 原始文操作”で説明します。

条件文は、文の最後に明示範囲符を書くことにより、無条件文にすることができます。「明示範囲符」とは、“END-動詞”の形式の語です。例えば、ON SIZE ERROR指定付きのADD文は、END-ADD指定なしのときは条件文、END-ADD指定付きのときは無条件文です。

無条件文は、そのいくつかを連続して書くことができます。無条件文の間には、分離符を書くこともできます。“書き方”で“無条件文-n”と示しているところには、分離符の終止符を含まない、いくつかの連続した無条件文を書くことができます。

下表に、条件文と無条件文を示します。

文	条件文と無条件の区別
ACCEPT文	ON EXCEPTION指定またはNOT ON EXCEPTION指定付きの場合、条件文(*1) これらのどの指定もない場合、無条件文
ADD文	ON SIZE ERROR 指定またはNOT ON SIZE ERROR 指定付きの場合、条件文(*1) これらのどの指定もない場合、無条件文
ALTER文	無条件文
CALL文	ON OVERFLOW 指定、ON EXCEPTION指定またはNOT ON EXCEPTION指定付きの場合、条件文(*1) これらのどの指定もない場合、無条件文
CANCEL文	無条件文
CLOSE文	無条件文
CLOSE文	無条件文

文	条件文と無条件の区別
COMPUTE文	ON SIZE ERROR 指定またはNOT ON SIZE ERROR 指定付きの場合、条件文(*1) これらのどの指定もない場合、無条件文
CONTINUE文	無条件文
DELETE文	INVALID KEY 指定またはNOT INVALID KEY 指定付きの場合、条件文(*1) これらのどの指定もない場合、無条件文
DISPLAY文	ON EXCEPTION指定またはNOT ON EXCEPTION指定付きの場合、条件文(*1) これらのどの指定もない場合、無条件文
DIVIDE文	ON SIZE ERROR 指定またはNOT ON SIZE ERROR 指定付きの場合、条件文(*1) これらのどの指定もない場合、無条件文
ENTRY文	無条件文
EVALUATE文	条件文(*1)
EXIT文	無条件文
EXIT PERFORM文	無条件文
EXIT PROGRAM文	無条件文
GENERATE文	無条件文
GO TO文	無条件文
IF文	条件文(*1)
INITIALIZE文	無条件文
INITIATE文	無条件文
INSPECT文	無条件文
MERGE文	無条件文
MOVE文	無条件文
MULTIPLY文	ON SIZE ERROR 指定またはNOT ON SIZE ERROR 指定付きの場合、条件文(*1) これらのどの指定もない場合、無条件文
OPEN文	無条件文
PERFORM文	無条件文
READ文	AT END指定、NOT AT END指定、INVALID KEY 指定またはNOT INVALID KEY 指定付きの場合、条件文(*1) これらのどの指定もない場合、無条件文
RELEASE文	無条件文
RETURN文	条件文(*1)
REWRITE文	INVALID KEY 指定またはNOT INVALID KEY 指定付きの場合、条件文(*1) これらのどの指定もない場合、無条件文
SEARCH文	条件文(*1)
SET文	無条件文
SORT文	無条件文
START文	INVALID KEY 指定またはNOT INVALID KEY 指定付きの場合、条件文(*1) これらのどの指定もない場合、無条件文
STOP文	無条件文

文	条件文と無条件の区別
STRING文	ON OVERFLOW 指定またはNOT ON OVERFLOW 指定付きの場合、条件文(*1) これらのどの指定もない場合、無条件文
SUBTRACT文	ON SIZE ERROR 指定またはNOT ON SIZE ERROR 指定付きの場合、条件文(*1) これらのどの指定もない場合、無条件文
SUPPRESS文	無条件文
TERMINATE文	無条件文
UNLOCK文	無条件文
UNSTRING文	ON OVERFLOW 指定またはNOT ON OVERFLOW 指定付きの場合、条件文(*1) これらのどの指定もない場合、無条件文
WRITE文	INVALID KEY 指定、NOT INVALID KEY 指定、END-OF-PAGE 指定またはNOT END-OF-PAGE 指定付きの場合、条件文(*1) これらのどの指定もない場合、無条件文

*1 ： 文の最後に明示範囲符を書くことにより、無条件文になります。

次の実行文でない文に制御の明示的な移行を起こす文を、「手続き分岐文」といいます。手続き分岐文には、以下のものがあります。

- ALTER文
- CALL文
- EXIT文
- EXIT PERFORM文
- EXIT PROGRAM文
- GO TO文
- OUTPUT PROCEDURE指定のMERGE文
- PERFORM文
- INPUT PROCEDURE指定またはOUTPUT PROCEDURE指定のSORT文

完結文

完結文には、完結文を構成する文の種類によって、条件完結文、無条件完結文および翻訳指示完結文の3種類があります。

「条件完結文」は、分離符の終止符で終わる条件文です。1つの条件文の前にいくつかの無条件文をつないで、1つの条件完結文にすることができます。

「無条件完結文」は、分離符の終止符で終わる無条件文です。いくつかの無条件文をつないで、1つの無条件完結文にすることができます。

「翻訳指示完結文」は、分離符の終止符で終わる翻訳指示文です。翻訳指示完結文は、1つの翻訳指示文と分離符の終止符だけで構成しなければなりません。

文の範囲

文の範囲を明に指定するために、明示範囲符または分離符の終止符を書くことができます。

明示範囲符は、以下の語です。

- END-ADD、END-CALL、END-COMPUTE、END-DELETE、END-DIVIDE、END-EVALUATE、END-IF、END-MULTIPLY、END-PERFORM、END-READ、END-RETURN、END-REWRITE、END-SEARCH、END-START、END-STRING、END-SUBTRACT、END-UNSTRING、END-WRITE
- END-ACCEPT、END-DISPLAY

明示範囲符を書くと、明示範囲符に対応する文の範囲が終了します。

以下に、明示範囲符を使って文の範囲を指定する例を示します。

```
IF A = 1 THEN          *> …  [1]                                  --+
  MOVE A TO B          *>                                           | IF文[1]の範囲
  IF X = 1 THEN        *> …  [2]                  --+               |
    MOVE X TO Y        *>                           |IF文[2]の範囲  |
  ELSE                 *>                           |               |
    MOVE Z TO Y        *>                           |               |
  END-IF               *> …  IF文[2]の明示範囲符    --+               |
END-IF                 *> …  IF文[1]の明示範囲符                  --+
```

分離符の終止符を書くと、分離符の終止符の前にあるすべての文の範囲が終了します。

以下に、分離符の終止符を使って文の範囲を指定する例を示します。

```
IF A = 1 THEN          *> …  [3]                                  --+
  MOVE A TO B          *>                                           | IF文[3]の範囲
  IF X = 1 THEN        *> …  [4]                  --+               |
    MOVE X TO Y        *>                           |IF文[4]の範囲  |
  ELSE                 *>                           |               |
    MOVE Z TO Y.       *>                         --+             --+
```

6.2 手続き部の見出し

手続き部の見出しは、手続き部の最初に書きます。

USING指定を書くと、呼ぶプログラムからパラメタを受け取ることができます。

RETURNING指定を書くと、呼ばれたプログラムに結果を返すことができます。

【書き方】

PROCEDURE DIVISION
[WITH { C | PASCAL | STDCALL } LINKAGE]
[USING {データ名-1} …]
[RETURNING データ名-2].

注意

WITH指定は[Win16][Win32]固有機能です。
RETURNING指定は[Win32][WinIPF][Winx64][Solaris][Linux][LinuxIPF][Linux64]固有機能です。

参照

[.NET]の手続き部の見出しは、“11.8.1 手続き部の見出し”を参照してください。

構文規則

1. プログラムがUSING指定付きのCALL文によって呼び出される場合だけ、そのプログラムの手続き部の見出しにUSING指定を書くことができます。

   ただし、このコンパイラでは、実行単位で最初に実行されるプログラムが実行時パラメタを受け取る場合にも、手続き部の見出しにUSING指定を書くことができます。

2. USING指定には、呼ぶプログラムから受け渡されるパラメタに対応するデータ名を指定します。呼ぶプログラムのCALL文のUSING指定に書いた項目と、手続き部の見出しのUSING指定に書いたデータ名との対応は、それぞれのUSINGに指定した順に左から右に対応付けられます。
3. データ名-1は、以下の規則に従わなければなりません。
   a. 連絡節で定義したデータ項目でなければなりません。
   b. データ名-1のデータ項目のレベル番号は、01または77でなければなりません。
   c. データ名-1のデータ記述項に、REDEFINES句を書いてはなりません。ただし、REDEFINES句の右辺には、データ名-1を書くことができます。
4. データ名-1の並びに、同じデータ名を書いてはなりません。
5. USING指定に書くことができるデータ名-1の個数の最大値については、“付録B システムの定量制限”を参照してください。
6. RETURNING指定は、プログラム定義に指定できます。
7. データ名-2は、連絡節のレベル番号01または77のデータ項目として定義しなくてはなりません。データ名-2のデータ記述項は、REDEFINES句を含んではなりません。連絡節の他のデータ項目に、REDEFINES データ名-2を書くこともできます。
8. 手続き部にENTRY文を含む場合、手続き部の見出しにRETURNING指定を書いてはなりません。
9. [WinIPF][LinuxIPF]では、RETURNING指定を手続き部見出しに記述した場合、データ名-2は、同一種類の浮動小数点データ項目から構成される集団項目であってはなりません。

WITH指定の規則

1. WITH指定は、呼ばれるプログラムの呼出し規約を指定します。WITH指定を省略指定した場合、COBOLの定めたリンケージの規約を使用します。
2. 含まれるプログラムの手続き部の見出しにWITH指定を記述してはいけません。
3. WITH指定は、プログラム定義の手続き部の見出しにだけ書くことができます。

一般規則

1. CALL文のBY CONTENT指定のパラメタは、CALL文の実行後、呼び出されたプログラムの中でデータ名-1によって参照することができます。しかし、呼び出されたプログラムの中でデータ名-1に値を設定しても、呼ぶプログラムでその値を受け取ることはできません。CALL文のBY CONTENT指定のデータ項目に対応する、手続き部の見出しのUSING指定のデータ項目は、用途および文字位置の個数が互いに同じでなければなりません。
2. CALL文のBY REFERENCE指定のパラメタは、CALL文の実行後、呼び出されたプログラムの中でデータ名-1によって参照することができます。また、呼び出されたプログラムの中でデータ名-1に値を設定すると、呼ぶプログラムでその値を受け取ることができます。CALL文のBY REFERENCE指定のデータ項目に対応する、手続き部の見出しのUSING指定のデータ項目は、文字位置の個数が互いに同じでなければなりません。
3. CALL文のBY VALUE指定のパラメタは、手続き部の見出しのUSING指定で受け取ることはできません。
4. データ名-1の内容は、つねにデータ名-1のデータ記述に従って参照されます。

5. 呼ばれるプログラムの連絡節で定義したデータ項目は、以下の条件の1つを満足する場合に、手続き部で使うことができます。

   a. 手続き部の

      - 見出し
      - またはENTRY文

      のUSING指定に書いたデータ項目である。

   b. a.の条件を満足するデータ項目に従属するデータ項目である。

   c. a.またはb.の条件を満足するデータ項目を、REDEFINES句またはRENAMES句の右辺に指定したデータ項目である。

   d. c.の条件を満足するデータ項目に従属するデータ項目である。

   e. a.～d.のいずれかの条件を満足するデータ項目に関連する条件名または指標名である。

6. RETURNING指定が書かれた場合、呼び出したプログラムの特殊レジスタPROGRAM-STATUSには値を返却しません。

WITH指定の規則

1. 呼ぶプログラムと呼ばれるプログラムの呼出し規約が異なってはなりません。異なった場合の実行結果は規定しません。
2. 同一翻訳単位内で、最外部プログラムの手続き部の見出しとENTRY文のWITH指定が異なってはなりません。

6.3 文に関する共通の規則

ここでは、文に関する共通の規則、および文に共通に書くことができる指定について説明します。

6.3.1 算術式

算術式は、算術演算子と一意名または定数を組み合わせて書きます。算術式は、以下の5種類です。

1. 以下の中の1つを書いたもの
   - 数字項目
   - 浮動小数点項目
   - 数字関数
   - 整数関数
   - 数字定数
   - 表意定数ZERO
2. 1.の要素を算術演算子でつないだもの
3. 2個の算術式を算術演算子でつないだもの
4. 算術式を括弧で囲んだもの
5. 算術式の前に単項演算子を書いたもの

算術演算子

算術演算子には、二項演算子と単項演算子があります。算術演算子を下表に示します。

分類	演算子	意味
二項演算子	+	加算
	-	減算

分類	演算子	意味
	*	乗算
	/	除算
	**	べき乗 (*1)
単項演算子	+	数字定数の+1を掛けることと同じ
	-	数字定数の-1を掛けることと同じ

*1 ： べき乗の評価の結果、正と負の両方の実数が生じた場合、正の実数がべき乗の値になります。

算術式の書き方の規則

1. 算術式で許される一意名、定数、算術演算子および括弧の組合せを、下表に示します。

先行する要素	後続する要素				
	一意名または定数	(	単項演算子 +,-	二項演算子 +,-,*,/,**	)
単項演算子 +,-	○	○	－	－	－
(	○	○	○	－	－
二項演算子 +,-,*,/,**	○	○	○	－	－
)	－	－	－	○	○
一意名または定数	－	－	－	○	○

○ ： 先行する要素に続いて後続する要素を書くことができます。
－ ： 先行する要素に続いて後続する要素をかくことができません。

2. 算術式は、単項演算子、左括弧、一意名または定数で始まり、右括弧、一意名または定数で終わらなければなりません。
3. 算術式中の一意名は、以下のいずれかでなければなりません。

   － 数字項目
   － 浮動小数点項目
   － 数字関数
   － 整数関数

   算術式中の定数は、数字定数または表意定数ZEROでなければなりません。
4. 算術演算子の前後には、空白を置かなければなりません。ただし、算術演算子と括弧の間の空白は、省略することができます。
5. 左括弧と右括弧は一対一に対応付け、左括弧を右括弧より前に書かなければなりません。算術式中の最初が単項演算子で、その算術式を一意名または他の演算子に続ける場合には、その単項演算子の直前に左括弧を書かなければなりません。

算術式の評価規則

1. 算術式の評価順序は、括弧を使って指定することができます。括弧を書かないときの算術演算子の評価順序は、以下のとおりです。

   － 1番目 ： 単項演算子の＋、－
   － 2番目 ： ＊＊
   － 3番目 ： ＊、／

－ 4番目 ： 二項演算子の＋、－

2. 括弧は、以下の場合に使います。
   － 同じ順位の算術演算子の間で評価順序を変更したい場合
   － 異なる順位の算術演算子の間で評価順序を変更したい場合
3. 括弧を書くと、括弧の中の算術式が先に評価されます。括弧が入れ子になっている場合は、一番内側の括弧の中の算術式が最初に評価され、順次外側が評価されます。
4. 同じ順位の算術演算子が続いている場合、算術演算子は左から右の順に評価されます。

6.3.2 ブール式

ブール式は、ブール演算子と一意名または定数を組み合わせて書きます。ブール式は、以下の5種類です。

1. ブール項目、ブール定数、表意定数ZEROおよびALLブール定数の中の1つを書いたもの
2. 1.の要素をブール演算子でつないだもの
3. 2個のブール式をブール演算子でつないだもの
4. ブール式を括弧で囲んだもの
5. ブール式の前に単項ブール演算子NOTを書いたもの

使い方の例については、“付録H サンプル集”の“H.33 ブール式”を参照してください。

ブール演算子

ブール演算子には、二項ブール演算子と単項ブール演算子があります。ブール演算子を下表に示します。

分類	演算子	意味
二項ブール演算子	AND	ブール積
	OR	ブール和
	EXOR	排他ブール和
単項ブール演算子	NOT	ブール否定

ブール式の書き方の規則

1. ブール式で許される一意名、定数、ブール演算子および括弧の組合せを、下表に示します。

先行する要素	後続する要素				
	一意名または定数	(	単項ブール演算子 NOT	二項ブール演算子 AND,OR,EXOR	)
単項ブール演算子 NOT	○	○	－	－	－
(	○	○	○	－	－
二項ブール演算子 AND,OR,EXOR	○	○	○	－	－
)	－	－	－	○	○
一意名または定数	－	－	－	○	○

○: 先行する要素に続いて後続する要素を書くことができます。
－: 先行する要素に続いて後続する要素をかくことができません。

2. ブール式は、単項ブール演算子、左括弧、一意名または定数で始まり、右括弧、一意名または定数で終わらなければなりません。
3. ブール式中の一意名は、ブール項目でなければなりません。ブール式中の定数は、ブール定数、表意定数ZEROまたはALLブール定数でなければなりません。
4. ブール式に、ブール項目またはブール定数を2つ以上書く場合、それらはすべて同じ長さでなければなりません。
5. ブール演算子の前後には、空白を置かなければなりません。ただし、ブール演算子と括弧の間の空白は、省略することができます。
6. 左括弧と右括弧は一対一に対応付け、左括弧を右括弧より前に書かなければなりません。

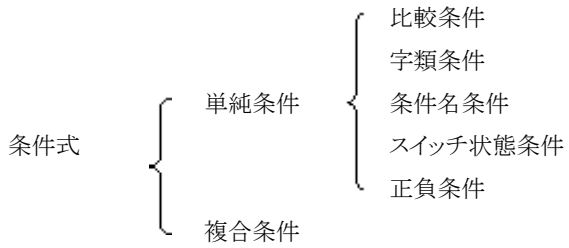
ブール式の評価規則

1. ブール式の評価順序は、括弧を使って指定することができます。括弧を書かないときのブール演算子の評価順序は、以下のとおりです。
   - 1番目: NOT
   - 2番目: AND
   - 3番目: OR、EXOR
2. 括弧は、以下の場合に使います。
   - 同じ順位のブール演算子の間で評価順序を変更したい場合
   - 異なる順位のブール演算子の間で評価順序を変更したい場合
3. 括弧を書くと、括弧の中のブール式が先に評価されます。括弧が入れ子になっている場合は、一番内側の括弧の中のブール式が最初に評価され、順次外側が評価されます。
4. 同じ順位のブール演算子が続いている場合、ブール演算子は左から右の順に評価されます。
5. AND、ORまたはEXORだけでつながれた2個のブール式、およびNOTで始まる1個のブール式におけるAND、OR、EXORおよびNOTはブール演算子です。論理演算子ではありません。

6.3.3 条件式

条件式は、条件が成立している(真である)かまたは成立していない(偽である)かを表す値(真理値)を持ちます。条件式は、真理値を検査することによって、プログラムの流れを制御するために使います。条件式は、EVALUATE文、IF文、PERFORM文およびSEARCH文に書くことができます。

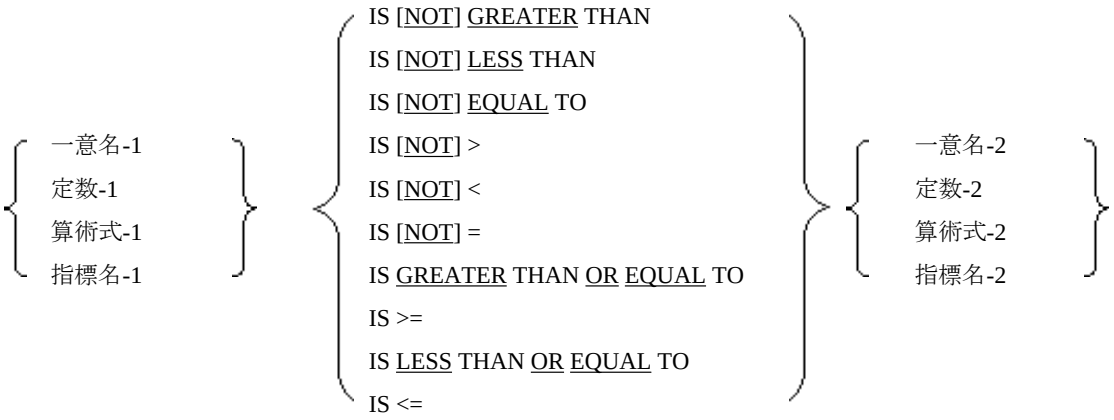
条件式には、以下の種類があります。



6.3.3.1 比較条件

比較条件は、2つの作用対象を比較するために使います。作用対象として、一意名、定数、算術式または指標名を書くことができます。

【書き方】



注意

“＞”、“＜”、“＝”、“＞＝”および“＜＝”は、他の記号との混同を避けるために、下線を付けていません。

1. 比較演算子の左側の作用対象(一意名-1、定数-1、算術式-1または指標名-1)を、「条件の左辺」といいます。比較演算子の右側の作用対象(一意名-2、定数-2、算術式-2または指標名-2)を「条件の右辺」といいます。
2. 比較条件には、1つ以上の変数を書かなければなりません。
3. 比較演算子を構成する予約語の前後には、1つ以上の空白を置かなければなりません。
4. 比較条件では、条件の左辺と右辺を比較演算子に従って比較して、真理値を決定します。比較条件で行われる比較の種類は、比較演算子で決められます。比較演算子の意味を、下表に示します。

比較演算子	意味
IS [NOT] GREATER THAN および IS [NOT] >	左辺が右辺より大きい ［左辺が右辺より大きくない］(*1)
IS [NOT] LESS THAN および IS [NOT] <	左辺が右辺より小さい ［左辺が右辺より小さくない］(*1)
IS [NOT] EQUAL TO および IS [NOT] =	左辺が右辺と等しい ［左辺が右辺と等しくない］(*1)
IS GREATER THAN OR EQUAL TO および IS >=	左辺が右辺より大きいか等しい (*2)
IS LESS THAN OR EQUAL TO および IS <=	左辺が右辺より小さいか等しい (*3)

*1 ： [ ]で囲まれた部分は、NOT を書いた場合を示します。
*2 ： “IS GREATER THAN OR EQUAL TO ”は“IS NOT LESS THAN”と等価であり、“IS >= ”は“IS NOT <”と等価です。
*3 ： “IS LESS THAN OR EQUAL TO”は“IS NOT GREATER THAN ”と等価であり、“IS <= ”は“IS NOT >”と等価です。

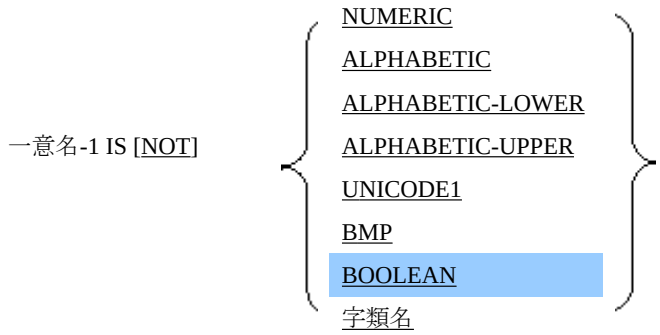
5. 比較条件での比較の規則については、“6.3.4 比較の規則”を参照してください。

6.3.3.2 字類条件

字類条件は、データ項目の内容の字類を検査するために使います。

使い方の例については、“付録H サンプル集”の“H.34 字類条件”を参照してください。

【書き方】



参照

[.NET]の字類条件は、“11.8.2.4.1 字類条件”を参照してください。

1. 字類条件では、字類検査の指定に従って一意名-1の内容を検査し、真理値を決定します。字類検査が真になる条件を、下表に示します。

字類検査の種類	字類検査の種類
NUMERIC 検査	一意名-1の内容が、数字0,1,2,3,…,9だけからなる、またはこれらに演算符号を付けたものである場合 (*1)
ALPHABETIC検査	一意名-1の内容が、英大文字A,B,C,…,Zと空白からなる、または英小文字a,b,c,…,zと空白からなる、または英大文字と英小文字と空白の組合せからなる場合
ALPHABETIC-LOWER 検査	一意名-1の内容が、英小文字a,b,c,…,zと空白からなる場合
ALPHABETIC-UPPER 検査	一意名-1の内容が、英大文字A,B,C,…,Zと空白からなる場合
UNICODE1検査	一意名-1の内容が、UNICODE1.1(ISO/IEC 10646-1:1993)で規定されている文字からなる場合
BMP検査	一意名-1の内容が、ISO/IEC 10646-1のBMP(Basic Multilingual Planeの略で0群0面を指す)で規定されている文字からなる場合
BOOLEAN 検査	一意名-1の内容が、ブール文字0,1だけからなる場合
字類名検査	一意名-1の内容が、特殊名段落のCLASS 句で指定した文字の組からなる場合

*1 ： 詳細は、6.を参照してください。

2. 一意名-1は、以下のいずれかでなければなりません。
   - 用途が表示用のデータ項目
   - 英数字関数または日本語関数の関数一意名
   - 内部10進項目

3. NUMERIC検査の場合、一意名-1に以下のデータ項目を指定することはできません。
   - 英字項目
   - 符号付き数字項目を従属する集団項目
   - 日本語項目
   - 日本語編集項目
   - ブール項目
4. ALPHABETIC検査、ALPHABETIC-LOWER検査、ALPHABETIC-UPPER検査および字類名検査の場合、一意名-1に以下のデータ項目を指定することはできません。
   - 数字項目
   - 日本語項目
   - 日本語編集項目
   - ブール項目
5. BOOLEAN検査の場合、一意名-1に以下のデータ項目を指定することはできません。
   - 数字項目
   - 英字項目
   - 日本語項目
   - 日本語編集項目
6. NUMERIC検査が真になる条件は、以下のとおりです。
   a. 一意名-1のデータ項目に演算符号を指定しなかった場合、その内容が数字だけからなり、かつ演算符号がないとき、真になります。一意名-1が内部10進項目の場合は、その符号部分が16進表現のFのとき真になります。
   b. 一意名-1のデータ項目に演算符号を指定した場合、その内容が数字だけからなり、かつ有効な演算符号が存在するとき、真になります。有効な演算符号が存在するときとは、以下の場合です。
      - 一意名-1のデータ項目のSIGN句にSEPARATE CHARACTERを指定した場合、その符号部分が標準データ形式の“＋”または“－”のとき。
      - 一意名-1が外部10進項目で、そのSIGN句にSEPARATE CHARACTERを指定しなかった場合、その符号部分が16進表現の4、5または3のとき。
      - 一意名-1が内部10進項目の場合、その符号部分が16進表現のC、DまたはFのとき。
7. NOTを書いた場合、NOTと必要語で1つの字類条件になります。NOTを書くと、真理値の真/偽がNOTを書かなかった場合の逆になります。例えば“NOT NUMERIC”は、作用対象が数字でないとき真になります。
8. UNICODE1検査およびBMP検査の場合、一意名-1に以下のデータ項目を指定することはできません。
   - 数字項目
   - ブール項目
9. UNICODE1検査およびBMP検査は、動作モードがUnicodeの場合だけ使用できます。

6.3.3.3 条件名条件

条件名条件は、条件変数の値が条件名の値と等しいかどうかを調べるために使います。

【書き方】

条件名-1

---

1. 条件名条件が真になる条件は、以下のとおりです。

   a. 条件名-1のデータ記述項のVALUE句にTHRUを書かなかった場合、条件変数の値がVALUE句で指定した値と等しいとき真になります。

   b. 条件名-1のデータ記述項のVALUE句にTHRUを書いた場合、条件変数の値が、VALUE句で指定した値の範囲内(THRUで指定した両端の値も含む)にあるとき真になります。

2. 条件変数の値を条件名-1の値と比較するときの規則は、比較条件での比較の規則に従います。比較の規則については、“6.3.4 比較の規則”を参照してください。

3. 条件名条件の例を、以下に示します。

   a. 条件変数および条件名を、以下のように定義したとします。

```
02  MONTH  PICTURE  99.              *>          条件変数(MONTH)の定義
  88  SPRING-M  VALUE  3  THRU   5.  *>      --+
  88  SUMMER-M  VALUE  6  THRU   8.  *>        | 条件名(SPRING-M、SUMMER-M、FALL-Mおよび
  88  FALL-M    VALUE  9  THRU  11.  *>        | WINTER-M)の定義
  88  WINTER-M  VALUE  12, 1, 2.     *>      --+
```

   b. 条件変数MONTHの値を検査するために、条件名条件を使って以下のIF文を書くことができます。

```
[1] IF SPRING-M  …    →  IF MONTH  >=  3  AND  <=  5  と等価
[2] IF SUMMER-M  …    →  IF MONTH  >=  6  AND  <=  8  と等価
[3] IF FALL-M    …    →  IF MONTH  >=  9  AND  <=  11  と等価
[4] IF WINTER-M  …    →  IF MONTH  =  12  OR  1  OR  2  と等価
```

6.3.3.4 スイッチ状態条件

スイッチ状態条件は、外部スイッチのオンとオフの状態を調べるために使います。

【書き方】

条件名-1

---

1. 外部スイッチは環境部の特殊名段落で定義し、そのオン状態またはオフ状態に条件名を対応付けなければなりません。

2. 外部スイッチが条件名-1の状態に設定されている場合、スイッチ状態条件が真になります。

6.3.3.5 正負条件

正負条件は、算術式の代数値がゼロより大きいか、小さいかまたは等しいかを調べるために使います。

【書き方】

```
算術式-1 IS [NOT] { POSITIVE }
                  { NEGATIVE }
                  { ZERO     }
```

---

1. 算術式-1には、少なくとも1つの一意名を書かなければなりません。

2. 正負条件では、算術式の代数値を検査し、POSITIVE、NEGATIVEまたはZEROの指定に従って、真理値を決定します。これらの必要語の意味を、下表に示します。

必要語	意味
POSITIVE	ゼロより大きい
NEGATIVE	ゼロより小さい
ZERO	ゼロに等しい

3. NOTを書いた場合、NOTと必要語で1つの正負条件になります。NOTを書くと、真理値の真/偽がNOTを書かなかった場合の逆になります。例えば、“NOT ZERO”は、算術式の値がゼロでないとき(正または負のとき)真になります。

6.3.3.6 複合条件

複合条件は、単純条件と論理演算子を組み合わせて書きます。

複合条件は、否定条件と組合せ条件で構成します。否定条件は、単純条件または複合条件の前に、論理演算子NOTを書いたものです。組合せ条件は、単純条件または複合条件を、論理演算子ANDまたはORでつないだものです。

複合条件の真理値は、個々の条件の真理値を評価し、すべての論理演算子を順に作用させた結果の真理値です。

論理演算子の意味を、下表に示します。

論理演算子	意味	真理値
NOT	論理否定	・ NOTの後の条件が偽の場合、真理値は真 ・ NOTの後の条件が真の場合、真理値は偽
AND	論理積	・ ANDの左辺と右辺の両方の条件が真の場合、真理値は真 ・ ANDの左辺と右辺の一方または両方の条件が偽の場合、真理値は偽
OR	論理和	・ ORの左辺と右辺の一方または両方の条件が真の場合、真理値は真 ・ ORの左辺と右辺の両方の条件が偽の場合、真理値は偽

否定条件

否定条件は、条件の真理値を逆転するために使います。

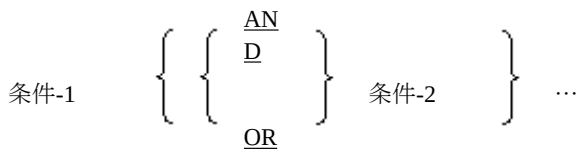
【書き方】

NOT 条件-1

組合せ条件

組合せ条件は、論理積または論理和を求めるために使います。

【書き方】



複合条件の書き方の規則

1. 複合条件で許される単純条件、論理演算子および括弧の組合せを、下表に示します。

先行する要素	後続する要素					
	単純条件	AND	OR	NOT	(	)
単純条件	－	○	○	－	－	○
AND	○	－	－	○	○	－
OR	○	－	－	○	○	－
NOT	○	－	－	－	○	－
(	○	－	－	○	○	－
)	－	○	○	－	－	○

○ ： 先行する要素に続いて後続する要素を書くことができます。
－ ： 先行する要素に続いて後続する要素をかくことができません。

2. 複合条件は、単純条件、NOTまたは“(”で始まり、単純条件または“)”で終わらなければなりません。
3. 論理演算子の前後には、空白を置かなければなりません。ただし、論理演算子と括弧の間の空白は、省略することができます。
4. 左括弧と右括弧は一対一に対応付け、左括弧を右括弧より前に書かなければなりません。

複合条件の評価規則

1. 複合条件の評価順序は、括弧を使って指定することができます。括弧を書かないときの論理演算子の評価順序は、以下のとおりです。

   － 1番目 ： NOT

   － 2番目 ： AND

   － 3番目 ： OR

2. 論理演算子の評価順序を変更したい場合、論理演算子による条件の結合範囲を括弧で囲みます。括弧を書くと、論理演算子によって結合された条件および括弧の中の論理演算子が、先に評価されます。括弧が入れ子になっている場合は、一番内側の括弧の中の条件と論理演算子が最初に評価され、順次外側が評価されます。
3. 同じ順位の論理演算子が続いている場合、論理演算子は左から右の順に評価されます。
4. 複合条件の中の個々の条件の評価は、すべての条件が評価されているかどうかに関係なく、複合条件の真理値が決まったときに終了します。
5. 複合条件の中の算術式の値および関数値は、それらを書いた条件が評価されるときに決まります。
6. 否定条件は、論理演算子NOTと連結された条件が評価されるときに評価されます。

複合条件の評価順序の例

図6.1 “条件-1 AND 条件-2 AND … 条件-n”の評価順序

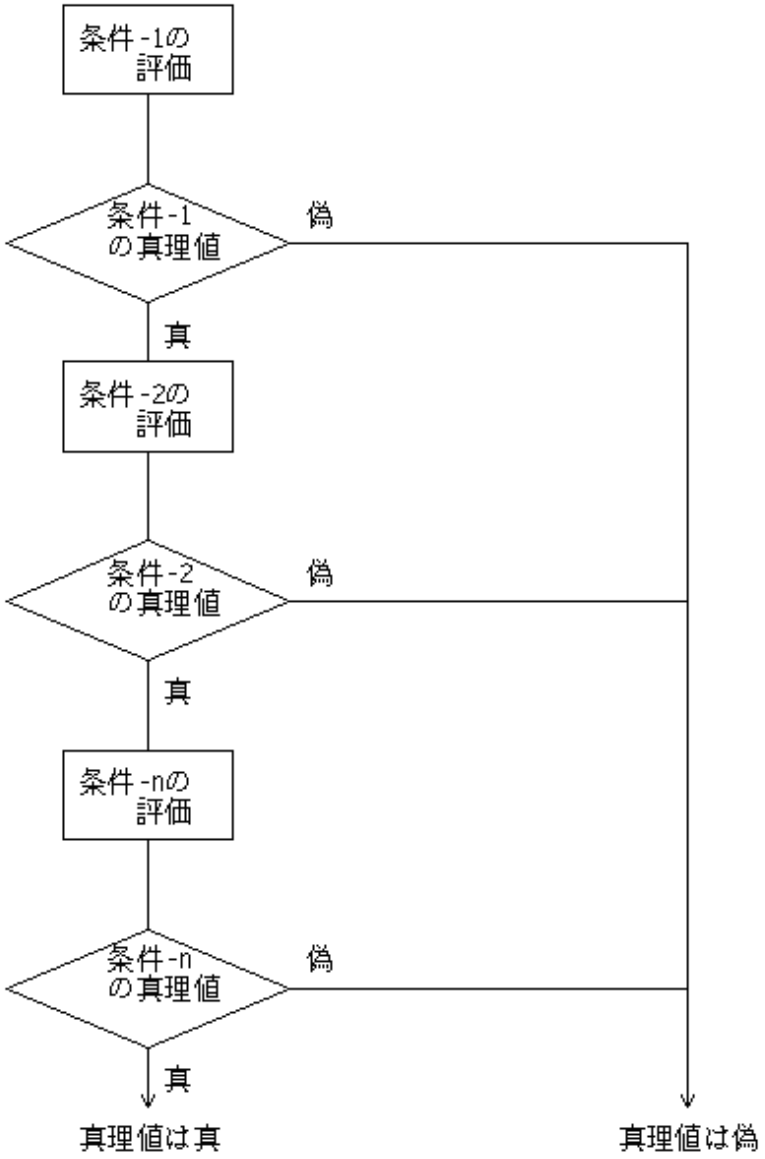


図6.2 “条件-1 OR 条件-2 OR … 条件-n”の評価順序

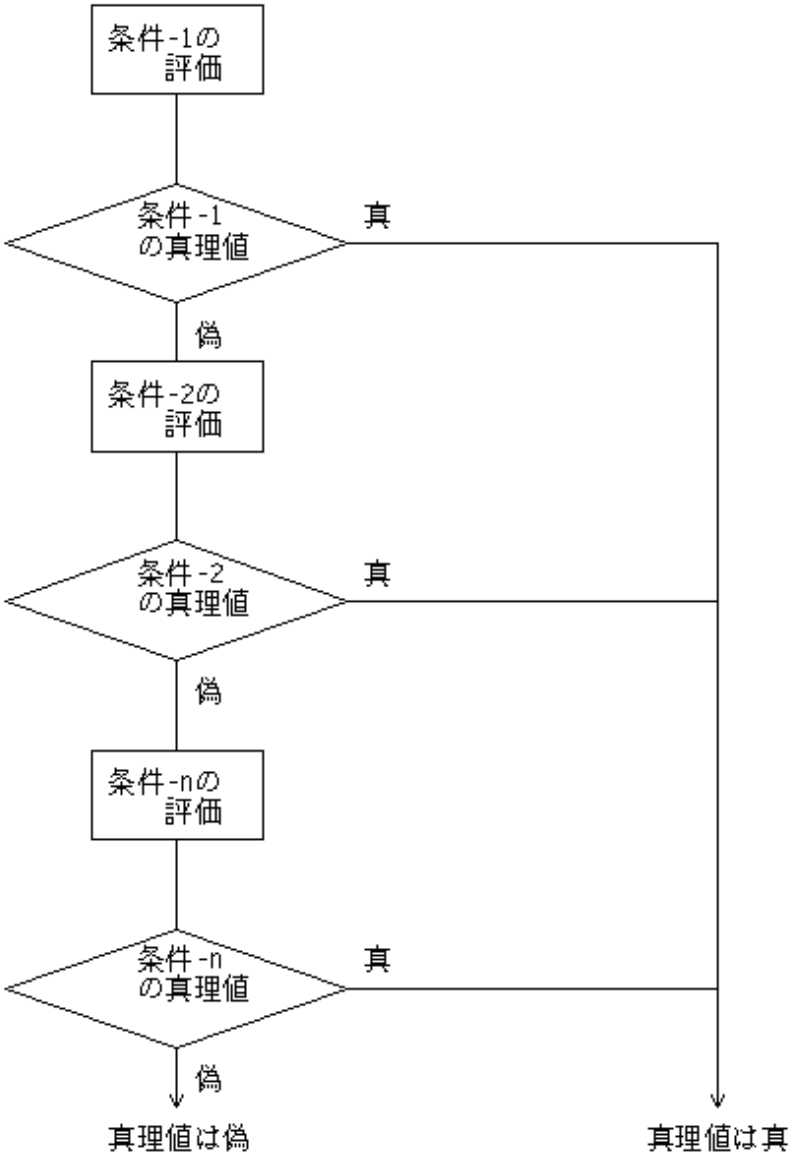


図6.3 “条件-1 OR 条件-2 AND 条件-3”の評価順序

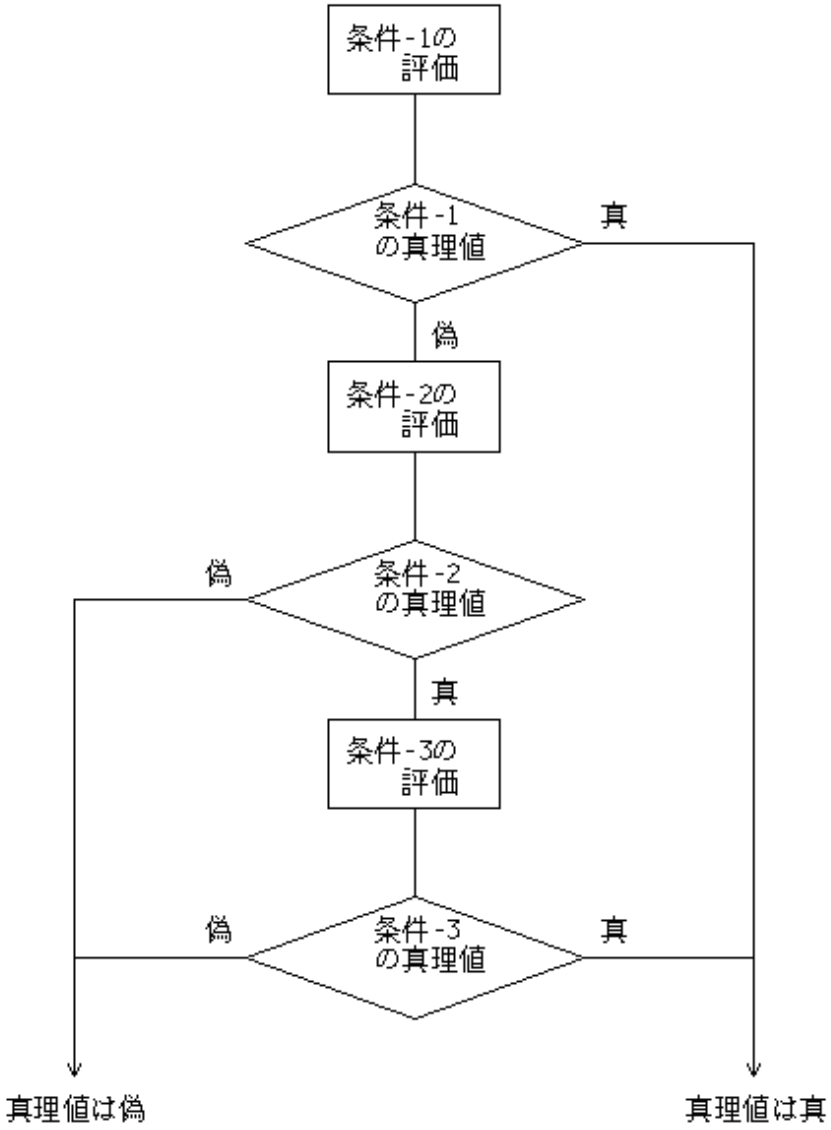
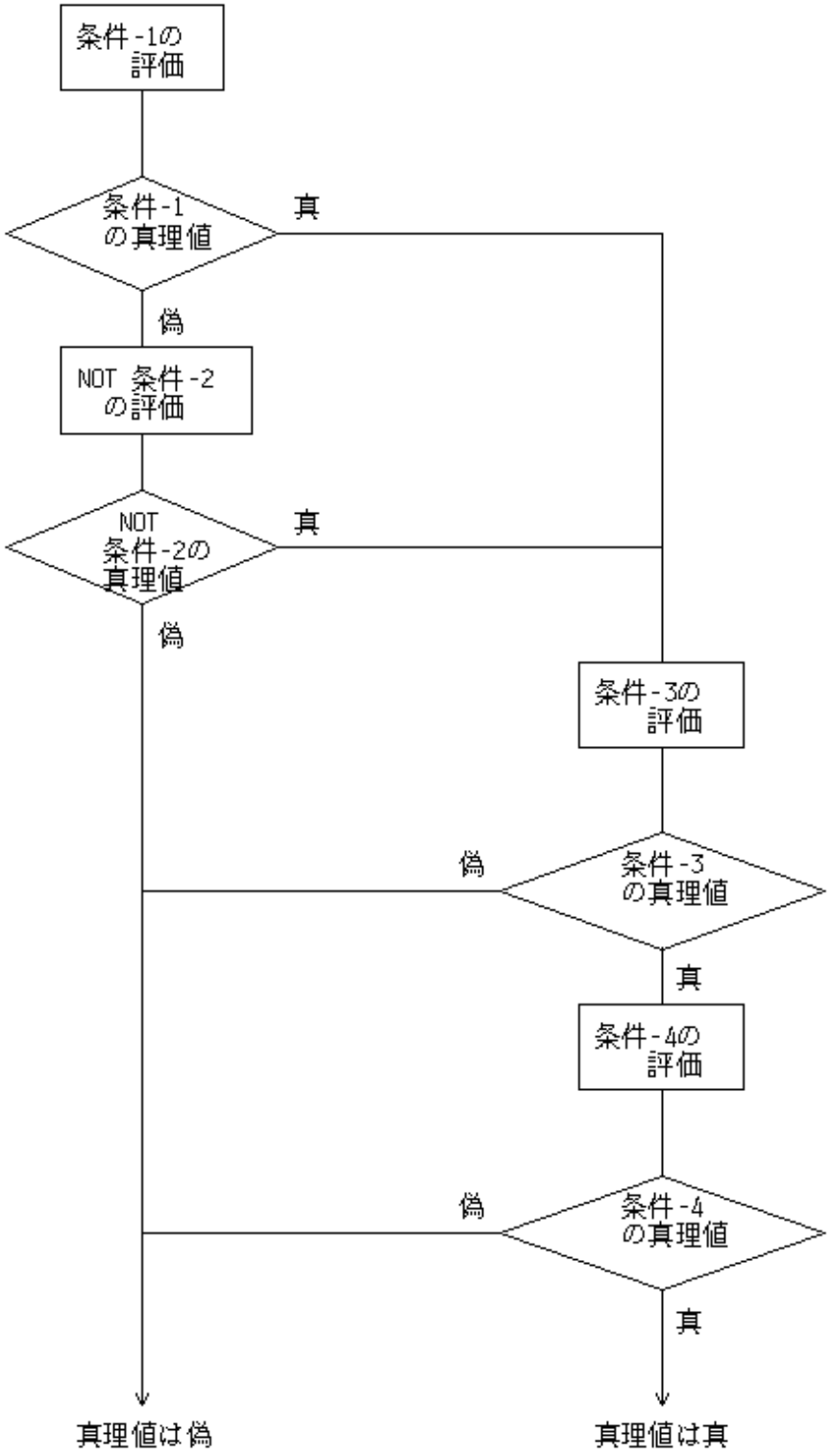


図6.4 “(条件-1 OR NOT 条件-2) AND 条件-3 AND 条件-4”の評価順序



6.3.3.7 組合せ比較条件の略記法

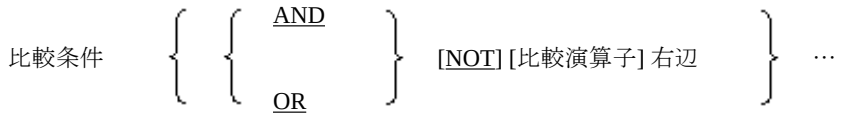
比較条件と論理演算子を組み合わせた複合条件を、「組合せ比較条件」といいます。組合せ比較条件の中に、論理演算子の評価順序を変更するための括弧がない場合、組合せ条件の一部を省略して書くことができます。

組合せ条件の省略方法として、以下の2つがあります。

- 後続の比較条件の左辺が直前のものと同じ場合、後続の比較条件の左辺を省略することができます。
- 後続の比較条件の左辺と比較演算子の両方が直前のものと同じ場合、後続の比較条件の左辺と比較演算子の両方を省略することができます。

使い方の例については、“付録H サンプル集”の“H.35 組合せ比較条件の略記法”を参照してください。

【書き方】



---

1. 一連の比較条件の中で、2つの省略方法の両方を使うことができます。
2. 省略した左辺には、その直前に書いた左辺が補われます。省略した比較演算子には、その直前に書いた比較演算子が補われます。省略した左辺と比較演算子の補充は、複合条件の中で略記されていない単純条件が現れるまで繰り返されます。左辺と比較演算子が補われた結果は、“6.3.3.6 複合条件”で説明した規則に従っていなければなりません。
3. 略記した組合せ比較条件の中にNOTを書いた場合、NOTは以下のように解釈されます。
    a. NOTの直後にGREATER、＞、LESS、＜、EQUAL、＝のいずれかが続く場合、NOTはその比較演算子の一部であるとみなされます。
    b. a.以外のNOTは、論理演算子であるとみなされます。したがって、省略した左辺と比較演算子が補われた結果は、否定条件になります。
4. 略記した組合せ比較条件の例を、下表に示します。

組合せ比較条件を略記した書き方	組合せ比較条件を略記しない書き方
a > b AND NOT < c OR d	((a > b) AND (a NOT < c)) OR (a NOT < d)
a NOT EQUAL b OR c	(a NOT EQUAL b) OR (a NOT EQUAL c)
NOT a = b OR c	(NOT (a = b)) OR (a = c)
NOT (a GREATER b OR < c)	NOT ((a GREATER b) OR (a < c))
NOT (a NOT > b AND c AND NOT d)	NOT (((a NOT > b) AND (a NOT > c)) AND (NOT (a NOT > d)))

__ ： 省略できる左辺および比較演算子

6.3.4 比較の規則

ここでは、比較条件の比較の規則について説明します。比較では、比較条件の左辺と右辺の組合せによって、以下のいずれかの規則が適用されます。

- 文字比較
- 数字比較
- 日本語文字比較
- ブール比較
- ポインタデータ比較
- 指標比較

ブール比較の規則は、比較条件の左辺と右辺のいずれか一方がブール項目の場合に適用されます。ポインタデータ比較の規則は、比較条件の左辺と右辺のいずれか一方がポインタデータ項目またはADDR関数の場合に適用されます。

指標比較の規則は、比較条件の左辺と右辺のいずれか一方が指標名または指標データ項目の場合に適用されます。

文字比較、数字比較および日本語文字比較の規則が適用される場合の作用対象の組合せは、下表のとおりです。

参照

[.NET]の作用対象の組合せは、“11.8.2.5 比較の規則”を参照してください。

条件の右辺/左辺	条件の左辺/右辺						
	集団	英字 英数字(*1) 英数字編集 数字編集	日本語 日本語編集	外部10進	2進 内部10進	浮動小数点	算術式(*2)
集団項目	文字比較	文字比較	文字比較	文字比較	文字比較	文字比較	―
英字項目 英数字項目(*1) 英数字編集項目 数字編集項目 文字定数(*3)	文字比較	文字比較	―	文字比較	―	―	―
日本語項目(*4) 日本語編集項目 日本語定数(*5)	文字比較	―	日本語文字比較	―	―	―	―
表意定数 SPACE HIGH-VALUE LOW-VALUE	文字比較	文字比較	日本語文字比較	文字比較	―	―	―
表意定数 QUOTE 記号文字	文字比較	文字比較	―	文字比較	―	―	―
外部10進項目	文字比較	文字比較	―	数字比較	数字比較	数字比較	数字比較
2進項目 内部10進項目	文字比較	―	―	数字比較	数字比較	数字比較	数字比較
数字定数	文字比較	文字比較(*6)	―	数字比較	数字比較	数字比較	数字比較
表意定数 ZERO	文字比較	文字比較	―	数字比較	数字比較	数字比較	数字比較
浮動小数点項目 浮動小数点定数	文字比較	―	―	数字比較	数字比較	数字比較	数字比較
算術式(*2)	―	―	―	数字比較	数字比較	数字比較	数字比較

－ ： 比較できない組み合わせです。

*1 ： 英数字項目は、英数字関数を含みます。

*2 ： 算術式は、数字関数および整数関数を含みます。

*3 ： 文字定数は、ALL文字定数を含みます。

*4 ： 日本語項目は、日本語関数を含みます。

*5 ： 日本語定数は、ALL日本語定数を含みます。

*6 ： 英数字関数と数字定数を比較することはできません。

文字比較

文字比較の規則は、以下のとおりです。

1. 2つの作用対象が、文字の大小順序に従って比較されます。
2. 2つの作用対象の文字位置の個数が等しい場合、対応する文字位置の文字が、最左端から最右端に向かって一組ずつ順に比較されます。比較の結果は、以下のように決定されます。
    a. 対応する文字がすべて等しい場合、2つの作用対象は等しいという結果になります。
    b. 対応する文字が等しくない組が最初に現れたとき、文字の大小順序で高い位置の文字を含む作用対象の方が大きいという結果になります。
3. 2つの作用対象の文字位置の個数が異なる場合、短い方の作用対象の右側には、長い方の作用対象の文字位置の個数と同じになるまで空白があるものとみなされます。比較の方法は、2.と同じです。
4. 数字作用対象
    － 外部10進項目、2進項目、内部10進項目、数字定数、表意定数ZERO
    － 浮動小数点項目、浮動小数点定数
    を文字作用対象
    － 集団項目、英字項目、英数字項目、英数字編集項目、数字編集項目、文字定数、表意定数SPACE、HIGH-VALUE、LOW-VALUE、QUOTE、記号文字
    － 日本語項目、日本語編集項目、日本語定数
    と比較する場合、以下の規則が適用されます。
    a. 数字作用対象は整数でなければなりません。
    b. 文字作用対象が基本項目または文字定数の場合、数字作用対象が文字作用対象と同じ大きさの英数字項目に転記したかのように扱われます。この英数字項目が文字作用対象と比較されます。
    c. 文字作用対象が集団項目の場合、まず数字作用対象が文字作用対象と同じ大きさの集団項目に転記したかのように扱われます。次に、この集団項目が文字作用対象と比較されます。
5. 作用対象のどちらかが強く型付けされた集団項目である場合、もう一方の作用対象は同じ型で強く型付けられた集団項目でなければなりません。
6. 文字の大小順序については“4.2.2.2 PROGRAM COLLATING SEQUENCE句”を参照してください。

数字比較

数字比較の規則は以下のとおりです。

1. 2つの作用対象が代数的な値に基づいて比較されます。
2. 値ゼロは、符号の有無に関係なくゼロとして比較されます。
3. 符号のない作用対象は、その符号部が正であるとみなされて比較されます。

日本語文字比較

日本語文字比較の規則は、以下のとおりです。

1. 2つの作用対象が、日本語文字の大小順序に従って比較されます。
2. 2つの作用対象の日本語文字位置の個数が等しい場合、対応する日本語文字位置の日本語文字が、最左端から最右端に向かって一組ずつ順に比較されます。比較の結果は、以下のように決定されます。
    a. 対応する日本語文字がすべて等しい場合、2つの作用対象は等しいという結果になります。
    b. 対応する日本語文字が等しくない組が最初に現れたとき、日本語文字の大小順序で高い位置の日本語文字を含む作用対象の方が大きいという結果になります。
3. 2つの作用対象の日本語文字位置の個数が異なる場合、短い方の作用対象の右側には、長い方の作用対象の日本語文字位置の個数と同じになるまで日本語空白があるものとみなされます。比較の方法は2.と同じです。

ブール比較

ブール比較の規則は、以下のとおりです。

1. ブール比較では、ブール項目を以下の作用対象と比較することができます。
    － ブール定数(ALLブール定数を含む)
    － 表意定数ZERO
2. ブール項目どうしを比較する場合、その用途は同じである必要はありません。
3. 比較演算子は、以下のいずれかでなければなりません。
    － IS〔NOT〕EQUAL TO
    － IS〔NOT〕＝
4. 2つの作用対象の長さ(ブール文字の数)は同じでなければなりません。
5. 2つの作用対象の対応するブール位置のブール文字が、最左端から最右端に向かって一組ずつ順に比較されます。対応するブール文字がすべて等しい場合、2つの作用対象は等しいという結果になります。
6. 作用対象としてブール式を書くことはできません。

ポインタデータ比較

ポインタデータ比較の規則は、以下のとおりです。

1. ポインタデータ比較では、ポインタデータ項目またはADDR関数を、以下の作用対象と比較することができます。
    － ポインタデータ項目
    － ADDR関数
    － 表意定数ZERO
2. ポインタデータ比較は、IF文またはEVALUATE文の比較条件にだけ書くことができます。
3. 比較演算子は、以下のいずれかでなければなりません。
    － IS〔NOT〕EQUAL TO
    － IS〔NOT〕＝

指標比較

指標比較の規則を、下表に示します。

条件の左辺/右辺	条件の右辺/左辺			
	指標名	指標データ項目	数字定数（整数だけ）	数字項目（整数だけ）
指標名	出現番号の比較	変換なしの比較	出現番号と整数の比較	出現番号と整数の比較
指標データ項目	変換なしの比較	変換なしの比較	－	－

－ ： 比較できない組み合わせです。
出現番号の比較 ： 指標名に対応する出現番号同士が比較されます。
出現番号と整数の比較 ： 指標名に対応する出現番号が、他の作用対象と比較されます。
変換なしの比較 ： 実際の値がそのまま比較されます。

6.3.5 転記の規則

ここでは、MOVE文の転記の規則を説明します。

文の実行によって、データ項目、定数または算術演算の結果がデータ項目に転記されることがあります。転記の規則は、明に書いたMOVE文だけでなく、このような暗黙のMOVE文でも適用されます。

転記の規則には、基本項目転記と集団項目転記があります。送出し側と受取り側のいずれか一方または両方が集団項目の場合、集団項目転記の規則が適用されます。そうでない場合、基本項目転記の規則が適用されます。

基本項目転記

基本項目転記では、受取り側の項類および用途によって、以下のいずれかの規則が適用されます。

- 英字転記
- 英数字・英数字編集転記
- 数字・数字編集転記
- 浮動小数点転記
- 日本語・日本語編集転記
- ブール転記
- ポインタデータ転記

基本項目転記では、必要ならば、内部表現形式の変換、編集または逆編集が行われます。

基本項目転記の作用対象の組合せを、下表に示します。

送り出し側		受け取り側						
		英字	英数字 英数字編集	数字 数字編集	浮動小数点	日本語 日本語編集	ブール	ポインタデータ
英字項目		[1]	[2]	－	－	－	－	－
英数字項目 (*1) 文字定数 (*2)		[1]	[2]	[3]	[4]	－	[6] (*8)	－
英数字編集項目		[1]	[2]	－	－	－	－	－
数字項目 数字定数	整数	－	[2] (*9)	[3]	[4]	－	－	－
	非整数	－	－	[3]	[4]	－	－	－
数字編集項目		－	[2]	[3]	[4]	－	－	－
浮動小数点項目 浮動小数点定数		－	－	[3]	[4]	－	－	－
日本語項目 (*3)		－	－	－	－	[5]	－	－

送り出し側	受け取り側						
	英字	英数字 英数字編集	数字 数字編集	浮動小数点	日本語 日本語編集	ブール	ポインタデータ
日本語編集項目 日本語定数 (*4)							
ブール項目 ブール定数 (*5)	―	[2] (*7)	―	―	―	[6]	―
ポインタデータ項目 (*6)	―	―	―	―	―	―	[7]
表意定数ZERO	―	[2]	[3]	[4]	―	[6]	[7]
表意定数SPACE	[1]	[2]	―	―	[5]	―	―
表意定数 HIGH-VALUE LOW-VALUE	―	[2]	―	―	[5]	―	―
表意定数QUOTE 記号文字	―	[2]	―	―	―	―	―

[1]～[7] ： 転記できる組合わせです。[1]～[7]は、以下の見出しの項番に対応します。

― ： 転記できない組合わせです。

*1 ： 英数字項目は、英数字関数を含みます。

*2 ： 文字定数は、ALL文字定数を含みます。

*3 ： 日本語項目は、日本語関数を含みます。

*4 ： 日本語定数は、ALL日本語定数を含みます。

*5 ： ブール定数は、ALLブール定数を含みます。

*6 ： ポインタデータ項目は、ADDR関数を含みます。

*7 ： ブール項目を英数字編集項目に転記することはできません。

*8 ： 文字定数をブール項目に転記することはできません。

*9 ： 送出し側がint型2進整数データ項目の場合は、転記できない組合せです。

英字転記:[1]

受取り側が英字項目の場合、標準桁よせ規則に従って、桁よせおよび必要な空白づめが行われます。

英数字・英数字編集転記:[2]

受取り側が英数字項目または英数字編集項目の場合、以下の規則が適用されます。

- 標準桁よせ規則に従って、桁よせおよび必要な空白づめが行われます。
- 送出し側が符号付き数字項目の場合、符号は転記されません。送出し側項目のSIGN句にSEPARATE指定を書いた場合、符号は転記されないので、送出し側の桁数が編集データ形式で1桁小さいものとして転記されます。
- 送出し側が数字編集項目の場合、逆編集(編集文字の除去)は行われません。
- 送出し側の用途が受取り側の用途と異なる場合、送出し側が受取り側の内部表現に変換されます。
- 送出し側が数字項目でPICTURE句に“P”を含む場合、“P”が示す桁位置はゼロであるとみなされます。“P”は送出し側の大きさに数えます。

数字・数字編集転記:[3]

受取り側が数字項目または数字編集項目の場合、以下の規則が適用されます。

- 標準桁よせ規則に従って、小数点の位置合わせおよび必要なゼロづめが行われます。そのゼロは、PICTURE句の記述に従って、他の文字に変換されることがあります。

  ただし、このコンパイラでは、翻訳オプションTRUNCを指定する必要があります。翻訳オプションTRUNCを指定せずに受取り側に2進項目を書いた場合、左端が切り捨てられないことがあります。左端が切り捨てらないために受取り側の項目に収まらなかった場合、転記の結果は保証されません。

- 送出し側が数字編集項目の場合、まず逆編集が行われ、編集される前の符号のない数値が求められます。その編集される前の数値が、受取り側に転記されます。
- 受取り側が符号付き数字項目の場合、受取り側には送出し側の符号と同じ符号が付けられます。このとき、必要ならば符号の表現形式が変換されます。送出し側に符号がない場合は、受取り側に正の符号が付けられます。
- 受取り側が符号なし数字項目の場合、送出し側の絶対値が転記されます。受取り側には符号は付けられません。
- 送出し側の項類が英数字の場合、送出し側のデータは符号なし整数とみなされて転記されます。
- 送出し側が浮動小数点項目かつ保持する値が受取り側より大きな小数部の桁数を持つ場合、受取り側の小数部の桁数＋1の位で四捨五入した値が転記されます。

浮動小数点転記:[4]

受取り側が浮動小数点項目の場合、非整数の数字項目への転記と同様に扱われます。

日本語・日本語編集転記:[5]

受取り側が日本語項目または日本語編集項目の場合、標準桁よせ規則に従って、桁よせおよび必要な空白づめが行われます。

ブール転記:[6]

受取り側がブール項目の場合、以下の規則が適用されます。

- 標準桁よせ規則に従って、桁よせおよび必要なブール文字のゼロづめが行われます。
- 送出し側が英数字項目の場合、送出し側は外部ブール項目として扱われます。

ポインタデータ転記:[7]

受取り側がポインタデータ項目の場合、以下の規則が適用されます。

- 送出し側がポインタデータ項目またはADDR関数の場合、送出し側の内容がそのまま転記されます。
- 送出し側が表意定数ZEROの場合、数値0が転記されます。

集団項目転記

送出し側または受取り側のどちらか一方または両方が集団項目の場合、以下の規則が適用されます。

- ブール項目、ポインタデータ項目または浮動小数点定数を、集団項目に転記することはできません。
- 集団項目を、ブール項目またはポインタデータ項目に転記することはできません。
- 集団項目転記は、英数字項目どうしの基本項目転記と同様に行われます。
- 内部表現形式の変換は行われません。
- 集団項目に従属する個々の基本項目と集団項目は考慮されません。集団項目全体が1つの英数字項目であるかのように転記されます。ただし、集団項目または集団項目に従属するデータ項目にOCCURS句を指定した場合は、別の規則が適用されます。OCCURS句を含む集団項目の転記の規則については、“5.4.6 OCCURS句”を参照してください。

6.3.6 算術文

ADD文、COMPUTE文、DIVIDE文、MULTIPLY文およびSUBTRACT文の5つを総称して、「算術文」といいます。算術文に共通する規則は、以下のとおりです。

1. 算術文の作用対象のデータ記述項は、同一である必要はありません。計算の過程において、必要な変換と小数点の位置合わせが行われます。
2. 算術演算の過程で、一時的な演算結果を格納するためのデータ項目が必要になることがあります。この一時的なデータ項目を「中間結果」といいます。中間結果のための記憶領域は、符号付き数字項目として、コンパイラによって用意されます。中間結果の桁数は、“付録D 中間結果”で説明する算法に従って決定されます。実行時、中間結果に一時的に格納された演算結果は、MOVE文の規則に従って、結果を格納するためのデータ項目に転記されます。

6.3.7 算術文における複数個の答

算術文には、結果の一意名(結果を格納するためのデータ項目)を2つ以上書くことができます。この場合、算術文の結果は以下の順に計算されます。

1. 文の初期評価の対象となっているすべてのデータ項目に対して、必要な計算が行われます。そして、その結果が一時的なデータ項目に格納されます。
2. 次に、結果の一意名の1つ1つに対して、1.で求めた一時的なデータ項目との計算が行われ、結果が格納されます。この計算は、結果の一意名を指定した順に、左から右に行われます。

複数個の答を計算する例を、以下に示します。tempは、コンパイラが用意した一時的な記憶領域を表します。

例

例1

“ADD a b c TO c d(c) e ”の計算のしかたは、以下の文を順に実行した場合と同じです。

```
ADD a b c GIVING temp
ADD temp TO c
ADD temp TO d(c)          …  c の値は、直前の加算で変更された値です。
ADD temp TO e
```

例2

“MULTIPLY a(i) BY i a(i) ”の計算のしかたは、以下の文を順に実行した場合と同じです。

```
MOVE a(i) TO temp
MULTIPLY temp BY i
MULTIPLY temp BY a(i)    …  i の値は、直前の乗算で変更された値です。
```

6.3.8 ROUNDED指定

算術文には、ROUNDED指定を書くことができます。

算術演算の結果の小数部の桁数が、結果の一意名の小数部の桁数より大きい場合、ROUNDED指定の有無に従って、以下の処理が行われます。

1. ROUNDED指定を書かなかった場合、算術演算の結果の小数部が、結果の一意名の桁数に合わせて切り捨てられます。
2. ROUNDED指定を書いた場合、切捨て部分の最上位の桁の値が5以上のとき、結果の一意名の最下位の桁の絶対値が1増やされます。

結果の一意名の整数部の下位桁をPICTURE句の文字“P”で定義した場合、四捨五入および切捨ては、実際に記憶領域を割り当てられている部分の、最右端の整数部に対して行われます。

注意

算術演算の結果が浮動小数点、かつ結果の一意名が固定小数点の場合、転記の規則によりROUNDED指定がなくても結果の一意名には四捨五入された値が転記されます(“6.3.5 転記の規則”参照)。この場合、ROUNDED指定の有無は、実行の結果に影響を与えません。

6.3.9 ON SIZE ERROR指定

算術文を実行すると、桁あふれ条件が発生することがあります。桁あふれ条件は、算術文にON SIZE ERROR指定を書くことによって検出することができます。

桁あふれ条件が発生する条件

桁あふれ条件は、以下の場合に発生します。

1. べき乗の底の値がゼロで、かつその指数の値がゼロ以下の場合
2. べき乗の評価の結果が実数でない場合
3. 除算の除数がゼロの場合
4. 演算結果の絶対値が、結果の一意名に格納できる最大の値を超えた場合ただし結果の一意名がint型2進整数データ項目の場合、演算結果が“表5.1 int型2進整数データ項目の大きさと値の範囲”に示されている値の範囲に収まらない場合

4.で、結果の一意名に格納できる最大の値とは、PICTURE句の文字列で指定した最大の値です。結果の一意名が2進項目の場合も、記憶領域に格納できる最大の値ではなく、PICTURE句の文字列で指定した最大の値です。なお、演算結果が、結果の一意名より小数点以下の桁を多く持つ場合、演算結果と同じだけの小数点以下の桁を持つPICTURE句の文字列で指定した最大の値を、結果の一意名に格納できる最大の値とみなします。また、結果の一意名が“P”を含むPICTURE句で定義されている場合、“P”を“9”で置き換えたPICTURE句の文字列で指定した最大の値を、結果の一意名に格納できる最大の値とみなします。

4.の桁あふれ条件は、1つの算術演算の最終結果に格納するときにだけ発生し、中間結果に格納するときは発生しません。ROUNDED指定を書いた場合、四捨五入が行われた後、4.の桁あふれ条件が検査されます。

結果の一意名を2つ以上書いた場合、それぞれの算術演算の結果を求めるときに、それぞれ桁あふれ条件が検査されます。

桁あふれ条件が発生したときの動作

桁あふれ条件が発生すると、結果の一意名の値は、以下の値になります。

- ON SIZE ERROR指定またはNOT ON SIZE ERROR指定を書いた場合、桁あふれ条件が発生した一意名の値は、算術文の実行前のまま変更されません。
- ON SIZE ERROR指定もNOT ON SIZE ERROR指定も書かなかった場合、桁あふれ条件が発生した一意名の値は、規定されません。
- 桁あふれ条件が発生しなかった一意名には、ON SIZE ERROR指定とNOT ON SIZE ERROR指定の有無に関係なく、算術演算の結果が格納されます。

算術演算の完了後、すなわち結果の一意名のすべての値が決まった後、以下の規則に従って制御が移ります。

- ON SIZE ERROR指定を書いた場合、ON SIZE ERROR指定の無条件文に制御が移ります。

  無条件文の実行後、算術文の最後に制御が移ります。

  ただし、無条件文で制御の明示移行を起こす手続き分岐文または条件文を実行した場合、その文の規則に従って制御が移ります。

- ON SIZE ERROR指定を書かなかった場合、算術文の最後に制御が移ります。

除算の除数がゼロの場合、異常終了します。

桁あふれ条件が発生しなかったときの動作

桁あふれ条件が発生しなかったとき、算術演算の完了後、以下の規則に従って制御が移ります。

- NOT ON SIZE ERROR指定を書いた場合、NOT ON SIZE ERROR指定の無条件文に制御が移ります。

  無条件文の実行後、算術文の最後に制御が移ります。

  ただし、無条件文で制御の明示移行を起こす手続き分岐文または条件文を実行した場合、その文の規則に従って制御が移ります。

- NOT ON SIZE ERROR指定を書かなかった場合、算術文の最後に制御が移ります。

6.3.10 CORRESPONDING指定

MOVE文、ADD文およびSUBTRACT文には、CORRESPONDING指定を書くことができます。CORRESPONDING指定は、集団項目に従属するデータ項目で同じデータ名を持つものどうしを対応付けます。

CORRESPONDING指定を“CORRESPONDING d1 TO d2”のように書く場合、d1、d2および対応付けるデータ項目は、以下の条件を満足しなければなりません。

- d1およびd2は、集団項目でなければなりません。d1およびd2に、レベル番号が66、77または88のデータ項目を指定することはできません。
- d1およびd2に、USAGE IS INDEX句を指定したデータ項目を指定することはできません。
- d1およびd2は、部分参照することはできません。
- 対応付けるデータ項目の名前は、それらに暗黙の修飾語を付加することによって一意にならなければなりません。
- MOVE文の場合、対応付けるデータ項目の少なくとも一方は基本項目でなければなりません。また、対応付けるデータ項目の組合せは、転記の規則に従うものでなければなりません。
- ADD文およびSUBTRACT文の場合、対応付けるデータ項目は共に数字項目でなければなりません。

“CORRESPONDING d1 TO d2”を書いた場合、d1に従属するデータ項目とd2に従属するデータ項目のうち、以下のすべての条件を満足するデータ項目どうしが対応付けられます。

- データ名が同じである。ただし、FILLER項目を除く。
- d1およびd2のそれぞれ直前までの修飾語の名前の系列が同じである。
- REDEFINES句、RENAMES句、OCCURS句またはUSAGE IS INDEX句を指定していない。また、REDEFINES句、RENAMES句、OCCURS句またはUSAGE IS INDEX句を指定したデータ項目に従属していない。

6.3.11 作用対象の重なり

異なるデータ記述項で定義したデータ項目を、1つの文の送出し側項目と受取り側項目に指定した場合、それらが記憶領域の一部または全部を共有するときは、その文の実行結果は規定されません。また、同じデータ記述項で定義したデータ項目を、1つの文の送出し側項目と受取り側項目に指定した場合、その文の実行結果は規定されないことがあります。実行結果が規定されない場合は、各文の一般規則で説明します。

6.3.12 INVALID KEY指定

相対ファイルまたは索引ファイルに対して、DELETE文、乱呼出しのREAD文、REWRITE文、START文またはWRITE文を実行すると、無効キー条件が発生することがあります。無効キー条件は、これらの入出力文にINVALID KEY指定を書くことによって検出することができます。また、無効キー条件が発生せずに入出力文の実行が成功したことを、NOT INVALID KEY指定を書くことによって検査することができます。ここでは、無効キー条件が発生する可能性のある入出力文の動作を、以下の3つに分類して説明します。

- 無効キー条件が発生したときの動作
- 無効キー条件以外の例外条件が発生したときの動作
- 無効キー条件もその他の例外条件も発生しなかったときの動作

無効キー条件が発生したときの動作

無効キー条件が発生すると、入出力文の実行は不成功に終わります。入出力状態に無効キー条件を示す値が設定された後、入出力文のINVALID KEY指定の有無、および関連するUSE AFTER STANDARD EXCEPTION手続きの有無に従って、制御が移ります。無効キー条件が発生したときの制御の移行を、下表に示します。

INVALID KEY 指定の有無	USE AFTER STANDARD EXCEPTION 手続きの有無	無効キー条件が発生したときの制御の移行
あり	あり または なし	1. INVALID KEY 指定の無条件文に制御が移ります。 2. 無条件文の実行後、入出力文の最後に制御が移ります。(*1)
なし	なし	・ ファイルにFILE STATUS 句を指定した場合、入出力文の最後に制御が移ります。 ・ ファイルにFILE STATUS 句を指定しなかった場合、実行結果は規定されません。
なし	なし	・ ファイルにFILE STATUS 句を指定した場合、入出力文の最後に制御が移ります。 ・ ファイルにFILE STATUS 句を指定しなかった場合、実行結果は規定されません。

*1 ： 無条件文で制御の明示移行を起こす手続き分岐文または条件文を実行した場合、その文の規則に従って制御が移ります。

無効キー条件以外の例外条件が発生したときの動作

無効キー条件以外の例外条件が発生すると、入出力文の実行は不成功に終わります。入出力状態に例外条件を示す値が設定された後、以下に示す規則に従って制御が移ります。

- 関連するUSE AFTER STANDARD EXCEPTION手続きを書いた場合、USE AFTER STANDARD EXCEPTION手続きに制御が移ります。そして、USE文の規則に従って、制御が移ります。
- 関連するUSE AFTER STANDARD EXCEPTION手続きを書かなかった場合、ファイルにFILE STATUS句を指定したときは、入出力文の最後に制御が移ります。FILE STATUS句を指定しなかったときは、実行結果は規定されません。

無効キー条件もその他の例外条件も発生しなかったときの動作

無効キー条件もその他の例外条件も発生しなかったとき、入出力文の実行は成功します。入出力状態にその旨を示す値が設定された後、以下に示す規則に従って制御が移ります。

- 入出力文にNOT INVALID KEY指定を書いた場合、NOT INVALID KEY 指定の無条件文に制御が移ります。そして、無条件文の実行後、入出力文の最後に制御が移ります。ただし、無条件文で制御の明示移行を起こす手続き分岐文または条件文を実行した場合、その文の規則に従って制御が移ります。
- 入出力文にNOT INVALID KEY指定を書かなかった場合、入出力文の最後に制御が移ります。

6.3.13 AT END指定

順ファイル、相対ファイルまたは索引ファイルに対して順呼出しのREAD文を実行すると、ファイル終了条件が発生することがあります。ファイル終了条件は、READ文にAT END指定を書くことによって検出することができます。また、ファイル終了条件が発生せずに入出力文の実行が成功したことを、NOT AT END指定を書くことによって検査することができます。

ここでは、READ文の動作を、以下の3つに分類して説明します。

- ファイル終了条件が発生したときの動作
- ファイル終了条件以外の例外条件が発生したときの動作
- ファイル終了条件もその他の例外条件も発生しなかったときの動作

ファイル終了条件が発生したときの動作

ファイル終了条件が発生すると、READ文の実行は不成功に終わります。入出力状態にファイル終了条件を示す値が設定された後、READ文のAT END指定の有無、および関連するUSE AFTER STANDARD EXCEPTION手続きの有無に従って、制御が移ります。ファイル終了条件が発生したときの制御の移行を、下表に示します。

AT END指定の有無	USE AFTER STANDARD EXCEPTION 手続きの有無	ファイル終了条件が発生したときの制御の移行
あり	あり または なし	1. AT END指定の無条件文に制御が移ります。 2. 無条件文の実行後、READ文の最後に制御が移ります。(*1)
なし	あり	1. USE AFTER STANDARD EXCEPTION手続きに制御が移ります。 2. USE AFTER STANDARD EXCEPTION手続きの実行後、READ文の最後に制御が移ります。
なし	なし	・ ファイルにFILE STATUS 句を指定した場合、READ文の最後に制御が移ります。 ・ ファイルにFILE STATUS 句を指定しなかった場合、実行結果は規定されません。

*1 ： 無条件文で制御の明示移行を起こす手続き分岐文または条件文を実行した場合、その文の規則に従って制御が移ります。

ファイル終了条件以外の例外条件が発生したときの動作

ファイル終了条件以外の例外条件が発生すると、READ文の実行は不成功に終わります。入出力状態に例外条件を示す値が設定された後、以下に示す規則に従って制御が移ります。

- 関連するUSE AFTER STANDARD EXCEPTION手続きを書いた場合、USE AFTER STANDARD EXCEPTION手続きに制御が移ります。そして、USE文の規則に従って、制御が移ります。
- 関連するUSE AFTER STANDARD EXCEPTION手続きを書かなかった場合、ファイルにFILE STATUS句を指定したときは、READ文の最後に制御が移ります。FILE STATUS句を指定しなかったときは、実行結果は規定されません。

ファイル終了条件もその他の例外条件も発生しなかったときの動作

ファイル終了条件もその他の例外条件も発生しなかったとき、READ文の実行は成功します。入出力状態にその旨を示す値が設定された後、以下に示す規則に従って制御が移ります。

- READ文にNOT AT END指定を書いた場合、NOT AT END指定の無条件文に制御が移ります。そして、無条件文の実行後、READ文の最後に制御が移ります。ただし、無条件文で制御の明示移行を起こす手続き分岐文または条件文を実行した場合、その文の規則に従って制御が移ります。
- READ文にNOT AT END指定を書かなかった場合、READ文の最後に制御が移ります。

6.3.14 矛盾するデータ

字類条件を除いて、手続き部でデータ項目の内容が参照されるとき、データ項目の内容がPICTURE句によるデータ項目の字類または関数による字類と矛盾する場合、手続き部での参照の結果は規定しません。

6.4 文

この節では、各文について説明します。

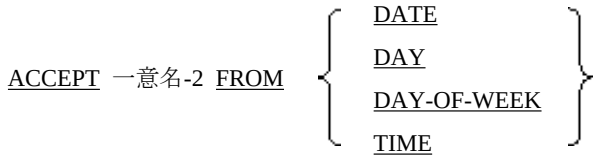
6.4.1 ACCEPT文（中核）

少量のデータを入力します。

【書き方1】データを入力する

ACCEPT 一意名-1 {[FROM 呼び名-1]}

【書き方2】年月日、曜日および時刻を得る



構文規則

1. 一意名-1は、以下のいずれかでなければなりません。
   - 英字項目、英数字項目、外部10進項目、内部10進項目、2進項目、または固定長の集団項目
   - 外部ブール項目
2. 呼び名-1は、環境部の特殊名段落で、機能名CONSOLEまたはSYSINに対応付けておかなければなりません。呼び名-1に機能名CONSOLEを指定する場合の一意名-1の大きさの最大値については、“付録B システムの定量制限”を参照してください。
3. 一意名-2は、英数字項目、英数字編集項目、数字編集項目、外部10進項目、内部10進項目、2進項目または固定長の集団項目でなければなりません。

一般規則

書き方1の規則

1. ACCEPT文は、以下のハードウェア装置からデータを読み込み、一意名-1に格納します。読み込まれたデータの編集および誤りの検査は、行われません。
   - CONSOLE(システム論理操作卓)
   - SYSIN(システム論理入力装置)
2. FROM指定を省略した場合、呼び名-1に、SYSINに対する呼び名を指定したものとみなされます。
3. 呼び名-1で指定した入出力装置がREAD文で入力操作を行う装置と同じ場合、結果は規定されません。
4. 呼び名-1にCONSOLEに対応付けた呼び名を書いた場合、以下の順に処理が行われます。
   a. システムによって作られたメッセージが自動的にシステム論理操作卓に表示され、ACCEPT文の実行が中断します。
   b. 利用者がシステム論理操作卓でメッセージを入力すると、ACCEPT文が再開されます。メッセージは、PICTURE句の記述に関係なく、左端から一意名-1に格納されます。入力したメッセージが一意名-1の長さより短い場合は、一意名-1の残りの部分には空白が詰められます。入力したメッセージが一意名-1の長さより長い場合は、メッセージの右側が一意名-1の長さに合わせて切り捨てられます。
5. 呼び名-1にSYSINに対応付けた呼び名を書いた場合、入力レコードを繰り返し読み込んで、順に受取り側データ項目に格納します。この処理を、受取り側データ項目が入力データで満たされるまで、または入力レコードがなくなるまで行います。

書き方2の規則

1. 書き方2のACCEPT文は、FROM指定に記述した情報を、MOVE文の規則に従って、一意名-2のデータ項目へ転記します。
   DATE(年月日)、DAY(年日)、DAY-OF-WEEK(曜日)およびTIME(時刻)は、ACCEPT文の中でだけ使える仮想のデータ項目です。COBOLプログラムの他の部分で使うことはできません。
2. DATEは、6桁の符号なし外部10進整数項目として扱われます。DATEを書いた場合、年の下2桁、月、日の順番に6桁の数字が一意名-2に転記されます。例えば、日付が1994年10月1日の場合、転記される値は941001です。
3. DAYは、5桁の符号なし外部10進整数項目として扱われます。DAYを書いた場合、年の下2桁、通日(1月1日からの日数)の順番に、5桁の数字が一意名-2に転記されます。例えば、日付が1994年10月1日の場合、転記される値は94274です。
4. TIMEは、8桁の符号なし外部10進整数項目として扱われます。TIMEを書いた場合、時(午前零時を起点とする24時間表示)、分、秒および100分の1秒の順番に、8桁の数字が一意名-2に転記されます。例えば、時間が午後2時41分の場合、転記される値は14410000です。
5. DAY-OF-WEEKは、1桁の符号なし外部10進整数項目として扱われます。DAY-OF-WEEKを書いた場合、曜日を表す1桁の数字が一意名-2に転記されます。曜日が月曜日、火曜日、…日曜日の場合、転記される値は、それぞれ1、2、…7です。

6.4.2 ACCEPT文(スクリーン操作)

画面からデータを入力します。

[WinIPF][Linux][LinuxIPF][Linux64][.NET]では、スクリーン操作機能は使用できません。

【書き方】

```
ACCEPT データ名-1
[ AT { LINE   NUMBER { 一意名-1 | 整数   } |
       COLUMN NUMBER { 一意名-2 | 整数-2 } } ]
  [ON EXCEPTION 無条件文-1]
  [NOT ON EXCEPTION 無条件文-2]
  [END-ACCEPT]
```

構文規則

1. データ名-1は、画面節で定義した画面項目でなければなりません。その画面項目は、以下のいずれかでなければなりません。
   - PICTURE句にTO指定またはUSING指定を書いた基本画面項目
   - PICTURE句にTO指定またはUSING指定を書いた基本画面項目を従属する集団画面項目
2. データ名-1は、修飾することができます。
3. 整数-1および整数-2は、符号なしの整数でなければなりません。
4. 一意名-1および一意名-2は、符号なしの整数項目でなければなりません。

5. LINE NUMBER指定とCOLUMN NUMBER指定は、任意の順に書くことができます。これらの指定の両方を書くこともできます。

一般規則

1. ACCEPT文は、データ名-1の画面項目に対応する画面上の領域からデータを読み込み、読み込んだデータを、データ名-1のPICTURE句のTO指定またはUSING指定に書いたデータ項目へ転記します。データ名-1に集団画面項目を指定した場合は、その画面項目に従属するすべての入力項目および更新項目が、この操作の対象になります。画面項目からデータ項目への転記は、転記の規則に従って行います。
2. データ名-1に集団画面項目を指定した場合、以下のように動作します。
   a. 1回のACCEPT文の実行で、集団画面項目に従属するすべての入力項目および更新項目への入力が可能になります。
   b. データの入力は、画面データ記述項のLINE NUMBER句およびCOLUMN NUMBER句で指定した、画面上の位置の順に、カーソルを移動して行います。
   c. データ名-1にAUTO句を指定した場合、カーソルは、自動的に次の入力領域または更新領域に移動します。
3. ACCEPT文は、入力キーの押下によって完了します。
4. 画面項目に対してAUTO句が有効な場合、ACCEPT文は以下の操作のときに完了します。
   a. データ名-1が集団画面項目の場合、最後の画面項目の領域に対してデータを入力をすると、ACCEPT文が完了します。
   b. データ名-1が基本画面項目の場合、その画面項目の領域に対して入力をすると、ACCEPT文が完了します。
5. LINE NUMBER指定では、データ名-1の画面項目に対応する画面の行番号を、物理画面の第1行を1として指定します。
6. COLUMN NUMBER指定では、データ名-1の画面項目に対応する画面の列番号を、物理画面の第1列を1として指定します。
7. AT指定を省略した場合、画面の第1行、第1列から入力操作が行われます。
8. ON EXCEPTION指定を書いた場合、入力操作が正常に終了しないときに無条件文-1が実行されます。
9. NOT ON EXCEPTION指定を書いた場合、入力操作が正常に終了すると無条件文-2が実行されます。正常終了の状態については、“4.2.3.6 CRT STATUS句”を参照してください。

6.4.3 ACCEPT文（コマンド行引数と環境変数の操作）

コマンド行の引数の個数、値または環境変数の値を入力します。

【書き方】

ACCEPT 一意名-1 FROM 呼び名-1
[ON EXCEPTION 無条件文-1]
[NOT ON EXCEPTION 無条件文-2]
[END-ACCEPT]

構文規則

1. 呼び名-1は、環境部の特殊名段落で、以下のいずれかの機能名に対応付けておかなければなりません。
   － ARGUMENT-NUMBER

- ARGUMENT-VALUE
- ENVIRONMENT-VALUE

2. 一意名-1は、以下の条件を満足するデータ項目でなければなりません。
   a. 呼び名-1に、機能名ARGUMENT-NUMBERに対応付けた呼び名を指定する場合、一意名-1は符号なし整数項目でなければなりません。
   b. 呼び名-1に、機能名ARGUMENT-VALUEまたはENVIRONMENT-VALUEに対応付けた呼び名を指定する場合、一意名-1は可変反復データ項目を従属しない集団項目または英数字項目でなければなりません。
3. 呼び名-1に、機能名ARGUMENT-NUMBERに対応付けた呼び名を指定する場合、ON EXCEPTION指定およびNOT ON EXCEPTION指定を書くことはできません。

一般規則

呼び名-1を機能名ARGUMENT-NUMBERに対応付けた場合の規則

コマンド行に指定した引数の個数が、転記の規則に従って一意名-1に転記されます。

呼び名-1を機能名ARGUMENT-VALUEに対応付けた場合の規則

1. 現在の引数位置指示子が示す引数の値が、転記の規則に従って一意名-1に転記されます。
2. ACCEPT文を実行すると、引数位置指示子の値が1だけ増加します。
3. ON EXCEPTION指定を書いた場合、ACCEPT文を実行する前の引数指示子の値がコマンド行の最後の引数を超えているとき、無条件文-1が実行されます。
4. NOT ON EXCEPTION指定を書いた場合、3.に示した例外条件が発生しなかったとき、無条件文-2が実行されます。

呼び名-1を機能名ENVIRONMENT-VALUEに対応付けた場合の規則

1. ACCEPT文を実行する前に、機能名ENVIRONMENT-NAMEに対応付けた呼び名を指定したDISPLAY文を実行しなければなりません。ACCEPT文を実行すると、先行するDISPLAY文によって位置付けられた環境変数の値が、転記の規則に従って一意名-1に転記されます。
2. ON EXCEPTION指定を書いた場合、以下のときに無条件文-1が実行されます。
   - ACCEPT文を実行する前に、環境変数を位置付けるためのDISPLAY文を実行しなかった場合
   - 先行するDISPLAY文で指定した環境変数が、存在しない場合
3. NOT ON EXCEPTION指定を書いた場合、2.に示した例外条件が発生しなかったとき、無条件文-2が実行されます。

6.4.4 ADD文（中核）

加算の結果を求めます。

【書き方1】加算の結果を被加数と置き換える

```
ADD { 一意名-1 } … TO {一意名-2 [ROUNDED]} …
    { 整数-1   }
    [ON SIZE ERROR 無条件文-1]
    [NOT ON SIZE ERROR 無条件文-2]
    [END-ADD]
```

【書き方2】加算の結果を被加数とは異なるデータ項目に格納する

```
ADD  { 一意名-1 }  … TO  { 一意名-2 }  GIVING
     { 定数-1   }        { 定数-2   }
     {一意名-3 [ROUNDED]} …
     [ON SIZE ERROR 無条件文-1]
     [NOT ON SIZE ERROR 無条件文-2]
     [END-ADD]
```

【書き方3】2つの集団項目に従属するデータ項目の対応をとって加算する

```
ADD  { CORRESPONDING }  一意名-1 TO 一意名-2
     { CORR          }
     [ROUNDED]
     [ON SIZE ERROR 無条件文-1]
     [NOT ON SIZE ERROR 無条件文-2]
     [END-ADD]
```

構文規則

1. 書き方1と書き方2の場合、一意名-1および一意名-2は、数字項目でなければなりません。書き方3の場合、一意名-1および一意名-2は、集団項目でなければなりません。
2. 一意名-3は、数字項目または数字編集項目でなければなりません。
3. 定数-1および定数-2は、数字定数でなければなりません。
4. CORRESPONDINGとCORRは同義語です。どちらも書くことができます。

一般規則

書き方1の規則

書き方1のADD文は、TOの前の各作用対象の和を一意名-2に加え、一意名-2に格納します。一意名-2の並びを書いた順に、この加算を行います。

書き方2の規則

書き方2のADD文は、TOの前の各作用対象の和を、TOの後の作用対象に加え、一意名-3に格納します。一意名-3の並びを書いた順に、この加算の結果を格納します。

書き方3の規則

書き方3のADD文は、一意名-1に従属するデータ項目と一意名-2に従属するデータ項目のうち、名前の修飾語の系列が同じものどうしの和を求めます。一意名-1に従属するデータ項目を加数、一意名-2に従属するデータ項目を被加数として和を求め、一意名-2に従属するデータ項目に格納します。この結果は、対応する一意名ごとに、別々のADD文を書いた結果と同じです。

書き方1～書き方3に共通する規則

1. END-ADD指定は、ADD文の範囲を区切ります。
2. ROUNDED指定、ON SIZE ERROR指定、CORRESPONDING指定、演算および転記の規則については、“6.3 文に関する共通の規則”を参照してください。

6.4.5 ALTER文（中核）

前もって決められた処理の順序を変更します。ALTER文は廃要素です。

【書き方】

ALTER {手続き名-1 TO [PROCEED TO] 手続き名-2} …

注意

[.NET]では、ALTER文は使用できません。

構文規則

1. 手続き名-1は、段落名でなければなりません。その段落は、DEPENDING指定なしのGO TO文1個からなる1つの完結文だけで構成しなければなりません。
2. 手続き名-2は、手続き部の段落名または節名でなければなりません。

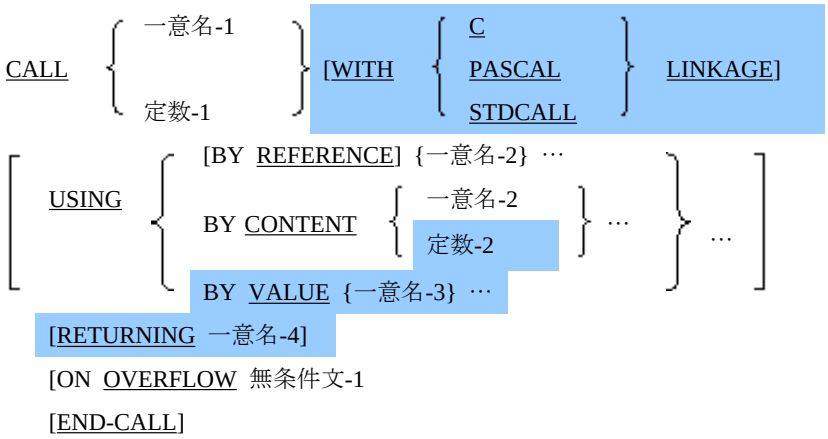
一般規則

ALTER文は、手続き名-1の段落に書いたGO TO文の行き先を、手続き名-2に変更します。

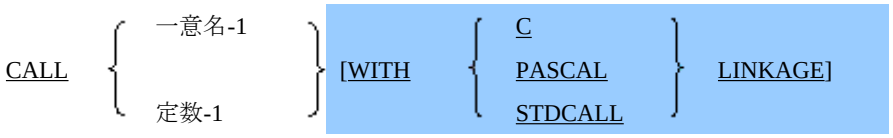
6.4.6 CALL文（プログラム間連絡）

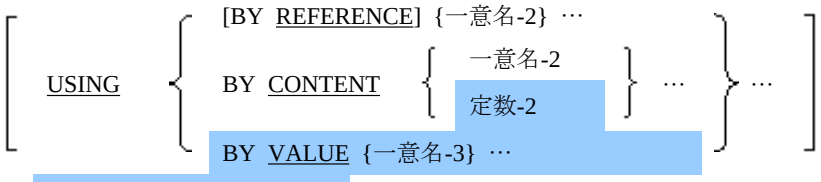
実行単位中の他のプログラムに制御を移します。

【書き方1】ON OVERFLOW指定



【書き方2】ON EXCEPTION指定





[RETURNING 一意名-4]

[ON EXCEPTION 無条件文-1]

[NOT ON EXCEPTION 無条件文-2]

[END-CALL]

注意

WITH指定は、[Win16][Win32]固有機能です。

RETURNING指定は、[Win32][WinIPF][Winx64][Solaris][Linux][LinuxIPF][Linux64][.NET]固有機能です。

構文規則

1. 一意名-1は、英数字項目でなければなりません。

   [Win32][WinIPF][Winx64][Solaris][Linux][LinuxIPF][Linux64][.NET]では、一意名-1に日本語項目も指定できます。

2. 以下は、ファイル節、作業場所節または連絡節で定義したデータ項目でなければなりません。

   - 一意名-2
   - 一意名-3
   - 一意名-4

3. 以下は、レベル番号01のデータ項目、レベル番号77のデータ項目または任意のレベルの基本項目でなければなりません。

   - 一意名-2
   - 一意名-4

   ただし、このコンパイラでは、任意のレベルの集団項目を書くことができます。

4. 一意名-2および一意名-4に内部ブール項目を指定する場合、内部ブール項目はバイト境界で始まるように定義する必要があります。

5. 一意名-3は、用途がCOMPUTATIONAL-5の9桁以下の数字項目、1文字の英字項目、1文字の英数字項目、またはSEPARATE指定のない1桁の外部10進整数項目でなければなりません。[Win32][WinIPF][Winx64][Solaris][Linux][LinuxIPF][Linux64][.NET]では、一意名-3に用途がBINARY-CHAR、BINARY-SHORT、BINARY-LONGの数字項目も指定できます。[WinIPF][Winx64][LinuxIPF][Linux64]では、一意名-3に用途がCOMPUTATIONAL-5の18桁以下の数字項目、用途がBINARY-DOUBLEの数字項目も指定できます。

6. [WinIPF][Winx64][LinuxIPF][Linux64]では、一意名-4に以下の項目を指定することはできません。

   - 用途がCOMPUTATIONAL-1の基本項目のみを従属する集団項目
   - 用途がCOMPUTATIONAL-2の基本項目のみを従属する集団項目

7. 定数-1は、文字定数でなければなりません。

   [Win32][WinIPF][Winx64][Solaris][Linux][LinuxIPF][Linux64][.NET]では、定数-1に日本語文字定数も指定できます。

8. 定数-2は、文字定数、16進文字定数または日本語文字定数でなければなりません。
9. USING指定の作用対象の個数の最大値については、“付録B システムの定量制限”を参照してください。

一般規則

1. CALL文は、実行単位中の他のプログラムを呼び出します。CALL文を書いたプログラムを、「呼ぶプログラム」といいます。CALL文の実行によって呼び出されるプログラムを、「呼ばれるプログラム」といいます。呼ばれるプログラムの名前は、一意名-1または定数-1で指定します。呼ばれるプログラムにパラメタを渡す場合、USING指定を書きます。呼ばれるプログラムが実行可能かどうかを検出する場合、ON OVERFLOW指定、ON EXCEPTION指定またはNOT ON EXCEPTION指定を書きます。
2. 呼ばれるプログラムが呼ぶプログラムと同じ翻訳単位に含まれるプログラムの場合、定数-1または一意名-1のデータ項目の内容は、呼ばれるプログラムの内部名を含まなければなりません。

   また、別の翻訳単位のプログラムの場合、定数-1または一意名-1のデータ項目の内容は、呼ばれるプログラムの外部名を含まなければなりません。
3. CALL文を実行したとき、CALL文で指定したプログラムが実行可能であるならば、制御は呼ばれるプログラムに移ります。
4. CALL文の実行による制御の移行は、ON OVERFLOW指定、ON EXCEPTION指定およびNOT ON EXCEPTION指定の有無によって異なります。CALL文の実行による制御の移行を、下表に示します。

ON EXCEPTION指定 または ON OVERFLOW指定の有無	NOT ON EXCEPTION指定の有無	CALL文の動作	
		呼ばれるプログラムが実行不可能の場合	呼ばれるプログラムが実行可能の場合
あり	あり	[1]無条件文-1に制御が移ります。 [2]無条件文-1の実行後、CALL文の終わりに制御が移 ります。(*1)	[1]呼ばれるプログラムに制御が移ります。 [2]呼ばれるプログラムから戻った後無条件文-2に制御が移ります。 [3]無条件文-2の実行後、CALL文の終わりに制御が移ります。(*2)
あり	なし	[1]無条件文-1に制御が移ります。 [2]無条件文-1の実行後、CALL文の終わりに制御が移ります。(*1)	[1]呼ばれるプログラムに制御が移ります。 [2]呼ばれるプログラムから戻った後CALL文の終わりに制御が移ります。
なし	あり	CALL文の動作は、規定されません。	[1]呼ばれるプログラムに制御が移ります。 [2]呼ばれるプログラムから戻った後無条件文-2に制御が移ります。 [3]無条件文-2の実行後、CALL文の終わりに制御が移ります。(*2)
なし	なし	CALL文の動作は、規定されません。	[1]呼ばれるプログラムに制御が移ります。

ON EXCEPTION指定 または ON OVERFLOW指定の有無	NOT ON EXCEPTION指定の有無	CALL文の動作	
		呼ばれるプログラムが実行不可能の場合	呼ばれるプログラムが実行可能の場合
			[2]呼ばれるプログラムから戻った後CALL文の終わりに制御が移ります。

*1 ： 無条件文-1に制御の明示移行を起こす手続き分岐または条件文を書いた場合、その文の規則に従って制御が移ります。
*2 ： 無条件文-2に制御の明示移行を起こす手続き分岐または条件文を書いた場合、その文の規則に従って制御が移ります。

5. 実行単位の2つ以上のプログラムには、同じプログラム名を付けることができます。CALL文に指定したプログラム名が、実行単位中に2つ以上存在する場合、呼ばれるプログラムは、名前の範囲の規則に従って、以下の手順で決定されます。

   a. CALL文を書いたプログラムに直接含まれるプログラムのうち、CALL文に指定したプログラム名が存在するかが検査されます。そのようなプログラム名が見つかった場合、そのプログラムが呼び出されます。

   b. 次に、CALL文を書いたプログラムを直接または間接に含むプログラムに直接含まれ、かつ共通属性を持つプログラムのうち、CALL文に指定したプログラム名が存在するかが検査されます。そのようなプログラム名が見つかった場合、そのプログラムが呼び出されます。

   c. 次に、別の翻訳単位中に、CALL文に指定したプログラムが存在するかが検査されます。そのようなプログラムが見つかった場合、そのプログラムが呼び出されます。

6. 呼ばれるプログラムが初期化属性を持たない場合、そのプログラムおよびその中に直接または間接に含まれるプログラムは、以下の場合に初期状態になります。

   ― 実行単位で最初に呼び出されたとき。

   ― その呼ばれるプログラムに対するCANCEL文の実行後、最初に呼び出されたとき。

   上記以外の場合、実行単位で2回目以降に呼ばれるプログラムおよびそれに直接または間接に含まれるプログラムは、そのプログラムから最後に戻ったときと同じ状態になっています。

7. 呼ばれるプログラムが初期化属性を持つ場合、そのプログラムおよびその中に直接または間接に含まれる各プログラムは、呼び出されるときはいつでも初期状態になります。

8. 呼ばれるプログラムが初期状態のとき、そのプログラムの内部ファイル結合子に関連するファイルは、開かれた状態ではありません。呼ばれるプログラムが初期状態でないとき、そのプログラムの内部ファイル結合子に関連するファイルの状態およびファイルの位置付けは、そのプログラムの実行が最後に終了したときと同じ状態になっています。プログラムの初期状態については、“2.3.7 プログラムの初期状態”を参照してください。

9. 呼ぶプログラムの呼ぶ処理または呼ばれるプログラムから戻る処理は、外部ファイル結合子に関連するファイルの状態または位置付けを変更しません。

10. COBOLプログラムを呼び出す場合、呼ばれるプログラムの手続き部の

    ― 見出し

    ― またはENTRY文

    にUSING指定を書いたときだけ、CALL文にUSING指定を書くことができます。その場合、USING指定の作用対象の数は、呼ぶプログラムと呼ばれるプログラムで同じでなければなりません。

11. CALL文のUSING指定の作用対象には、呼ばれるプログラムに渡すパラメタを指定します。呼ばれるプログラムがCOBOLプログラムの場合、プログラムの手続き部の

    ― 見出し

－ またはENTRY文

のUSING指定でパラメタを受け取ります。各パラメタは、USING指定に書いた順によって対応付けられます。例えば、COBOLプログラムに2つのパラメタを渡す場合、CALL文のUSING指定の1番目および2番目のパラメタが、呼ばれるプログラムの手続き部の見出しのそれぞれ1番目および2番目のデータ名と対応付けられます。

12. CALL文のUSING指定に書いたパラメタの値は、CALL文が実行されたときに、呼ばれるプログラムで使用可能になります。

13. 以下には、2つ以上のパラメタを書くことができ、これらの指定を組み合わせることもできます。

    － BY CONTENT指定

    － BY REFERENCE指定

    － BY VALUE指定

    これらの指定を組み合わせた場合、1つの指定は、別のBY CONTENT指定、BY REFERENCE指定またはBY VALUE指定が現れるまでのパラメタに対して有効です。最初のパラメタの前にBY CONTENT指定、BY REFERENCE指定およびBY VALUE指定のうちのどれも書かなかった場合、BY REFERENCE指定を書いたものとみなされます。

14. BY REFERENCE指定を明にまたは暗に指定した場合、呼ばれるプログラムのパラメタと呼ぶプログラムのパラメタは同じ記憶領域を占有するものとみなされます。パラメタの文字位置の個数は、呼ぶプログラムと呼ばれるプログラムで同じでなければなりません。

15. BY CONTENT指定を書いた場合、呼ぶプログラムで指定したパラメタの値は、呼ばれるプログラムの手続き部の

    － 見出し

    － またはENTRY文

    に書いたデータ項目で参照することができます。しかし、呼ばれるプログラムでこのデータ項目を変更しても、その値は、呼ぶプログラムのパラメタに反映されません。パラメタの用途および文字位置の個数は、呼ぶプログラムと呼ばれるプログラムで同じでなければなりません。

16. BY CONTENT指定に定数-2を指定した場合、呼ばれるプログラムの対応するデータ項目と定数-2の属性および文字位置の個数は、互いに同じでなければなりません。

17. BY VALUE指定を書いた場合、パラメタの値が、直接呼ばれるプログラムに渡されます。呼ばれるプログラムは、値によるパラメタを受け取ることができる言語で書いたプログラムでなければなりません。呼ばれるプログラムで、BY VALUE指定によって渡された値を変更しても、呼ぶプログラムのデータ項目の値は変更されません。

18. 呼ばれるプログラムでは、呼ぶプログラムを直接または間接に呼ぶCALL文を実行することはできません。

19. 宣言部の中でCALL文を実行する場合、そのCALL文は、制御が渡されていて実行が完了していない呼ばれるプログラムを直接または間接に呼ぶものであってはいけません。

20. END-CALL指定は、CALL文の範囲を区切ります。

21. 呼ばれるプログラムの手続き部の先頭に制御を移すためには、定数-1または一意名-1に、呼ばれるプログラムのプログラム名を指定します。

    手続き部の先頭以外の入口点に制御を移すためには、定数-1または一意名-1に、ENTRY文に指定した二次入口名を指定します。

22. 大域名を参照できるプログラムを呼び出す場合、CALL文のUSING指定にその大域名を書くことはできません。

23. CALL文の実行の始めに、以下が評価されます。

    － 一意名-1

    － 一意名-2

    － 一意名-4

24. RETURNING指定が書かれた場合、一意名-1または定数-1で識別されるプログラムの結果が一意名-4に格納されます。

25. RETURNING指定が書かれた場合、呼び出されるプログラムの手続き部の見出しにRETURNING指定が書かれていなければなりません。
26. RETURNING指定が書かれなかった場合、呼び出されるプログラムの手続き部の見出しにRETURNING指定が書かれていてはなりません。
27. 呼び出されるプログラムでRETURNING指定の項目に値を設定しない場合、CALL文のRETURNING指定に書かれた項目の値は不定となります。
28. RETURNING指定のパラメタの用途および文字位置の個数は、呼ぶプログラムと呼ばれるプログラムで同じでなければなりません。

WITH指定の規則

1. WITH指定は、各呼出し規約に従って作成されたプログラムを呼び出すことを指定します。WITH指定が省略された場合、COBOLの定めたリンケージの規約を使用します。

   呼出し規約を、以下に示します。

   - COBOLの呼出し規約
   - Cの呼出し規約(WITH C LINKAGE)
   - PASCALの呼出し規約(WITH PASCAL LINKAGE)
   - STDCALLの呼出し規約(WITH STDCALL LINKAGE)
2. 呼ぶプログラムと呼ばれるプログラムの呼出し規約が異なった場合の実行結果は規定しません。
3. WITH指定は、含まれるプログラムの呼出しまたは含まれるプログラムである可能性のあるプログラムの呼出しに指定してはなりません。

6.4.7 CANCEL文(プログラム間連絡)

次回に呼び出されたときのプログラムの状態を初期状態にします。

【書き方】

CANCEL { 一意名-1 / 定数-1 } …

構文規則

1. 一意名-1は、英数字項目でなければなりません。

   [Win32][WinIPF][Winx64][Solaris][Linux][LinuxIPF][Linux64][.NET]では、一意名-1に日本語項目も指定できます。
2. 定数-1は、文字定数でなければなりません。

   [Win32][WinIPF][Winx64][Solaris][Linux][LinuxIPF][Linux64][.NET]では、定数-1に日本語文字定数も指定できます。

一般規則

1. CANCEL文は、実行単位とプログラムとの論理的な関係を取り消します。取り消すプログラムの名前は、一意名-1または定数-1に指定します。CANCEL文を実行すると、一意名-1または定数-1に指定したプログラムは、CANCEL文を書いたプログラムの実行単位と論理的な関係を持たなくなります。

2. 取り消すプログラムがCANCEL文を書いたプログラムと同じ翻訳単位に含まれるプログラムの場合、定数-1または一意名-1のデータ項目の内容は、取り消すプログラムの内部名を含まなければなりません。

   また、別の翻訳単位のプログラムの場合、定数-1または一意名-1のデータ項目の内容は、取り消すプログラムの外部名を含まなければなりません。

3. CANCEL文を実行すると、CANCEL文に指定したプログラムおよびそれに含まれるすべてのプログラムが取り消されます。これは、翻訳単位に書いた入れ子のプログラムで、含まれるプログラムから含むプログラムの順にCANCEL文を実行した結果と同じです。
4. CANCEL文の実行が成功した後、CANCEL文に指定したプログラムをその実行単位から再度呼び出したとき、そのプログラムは初期状態になります。これらの規則は、暗黙的なCANCEL文に対しても適用されます。
5. CANCEL文に指定したプログラムは、すでに呼び出されていて、まだEXIT PROGRAM文が実行されていないプログラムを直接または間接に参照してはいけません。例えば、プログラムAの実行中に呼び出されたプログラムの中で、CANCEL“A”を実行することはできません。
6. CANCEL文の実行によって取り消されたプログラムは、再度CALL文を実行したときにだけ論理的な関係が付けられます。
7. CALL文によって呼び出されたプログラムは、以下のいずれかの場合に取り消されます。
   a. そのプログラムに対するCANCEL文を実行した場合
   b. そのプログラムを含む実行単位が終了した場合
   c. そのプログラムが初期化属性を持つ場合で、そのプログラムでEXIT PROGRAM文を実行した場合

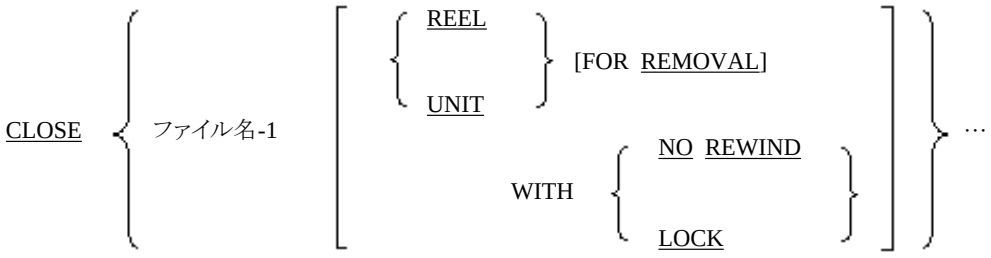
   上記のb.およびc.を、「暗黙的なCANCEL文」といいます。

6.4.8 CLOSE文（順ファイル・相対ファイル・索引ファイル・表示ファイル・報告書作成）

ファイルの処理を終了させます。

表示ファイルは、富士通拡張仕様です。

【書き方1】順ファイル・報告書ファイル



【書き方2】相対ファイル・索引ファイル・表示ファイル

CLOSE {ファイル名-1 [WITH LOCK]} …

注意

[WinIPF][Winx64][LinuxIPF][Linux64][.NET]では報告書作成機能は使用できません。

構文規則

書き方1および書き方2に共通する規則

ファイル名-1の並びに、ORGANIZATION句の指定の異なるファイルまたはACCESS MODE句の指定の異なるファイルを指定することもできます。

書き方1の規則

1. NO REWIND指定を書く場合、ファイル名-1はレコード順ファイルでなければなりません。
2. REEL指定またはUNIT指定を書く場合、ファイル名-1はレコード順ファイルまたはFORMAT句なしの印刷ファイルでなければなりません。

一般規則

CLOSE文の動作は、記憶媒体によって異なります。ファイルを以下の3つに分類して説明します。

1. 「非リール/ユニット」(順ファイル、相対ファイル、索引ファイル、表示ファイルおよび報告書ファイル)

   巻き戻しやリール/ユニットの概念が意味を持たない入出力媒体のファイル

2. 「順単一リール/ユニット」(順ファイルおよび報告書ファイル)

   1つのリール/ユニットに完全に含まれる順ファイル

3. 「順複数リール/ユニット」(順ファイルおよび報告書ファイル)

   2つ以上のリール/ユニットにまたがる順ファイル

各ファイルに共通する規則

1. CLOSE文を実行すると、ファイル名-1の入出力状態の値が更新されます。
2. ファイル名-1を2つ以上書いた場合のCLOSE文の実行結果は、ファイル名-1を書いた順に、各ファイルに対して別々のCLOSE文を実行した結果と同じです。

相対ファイル、索引ファイル、表示ファイル、非リール/ユニットの順ファイルおよび報告書ファイルに共通する規則

1. ファイル名-1のファイルは、CLOSE文を実行する前に開いておかなければなりません。
2. RELL/UNIT指定のないCLOSE文の実行により、ファイル名-1のファイルに対して、以下の処理が行われます。
   a. ファイル閉じ

      標準の終了手続きを実行します。

   b. ファイルのロック

      LOCK指定を書いた場合、ファイルをロックし、その実行単位で再び開けないようにします。

3. RELL/UNIT指定のないCLOSE文の実行が成功すると、ファイル名-1のファイルは以下の状態になります。
   a. ファイル名-1のレコード領域は参照できなくなります。
   b. ファイル名-1のファイル結合子が保持しているすべてのレコードのロックおよびファイル名-1のファイルのロックが、他の実行単位のために解除されます。
   c. ファイル名-1のファイルは、開いた状態から取り外され、ファイル名-1のファイル結合子と関連しなくなります。
4. CLOSE文の実行が不成功になった後にレコード領域を参照したときの実行結果は、規定されません。
5. 順ファイル、相対ファイルおよび索引ファイルの場合、入力モードで開かれた不定ファイルが存在しないとき、そのファイルに対してファイル終了処理は行われず、ファイル位置指示子は変わりません。
6. RELL/UNIT指定のCLOSE文をしても、ファイルは開かれたままであり、各ファイルに共通する規則1.の動作を除いては、何も行われません。

リール/ユニットの順ファイルおよび報告書ファイルの規則

1. リール/ユニットファイルに対するCLOSE文の実行結果を、下表に示します。

CLOSE文の書き方	順単一リール/ユニット	順複数リール/ユニット
CLOSE	(c), (g)	(a), (c), (g)
CLOSE WITH LOCK	(c), (e), (g)	(a), (c), (e), (g)
CLOSE WITH NO REWIND	(b), (c)	(a), (b), (c)
CLOSE REEL/UNIT	(f), (g)	(f), (g)
CLOSE REEL/UNIT FOR REMOVAL	(d), (f), (g)	(d), (f), (g)

表中の括弧付き英字は、以下の項番に対応します。実行結果がオープンモードによって異なるときは、オープンモードごとに分けて説明します。分けていない場合は、任意のオープンモードで開いたファイルに適用します。

a. 前のリール/ユニットへの影響

- 入力または入出力両用モードで開いたファイルの場合

  現在のリール/ユニットより前のすべてのリール/ユニットは、前にREEL/UNIT指定付きのCLOSE文で処理されている場合を除いて、閉じられます。現在のリール/ユニットがそのファイルの最後のものでない場合、現在のものに続くリール/ユニットは処理されません。

- 出力モードで開いたファイルの場合

  現在のリール/ユニットより前のすべてのリール/ユニットは、前にREEL/UNIT指定付きのCLOSE文で処理されている場合を除いて、閉じられます。

b. 現在のリール/ユニットの巻戻しはしません。

  現在のリール/ユニットは、巻き戻されず、そのままの位置に置かれます。

c. ファイル閉じ

- 入力または入出力両用モードで開いたファイルの場合

  そのファイルが終了の位置にあり、かつラベルレコードが指定されている場合、ラベルは標準のラベル手続きに従って処理されます。その後、標準の終了手続きが実行されます。ファイルが終了の位置にあり、かつラベルレコードが指定されていない場合、またはファイルが終了の位置にない場合、標準の終了手続きだけが実行され、終わりラベル処理は行われません。

- 出力モードで開いたファイルの場合

  ラベルレコードが指定されている場合、ラベルは標準のラベル手続きに従って処理されます。その後、標準の終了手続きが実行されます。ラベルレコードの指定がない場合、ラベル処理は行われずに、標準の終了手続きが実行されます。

- 拡張モードで開いたファイルの場合

  標準の終了手続きが実行されます。

d. リール/ユニットの取外し

  現在のリール/ユニットを可能ならば巻き戻し、リール/ユニットをこの実行単位から論理的に取り外します。リール/ユニットを、ファイル中の適切な順番で再び処理するためには、REEL/UNIT指定なしのCLOSE文を実行した後、OPEN文を実行しなければなりません。

  このコンパイラでは、FOR REMOVAL指定は注釈とみなされます。[.NET]では、FOR REMOVAL指定は指定できません。

e. ファイルのロック

  ファイルをロックし、その実行単位で再び開けないようにします。

f. リール/ユニット閉じ

- 入力または入出力両用モードで開いたファイルの場合

  現在のリール/ユニットが最後のリール/ユニットであるかまたはそのファイルの唯一のリール/ユニットである場合、またはそのリールが非リール/ユニット記憶媒体にある場合、リール/ユニットの交換は行われず、ボリューム指示子は変わりません。

そのファイルにもう1つのリール/ユニットがあれば、リール/ユニットの交換が行われ、ボリューム指示子はそのファイルの次のリール/ユニットを指すように更新され、標準の始めリール/ユニットラベル手続きが実行されます。現在ボリュームにデータレコードがなければ、さらにリール/ユニット交換が行われます。

- 出力モードで開いたファイル(リール/ユニット記憶媒体)の場合

  標準の終わりリール/ユニットラベル手続きが実行されます。

  リール/ユニットの交換が行われます。ボリューム指示子が新しいリール/ユニットを指すように更新されます。

  標準の始めリール/ユニットラベル手続きが実行されます。

  次に実行するWRITE文で、次のリール/ユニット上の次の位置にレコードを書き出すための準備が行われます。

g. 巻戻し

現在のリール/ユニットは、その物理的に始めの位置に位置付けられます。

2. CLOSE文を実行すると、ファイル名-1のファイル結合子が保持しているすべてのレコードのロック、およびファイル名-1のファイルのロックが、他の実行単位のために解除されます。
3. 入力モードで開かれた不定ファイルが存在しない場合、そのファイルに対してファイル終了手続きもリール/ユニット手続きも実行されず、ファイル位置指示子もボリューム指示子も変わりません。
4. REEL指定およびUNIT指定を省略した場合、CLOSE文の実行が成功すると、ファイル名-1のファイルは以下の状態になります。
   a. ファイル名-1のレコード領域は参照できなくなります。CLOSE文の実行が不成功になった後にレコード領域を参照したときの実行結果は、規定されません。
   b. ファイル名-1のファイルは、開いた状態から取り外され、ファイル名-1のファイル結合子と関連しなくなります。

報告書ファイルの規則

報告書ファイルに関連する開始されているすべての報告書は、CLOSE文を実行する前に、TERMINATE文を実行して終了しておかなければなりません。

6.4.9 COMPUTE文（中核）

算術式またはブール式を演算します。

使い方の例については、“付録H サンプル集”の“H.36 COMPUTE文”を参照してください。

【書き方1】算術式を演算する

COMPUTE {一意名-1 [ROUNDED]} … = 算術式-1
[ON SIZE ERROR 無条件文-1]
[NOT ON SIZE ERROR 無条件文-2]
[END-COMPUTE]

【書き方2】ブール式を演算する

COMPUTE {一意名-2} … = ブール式-1

注意

“＝”は必要語です。他の記号との混同を避けるために下線を付けていません。

構文規則

1. 一意名-1は、数字項目または数字編集項目でなければなりません。
2. 一意名-2は、ブール項目でなければなりません。

一般規則

書き方1の規則

1. 書き方1のCOMPUTE文は、算術式-1を演算し、その結果を一意名-1に格納します。一意名-1の並びを書いた順に、この演算の結果を格納します。
2. 算術式-1に1つの一意名だけまたは1つの定数だけを書いた場合、一意名または定数の値が一意名-1に格納されます。

書き方2の規則

1. 書き方2のCOMPUTE文は、ブール式-1を演算し、その結果を一意名-2に格納します。一意名-2の並びを書いた順に、この演算の結果を格納します。
2. ブール式-1に1つの一意名だけまたは1つの定数だけを書いた場合、一意名または定数の値が一意名-2に格納されます。

書き方1および書き方2に共通する規則

1. END-COMPUTE指定は、COMPUTE文の範囲を区切ります。
2. ROUNDED指定、ON SIZE ERROR指定、算術式、ブール式、演算および転記の規則については、“6.3 文に関する共通の規則”を参照してください。

6.4.10 CONTINUE文（中核）

実行可能な文がないことを示します。

【書き方】

CONTINUE

構文規則

CONTINUE文は、条件文または無条件文が書けるところならば、どこにでも書くことができます。

一般規則

CONTINUE文は、プログラムの実行に何の影響も与えません。

6.4.11 DELETE文（相対ファイル・索引ファイル）

大記憶ファイルからレコードを論理的に取り除きます。

【書き方】

DELETE ファイル名-1 RECORD

[INVALID KEY 無条件文-1]

[NOT INVALID KEY 無条件文-2]

[END-DELETE]

構文規則

1. ファイル名-1に順呼出し法のファイルを指定する場合、INVALID KEY指定またはNOT INVALID KEY指定を書くことはできません。
2. ファイル名-1に乱呼出し法または動的呼出し法のファイルを指定し、ファイル名-1に関連するUSE AFTER STANDARD EXCEPTION手続きを書かない場合、INVALID KEY指定を書かなければなりません。

   ただし、このコンパイラでは、USE AFTER STANDARD EXCEPTION手続きとINVALID KEY指定の両方を省略することができます。
3. 索引ファイルの場合、ファイル名-1は、DUPLICATES指定付きのRECORD KEY句を指定した乱呼出し法のファイルであってはいけません。

一般規則

相対ファイルおよび索引ファイルに共通する規則

1. ファイル名-1のファイルは、大記憶ファイルでなければなりません。
2. ファイル名-1のファイルは、DELETE文を実行する前に、入出力両用モードで開いておかなければなりません。
3. ファイル名-1に順呼出し法のファイルを指定した場合、DELETE文を実行する前に、ファイル名-1のファイルに対するREAD文を実行しなければなりません。READ文の実行が成功した場合、DELETE文を実行すると、READ文によって読み込まれたレコードがファイルから取り除かれます。
4. 他のファイル結合子によってロックされているレコードを削除しようとすると、DELETE文の実行は不成功になります。そして、レコードのロックを示す値が、ファイル名-1の入出力状態に設定されます。
5. DELETE文の実行が成功すると、削除の対象となるレコードがファイルから論理的に取り除かれます。それ以後、そのレコードを参照することはできません。
6. DELETE文を実行しても、ファイル位置指示子は変更されません。
7. DELETE文を実行しても、そのレコード領域の内容、およびファイル名-1のファイル記述項のRECORD句のDEPENDING ON指定に指定したデータ項目の内容は変更されません。
8. DELETE文を実行すると、ファイル名-1の入出力状態の値が更新されます。
9. DELETE文の実行が成功すると、存在するレコードのロックが解除されます。
10. DELETE文の実行中に無効キー条件が発生すると、ファイル名-1の入出力状態に無効キー条件を示す値が設定された後、“6.3.12 INVALID KEY指定”の規則に従って制御が移ります。
11. ELETE文の実行中に例外条件が発生しなかった場合、ファイル名-1の入出力状態が設定された後、“6.3.12 INVALID KEY指定”の規則に従って制御が移ります。
12. END-DELETE指定は、DELETE文の範囲を区切ります。

相対ファイルの規則

ファイル名-1に乱呼出し法または動的呼出し法のファイルを指定した場合、DELETE文を実行する前に、削除するレコードの相対レコード番号を相対キー項目に設定する必要があります。DELETE文を実行すると、相対レコード番号を持つレコードがファイルから論理的に取り除かれます。相対レコード番号を持つレコードがファイル中に存在しない場合は、無効キー条件が発生し、DELETE文の実行は不成功になります。

索引ファイルの規則

ファイル名-1に乱呼出し法または動的呼出し法のファイルを指定した場合、DELETE文を実行する前に、削除するレコードのキー値を主レコードキー項目に設定する必要があります。DELETE文を実行すると、そのキー値を持つレコードがファイルから論理的に取り除かれます。そのキー値を持つレコードがファイル中に存在しない場合は、無効キー条件が発生し、DELETE文の実行は不成功になります。

6.4.12 DISPLAY文（中核）

少量のデータを表示します。

【書き方】

```
DISPLAY { 一意名-1 } … [UPON 呼び名-1]
        { 定数-1  }
        [WITH NO ADVANCING]
```

構文規則

1. 定数-1に数字定数を書く場合、符号なし整数でなければなりません。

   ただし、このコンパイラでは、定数-1に符号付きまたは小数部を持つ数字定数を指定することもできます。

2. 呼び名-1は、環境部の特殊名段落で、以下のいずれかの機能名と対応付けておかなければなりません。

   - CONSOLE
   - SYSOUT
   - SYSERR
   - SYSPUNCH

一般規則

1. DISPLAY文は、一意名-1または定数-1の内容を、それらを指定した順にハードウェア装置に転送します。
2. 定数-1に表意定数を書いた場合、表意定数の1回の繰返しだけが表示されます。
3. 一意名-1を書いた場合、以下の形式で表示されます。

   a. 一意名-1に、集団項目または用途が表示用の基本項目(SEPARATE指定のない符号付き外部10進項目を除く)を指定した場合、一意名-1の内容が変換されずにそのまま表示されます。

   b. 一意名-1に、2進項目、内部10進項目、指標データ項目、またはSEPARATE指定のない符号付き外部10進項目を指定した場合、SIGN LEADING SEPARATE指定の外部10進項目に変換され、以下の形式で表示されます。

      - 一意名－1が符号付きの場合　：s9(n)9(m)
      - 一意名－1が符号なしの場合　：9(n)9(m)
      - 一意名－1が指標データ項目の場合：9(n)

      ここで“s”は符号(＋または－)を表し、“9(n)”と“9(m)”は、それぞれ整数部の桁数と小数部の桁数を表します。指標データ項目の場合、nの値は9です。ただし、[WinIPF][Winx64][LinuxIPF][Linux64]では、指標データ項目の場合、nの値は18です。

c. 一意名-1に、int型2進整数データ項目を指定した場合、“表5.1 int型2進整数データ項目の大きさと値の範囲”に定められている最大値を格納できる桁数を持つ用途がCOMP-5の2進項目とみなし、b)の規則が適用されます。

下表にint型2進整数データ項目の表示形式を示します。

一意名-1の種別	最大値	表示形式(*1)
BINARY-CHAR UNSIGNED	+255	9(3)
BINARY-SHORT SIGNED	+32767	s9(5)
BINARY-LONG SIGNED	+2147483647	s9(10)
BINARY-DOUBLE SIGNED	+9223372036854775807	s9(19)

*1: ここで“s”は符号(＋または－)を表し、“9(n)”は、整数部を表します。

d. 一意名-1に内部ブール項目を指定した場合、外部ブール項目に変換されて表示されます。

e. 一意名-1に内部浮動小数点項目を指定した場合、外部浮動小数点項目に変換され、以下の形式で表示されます。

- 単精度内部浮動小数点項目の場合 ： s.9(8)Es99
- 倍精度内部浮動小数点項目の場合 ： s.9(17)Es99

ここで、“s”は符号(+または-)を表し、“9(8)”と“9(17)”および“99”は数字の桁数を表します。

f. 一意名-1に日本語項目または日本語編集項目を指定した場合、呼び名指定なしのCHARACTER TYPE IS MODE-2句またはCHARACTER TYPE IS MODE-1句を一意名-1に指定したものとして表示されます。

g. 一意名-1にポインタデータ項目を指定した場合、16進表現で表示されます。

4. DISPLAYの後に2つ以上の作用対象を書いた場合、すべての作用対象のバイト数(変換が必要ならば、変換された後のバイト数)の和が、転送されるデータのバイト数になります。

5. DISPLAY文を実行すると、一意名-1または定数-1の内容が、それらを書いた順に以下のように転送され表示されます。以下で、“転送されるデータ”とは、UPONの前に書いた作用対象の内容を、それらを書いた順に結合したデータのことです。

   a. 転送されるデータのバイト数が1回に転送できるバイト数と同じ場合、データがそのまま転送され表示されます。

   b. 転送されるデータのバイト数が1回に転送できるバイト数より大きい場合、データが繰り返し転送され表示されます。各繰返しでは、データの左端を装置の左端に合わせて転送されます。繰返しの最後の転送では、a.またはc.の規則が適用されます。

   c. 転送されるデータのバイト数が1回に転送できるバイト数より小さい場合、転送されるデータの左端が装置の左端に合わせて転送され表示されます。装置が可変長のレコードを受け入れていない場合は、データが転送されなかった部分には、空白が埋められます。

6. UPON指定を省略した場合、SYSOUTに対応付けた呼び名を呼び名-1に指定したものとみなされます。

7. NO ADVANCING指定を省略した場合、最後の作用対象をハードウェア装置に転送した後、ハードウェア装置の行位置が、次の行の最左端に位置付けられます。

8. NO ADVANCING指定を書いた場合、最後の作用対象を表示した後、改行されません。最後の作用対象を表示した後、ハードウェア装置は以下のように行送りを行います。

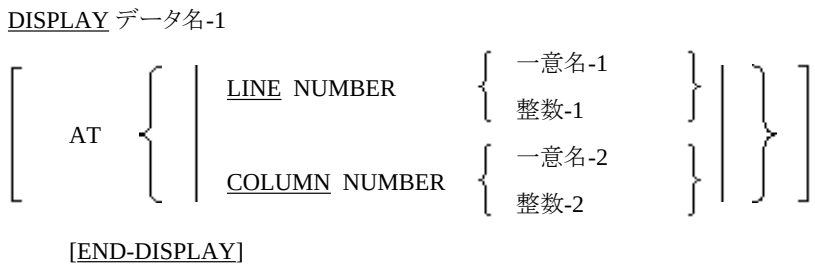
   a. 特定の文字位置に位置決めができるハードウェア装置の場合、最後に表示された作用対象の最後の文字の直後の文字位置に、位置決めされたままになります。

   b. 特定の文字位置に位置決めができないハードウェア装置の場合、可能ならば縦の行送りの抑制だけが行われます。ハードウェア装置が重ね打ちの機能を持つ場合は、すでに表示された行に重ね打ちされます。

6.4.13 DISPLAY文（スクリーン操作）

データを画面に表示します。

【書き方】



注意

[WinIPF][Linux][LinuxIPF][Linux64][.NET]では、スクリーン操作機能は使用できません。

構文規則

1. データ名-1は、画面節で定義した画面項目でなければなりません。その画面項目は、以下のいずれかでなければなりません。
   a. VALUE句を指定した基本画面項目
   b. PICTURE句にFROM指定またはUSING指定を書いた基本画面項目
   c. a.またはb.の基本画面項目を従属する集団画面項目
2. データ名-1は、修飾することもできます。
3. 整数-1および整数-2は、符号なしの整数でなければなりません。
4. 一意名-1および一意名-2は、符号なしの整数項目でなければなりません。
5. LINE NUMBER指定とCOLUMN NUMBER指定は、任意の順に書くことができます。これらの指定の両方を書くこともできます。

一般規則

1. DISPLAY文は、以下のデータを、データ名-1の画面項目に対応する画面上の領域に表示します。
   a. データ名-1に定数項目を指定した場合、VALUE句に指定した値を表示します。
   b. データ名-1に出力項目または更新項目を指定した場合、PICTURE句のFROM指定またはUSING指定に書いたデータ項目の値を画面項目へ転記し、そのデータを表示します。データ項目から画面項目への転記は、転記の規則に従って行います。
   c. データ名-1に集団画面項目を指定した場合、その画面項目に従属するすべての画面項目を表示します。
2. LINE NUMBER指定では、データ名-1の画面項目に対応する画面の行番号を、物理画面の第1行を1として指定します。
3. COLUMN NUMBER指定では、データ名-1の画面項目に対応する画面の列番号を、物理画面の第1列を1として指定します。
4. AT指定を省略した場合、画面の第1行、第1列から表示操作が行われます。

6.4.14 DISPLAY文（コマンド行引数と環境変数の操作）

コマンド行の引数の位置付け、環境変数を位置付けおよび環境変数に値の設定を行います。

【書き方】

DISPLAY { 一意名-1 / 定数-1 } … UPON 呼び名-1

[ON EXCEPTION 無条件文-1]

[NOT ON EXCEPTION 無条件文-2]

[END-DISPLAY]

構文規則

1. 呼び名-1は、環境部の特殊名段落で、以下のいずれかの機能名に対応付けておかなければなりません。
   - ARGUMENT-NUMBER
   - ENVIRONMENT-NAME
   - ENVIRONMENT-VALUE
2. 一意名-1および定数-1は、以下の条件を満足しなければなりません。
   a. 呼び名-1に、機能名ARGUMENT-NUMBERに対応付けた呼び名を指定する場合、一意名-1は符号なし整数項目でなければなりません。定数-1は、符号なし整数でなければなりません。
   b. 呼び名-1に、機能名ENVIRONMENT-NAMEまたはENVIRONMENT-VALUEに対応付けた呼び名を指定する場合、一意名-1は可変反復データ項目を従属しない集団項目または英数字項目でなければなりません。定数-1は、文字定数でなければなりません。
3. ON EXCEPTION指定およびNOT ON EXCEPTION指定は、呼び名-1に機能名ENVIRONMENT-VALUEに対応付けた呼び名を指定する場合にだけ、書くことができます。

一般規則

呼び名-1を機能名ARGUMENT-NUMBERに対応付けた場合の規則

一意名-1または定数-1では、コマンド行の引数の位置を指定します。一意名-1および定数-1の値は、0～99でなければなりません。DISPLAY文を実行すると、引数位置指示子が、一意名-1または定数-1に設定した位置に位置付けられます。

呼び名-1を機能名ENVIRONMENT-NAMEに対応付けた場合の規則

一意名-1または定数-1では、環境変数の名前を指定します。DISPLAY文を実行すると、一意名-1または定数-1に設定した名前を持つ環境変数に対して、入出力操作が可能になります。

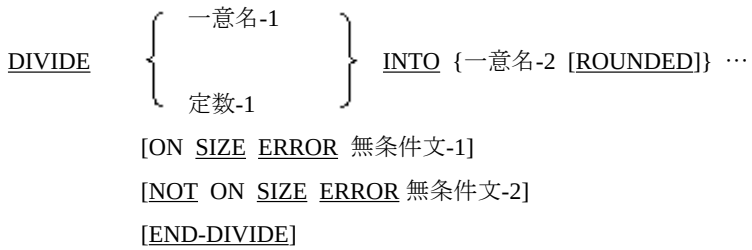
呼び名-1を機能名ENVIRONMENT-VALUEに対応付けた場合の規則

1. 一意名-1または定数-1では、環境変数に設定する値を指定します。
2. DISPLAY文を実行する前に、機能名ENVIRONMENT-NAMEに対応付けた呼び名を指定したDISPLAY文を実行しなければなりません。DISPLAY文を実行すると、先行するDISPLAY文で位置付けられた環境変数に、一意名-1または定数-1に設定した値が設定されます。
3. ON EXCEPTION指定を書いた場合、以下のときに無条件文-1が実行されます。
   a. DISPLAY文を実行する前に、環境変数を位置付けるためのDISPLAY文を実行しなかった場合
   b. 環境変数に値を設定するための十分な領域を確保できなかった場合
4. NOT ON EXCEPTION指定を書いた場合、2.に示した例外条件が発生しなかったとき、無条件文-2が実行されます。

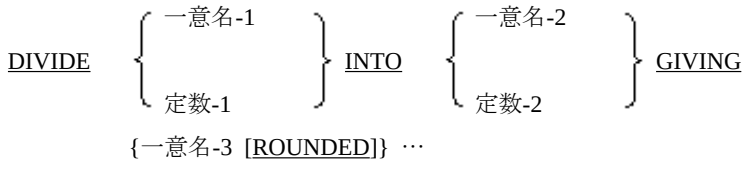
6.4.15 DIVIDE文（中核）

除算を行い、その商と剰余を求めます。

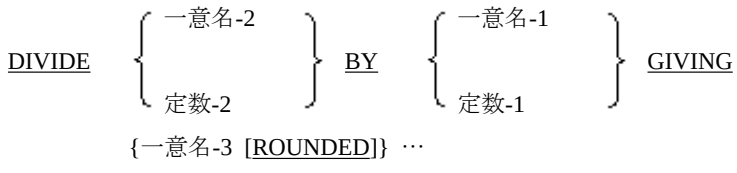
【書き方1】商を求め、被除数と置き換える



【書き方2】商を求め、被除数とは異なるデータ項目に格納する（その1）



【書き方3】商を求め、被除数とは異なるデータ項目に格納する（その2）



【書き方4】商と剰余を求める（その1）

【書き方5】商と剰余を求める（その2）

```
DIVIDE { 一意名-2 } BY { 一意名-1 } GIVING 一意名-3
       { 定数-2  }     { 定数-1  }
  [ROUNDED] REMAINDER 一意名-4
  [ON SIZE ERROR 無条件文-1]
  [NOT ON SIZE ERROR 無条件文-2]
  [END-DIVIDE]
```

構文規則

1. 一意名-1および一意名-2は、数字項目でなければなりません。
2. 一意名-3および一意名-4は、数字項目または数字編集項目でなければなりません。
3. 定数-1および定数-2は、数字定数でなければなりません。

一般規則

書き方1の規則

書き方1のDIVIDE文は、INTOの前の作用対象で一意名-2を割り、商を一意名-2に格納します。一意名-2の並びを書いた順に、この除算を行います。

書き方2の規則

書き方2のDIVIDE文は、INTOの前の作用対象でINTOの後の作用対象を割り、商を一意名-3に格納します。一意名-3の並びを書いた順に、この商を格納します。

書き方3の規則

書き方3のDIVIDE文は、BYの前の作用対象をBYの後の作用対象で割り、商を一意名-3に格納します。一意名-3の並びを書いた順に、この商を格納します。

書き方4の規則

書き方4のDIVIDE文は、INTOの前の作用対象でINTOの後の作用対象を割り、商を一意名-3に格納し、剰余を一意名-4に格納します。

書き方5の規則

書き方5のDIVIDE文は、BYの前の作用対象をBYの後の作用対象で割り、商を一意名-3に格納し、剰余を一意名-4に格納します。

書き方4および書き方5に共通する規則

1. 一意名-4を添字付けした場合、その添字は、剰余が一意名-4に格納される直前に評価されます。
2. 剰余は、以下の式に従って求められます。

   剰余　＝　被除数　－　商　×　除数

   被除数は、一意名-2または定数-2の値です。除数は、一意名-1または定数-1の値です。剰余を計算するために使う商として、以下の値が使われます。

   a. 一意名-3に数字編集項目を指定した場合、PICTURE句で指定した形式に編集される前の値が使われます。

   b. ROUNDED指定を書いた場合、商を四捨五入する前の切り捨てられた値が使われます。この「剰余を計算するために使う商」の桁数、小数点位置および符号の有無は、一意名-3と同じです。

   c. 上記以外の場合、一意名-3の値が使われます。

3. 一意名-4の精度は、2.の規則に従って決定されます。一意名-4には、必要に応じて小数点の位置合わせと切捨てが行われた値が設定されます。
4. ON SIZE ERROR指定を書いた場合、実行結果は以下のようになります。
    a. 商に桁あふれが起こった場合、一意名-3および一意名-4の内容は変更されません。
    b. 剰余に桁あふれが起こった場合、一意名-4の内容は変更されません。商と剰余のどちらで桁あふれが起こったかは、利用者が調べる必要があります。

書き方1～書き方5に共通する規則

1. END-DIVIDE指定は、DIVIDE文の範囲を区切ります。
2. ROUNDED指定、ON SIZE ERROR指定、演算および転記の規則については、“6.3 文に関する共通の規則”を参照してください。

6.4.16 ENTRY文（プログラム間連絡）

呼ばれるプログラムの二次入口点を指定します。

【書き方】

```
ENTRY 定数-1
        [WITH { C       } LINKAGE]
              { PASCAL  }
              { STDCALL }
        [USING {データ名-1} … ]
```

注意

WITH指定は[Win16][Win32]固有機能です。

[.NET]では、ENTRY文は使用できません。

構文規則

1. 定数-1は、文字定数でなければなりません。定数-1は、プログラム名の記述規則に従わなければなりません。プログラム名の記述規則については“3.1.1 プログラム名段落（PROGRAM-ID）”を参照してください。
2. 定数-1は、実行単位中の他の入口名および他のプログラム名と同じであってはいけません。
3. USING指定の規則については、“6.2 手続き部の見出し”を参照してください。
4. USING指定に書くことができるデータ名-1の個数の最大値については、“付録B システムの定量制限”を参照してください。
5. 手続き部の見出しにRETURNING指定を書いた場合、その手続き部にENTRY文を書いてはなりません。

WITH指定の規則

WITH指定は、呼出し規約を指定します。WITH指定を省略した場合、COBOLの定めたリンケージの規約を使用します。

一般規則

1. ENTRY文は、二次入口点を指定します。二次入口点の名前は、定数-1で指定します。CALL文の定数または一意名に、二次入口点の名前を指定した場合、CALL文を実行したときに二次入口点に制御が移ります。呼ぶプログラムからパラメタを受け取る場合、USING指定を書きます。
2. CALL文以外の実行によってENTRY文に制御が移った場合、何もしないでENTRY文の次の文に制御が移ります。
3. 内部プログラムにENTRY文を書くことはできません。

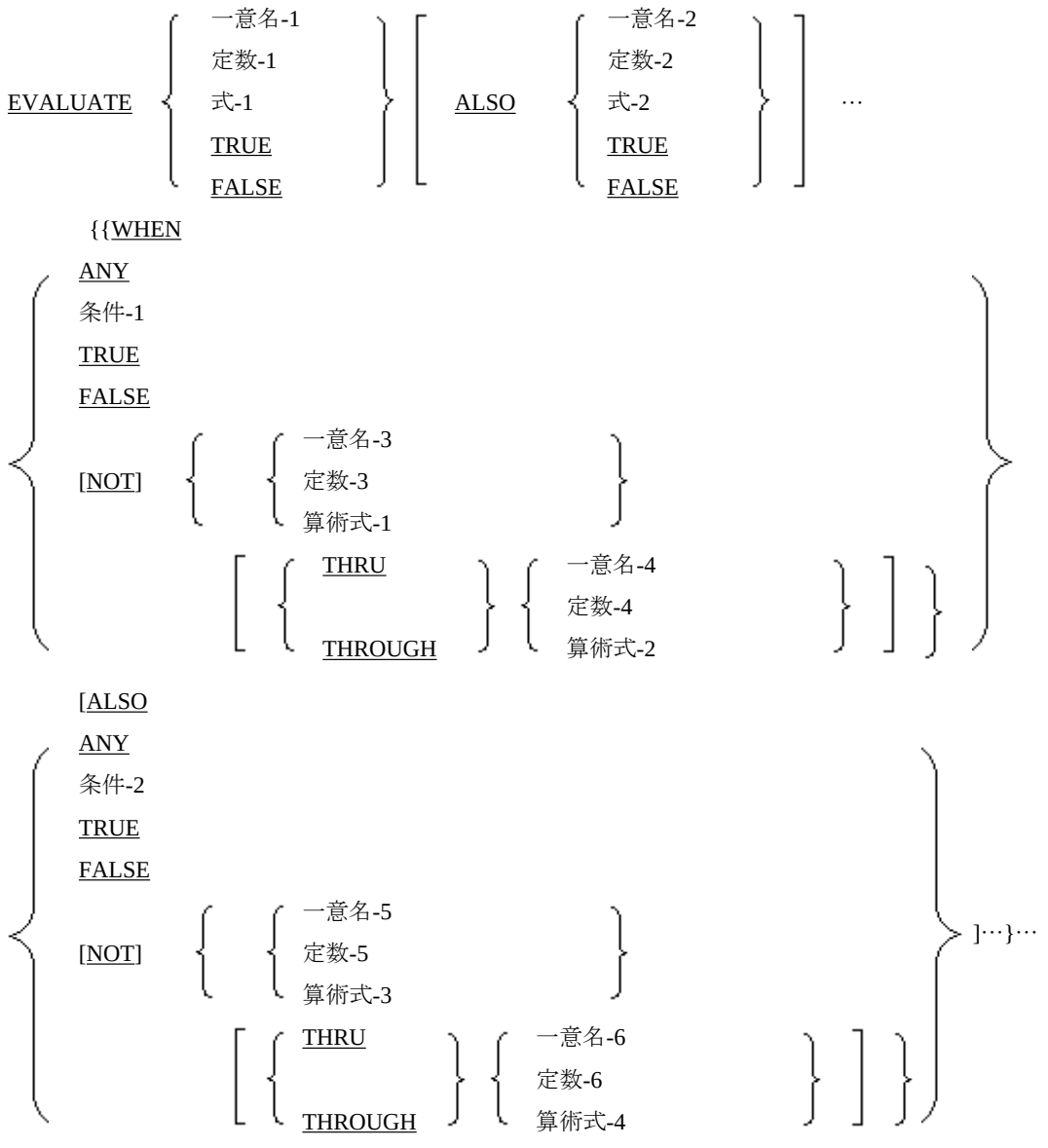
WITH指定の規則

同一翻訳単位内で、最外部プログラムの手続き部の見出しとENTRY文のWITH指定が異なってはなりません。

6.4.17 EVALUATE文（中核）

複数の条件を評価し、評価結果に対応する文を実行します。

【書き方】



[WHEN OTHER 無条件文-2]

[END-EVALUATE]

構文規則

1. EVALUATE文の最初のWHEN指定より前に書く作用対象(TRUEおよびFALSEも含む)を、「選択主体」といいます。
2. EVALUATE文のWHEN指定に書く作用対象(TRUE、FALSEおよびANYも含む)を、「選択対象」といいます。WHEN指定にTHROUGH指定を書いた場合は、THROUGH指定でつながる2つの作用対象が1つの選択対象になります。
3. THRUとTHROUGHは同義語です。
4. THROUGH指定でつながる2つの作用対象は、同じ字類の一意名、定数または算術式でなければなりません。
5. 1つのWHEN指定の中には、選択主体の数と同じ数の選択対象を書かなければなりません。
6. 1つのWHEN指定の中の各選択対象は、1番目の選択対象と1番目の選択主体、2番目の選択対象と2番目の選択主体というように、選択主体に対応付けて書かなければなりません。選択対象と選択主体を対応付けるときの規則は、以下のとおりです。
   a. 選択対象に一意名、定数または算術式を書く場合、選択主体には、選択対象と比較できる作用対象を書かなければなりません。
   b. 選択対象に条件-1、条件-2またはTRUEまたはFALSEを書く場合、選択主体には、条件式、TRUEまたはFALSEを書かなければなりません。
   c. 選択対象にANYを書く場合、任意の選択主体を書くことができます。
7. THROUGH指定でつながる2つの作用対象に、ブール項目またはブール定数を書くことはできません。
8. 式-1および式-2に、ブール式を書くことはできません。

一般規則

1. EVALUATE文は、それぞれの選択主体の値と選択対象の値を評価して条件を満足するWHEN指定を選択し、そのWHEN指定の無条件文を実行します。
2. それぞれの選択主体と選択対象には、以下の値が割り当てられます。
   a. 選択主体として一意名-1または一意名-2を書いた場合、選択主体には、その一意名の値と字類が割り当てられます。同様に、選択対象としてNOT指定もTHROUGH指定もない一意名-3または一意名-5を書いた場合、選択対象には、その一意名の値と字類が割り当てられます。
   b. 選択主体として定数-1または定数-2を書いた場合、選択主体には、その定数の値と字類が割り当てられます。同様に、選択対象としてNOT指定もTHROUGH指定もない定数-3または定数-5を書いた場合、選択対象には、その定数の値と字類が割り当てられます。ただし、定数-3または定数-5に表意定数ZEROを指定した場合、選択対象の字類には対応する選択主体の字類が割り当てられます。
   c. 選択主体の式-1または式-2に算術式を書いた場合、選択主体には、算術式の評価の規則に従って求められた数値が割り当てられます。同様に、選択対象としてNOT指定もTHROUGH指定もない算術式-1または算術式-3を書いた場合、選択対象には、算術式の評価の規則に従って求められた数値が割り当てられます。
   d. 選択主体の式-1または式-2に条件式を書いた場合、選択主体には、条件式の評価の規則に従って求められた真理値が割り当てられます。同様に、選択対象として条件-1、条件-2を書いた場合、選択対象には、条件式の評価の規則に従って求められた真理値が割り当てられます。
   e. 選択主体としてTRUEまたはFALSEを書いた場合、選択主体には真理値が割り当てられます。TRUEを書いた場合は真理値“真”が割り当てられ、FALSEを書いた場合は真理値“偽”が割り当てられます。同様に、選択対象としてTRUEまたはFALSEを書いた場合、選択対象には真理値が割り当てられます。
   f. 選択対象としてANYを書いた場合、選択対象は評価されません。

g. 選択対象にTHROUGH指定を書きNOT指定を省略した場合、選択主体の値のうち、THROUGH指定の直前の作用対象以上でTHROUGH指定の直後の作用対象以下であるすべての値が、選択対象に割り当てられます。

h. 選択対象にNOT指定を書きTHROUGH指定を省略した場合、NOT指定を書かなかったときに割り当てられる値と等しくない値が、選択対象に割り当てられます。同様に、選択対象にNOT指定とTHROUGH指定の両方を書いた場合、NOT指定を書かなかったときに割り当てられる値の範囲に含まれない値が、選択対象に割り当てられます。ここで、“NOT指定を書かなかったときに割り当てられる値”は、上記のa.～h.で説明した値です。

3. 2.の規則に従って割り当てられた選択主体の値と選択対象の値が比較され、条件を満足するWHEN指定が決定されます。この比較は、以下のように行われます。

   a. 1つのWHEN指定の中の各選択対象が、1番目の選択対象と1番目の選択主体、2番目の選択対象と2番目の選択主体というように、対応する選択主体と比較されます。比較の規則は以下のとおりです。

      選択対象と選択主体の値が2.のa.、b.、c.、g.、またはh.の規則に従って割り当てられた場合、選択対象の値が、選択主体の値と比較されます。そして、選択対象に割り当てられた値が、選択主体に割り当てられた値と同じ場合、比較が真になります。

      選択対象の値が2.のd.またはe.の規則に従って割り当てられた場合、選択対象の真理値と選択主体の真理値が比較されます。そして、真理値が同じ場合、この比較が真になります。

      選択対象としてANYを書いた場合、選択主体の値に関係なく、つねに比較が真になります。

   b. 1つのWHEN指定の中のすべての選択対象について a.の比較が真になった場合、そのWHEN指定が条件を満足するWHEN指定として選択されます。

   c. 上記のa.～b.が、WHEN指定を書いた順に、条件を満足するWHEN指定が選択されるか、またはすべてのWHEN指定が処理されるまで繰り返されます。

4. 3.の比較が完了した後、EVALUATE文の実行は以下のように進みます。

   a. 条件を満足するWHEN指定が選択された場合、選択されたWHEN指定に続く最初の無条件文-1が実行されます。そして、無条件文-1の実行後、EVALUATE文の実行は終了します。

   b. WHEN指定が選択されなかった場合、WHEN OTHER指定を書いたときは、無条件文-2が実行されます。そして、無条件文-2の実行後、EVALUATE文の実行は終了します。

   c. WHEN指定が選択されなかった場合、WHEN OTHER指定を省略したときは、EVALUATE文の実行は終了します。

5. END-EVALUATE指定は、EVALUATE文の範囲を区切ります。

6. 条件式、算術式および比較の規則については、“6.3 文に関する共通の規則”を参照してください。

例

EVALUATE文の例(その1)

```
*〔データ部〕
  01  MAIL-RECORD.
      02  KIND     PIC X.
      02  WEIGHT   PIC 9(4).
      02  CHARGE   PIC 9(5).
      02  FLAG     PIC X(5).
*〔手続き部〕
     EVALUATE KIND ALSO WEIGHT
       WHEN "1" ALSO 0 THRU 25
         MOVE 60 TO CHARGE
       WHEN "1" ALSO 26 THRU 50
         MOVE 70 TO CHARGE
       WHEN "2" ALSO ANY
         MOVE 40 TO CHARGE
       WHEN OTHER
```

```
        MOVE "ERROR" TO FLAG
    END-EVALUATE.
```

データ部とEVALUATE文を上記のように書いた場合、EVALUATE文は、KINDとWEIGHTの値に従って下表のように実行されます。

条件		実行される文
KINDの値	WEIGHTの値	
"1"	0～25	MOVE 60 TO CHARGE
"1"	26～50	MOVE 70 TO CHARGE
"2"	どの値でも	MOVE 40 TO CHARGE
上記以外		MOVE "ERROR" TO FLAG

例

EVALUATE文の例(その2)

```
*〔データ部〕
  01  EMPLOYMENT-RECORD.
      02  SEX       PIC X.
      02  AGE       PIC 9(2).
      02  MARKS     PIC 9(3).
      02  RESULT    PIC X(4).
*〔手続き部〕
     EVALUATE SEX ALSO AGE ALSO TRUE
       WHEN "F" ALSO 18 THRU 22 ALSO
         MARKS > 74
       WHEN "F" ALSO 23 THRU 29 ALSO
         MARKS > 84
       WHEN "M" ALSO 18 THRU 39 ALSO
         MARKS > 79
         MOVE "PASS" TO RESULT
       WHEN OTHER MOVE "FAIL" TO RESULT
     END-EVALUATE.
```

データ部とEVALUATE文を上記のように書いた場合、EVALUATE文は、SEX、AGEおよびMARKSの値に従って下表のように実行されます。

条件			実行される文
SEXの値	AGEの値	MARKSの値	
"F"	18～22	>74	MOVE "PASS" TO RESULT
"F"	23～29	>84	
"M"	18～39	>79	
上記以外			MOVE "FAIL" TO RESULT

6.4.18 EXIT文（中核）

一連の手続きの共通の出口を指定します。

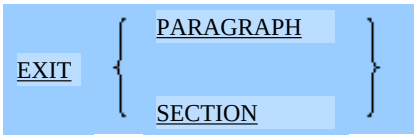
参照

[.NET]のEXIT文は、“11.8.3.6 EXIT文”を参照してください。

【書き方1】

EXIT

【書き方2】



構文規則

書き方1の規則

EXIT文だけで一つの完結文になっていなければならず、その段落にはこの完結文だけを書きます。

書き方2の規則

SECTION指定のEXIT文は、節中になければなりません。

一般規則

書き方1の規則

EXIT文は、利用者がプログラム上のある地点に手続き名を付けるためにだけ書きます。EXIT文は、プログラムの翻訳および実行においてそのほかの影響を与えません。

書き方2の規則

1. PARAGRAPH指定のEXIT文の実行は、そのEXIT文が含まれる段落の最後の文の直後に暗黙のCONTINUE文を補い、そこに制御を移行します。
2. SECTION指定のEXIT文の実行は、そのEXIT文が含まれる節中の最後の段落の最後の文の直後に暗黙のCONTINUE文を補い、そこに制御を移行します。

6.4.19 EXIT PERFORM文（中核）

うちPERFORM文の出口を指定します。

【書き方】

EXIT [TO TEST OF] PERFORM

一般規則

1. EXIT PERFORM文は、うちPERFORM文の出口を指定します。EXIT PERFORM文は、うちPERFORM文の中にだけ書くことができます。
2. EXIT PERFORM文は、それが含まれる最も内側のうちPERFORM文と対応付けられます。
3. TEST指定なしのEXIT PERFORM文を実行すると、対応するうちPERFORM文の終わりに制御が移ります。

4. TEST指定付きのEXIT PERFORM文は、終了条件付きのPERFORM文(書き方2、書き方3または書き方4のPERFORM文)の中にだけ書くことができます。TEST指定付きのEXIT PERFORM文を実行すると、対応するうちPERFORM文の検査機構に制御が移ります。検査機構とは、以下の処理のことです。
    a. 書き方2のPERFORM文で、TIMESの前に指定した繰り返し回数を検査する処理
    b. 書き方3および書き方4のPERFORM文で、UNTILの後に指定した条件を検査する処理

6.4.20 EXIT PROGRAM文（プログラム間連絡）

呼ばれるプログラムの論理的な終わりを指定します。

【書き方】

EXIT PROGRAM

構文規則

1. 完結文中の一連の無条件文の並びにEXIT PROGRAM文を書く場合、EXIT PROGRAM文はその並びの最後でなければなりません。
2. GLOBAL指定付きのUSE文の宣言手続きの中に、EXIT PROGRAM文を書くことはできません。

一般規則

1. EXIT PROGRAM文は、呼び出されたプログラムの論理的な終わりを指定します。EXIT PROGRAM文を実行すると、EXIT PROGRAM文を書いたプログラムを呼び出した地点の直後に制御が戻ります。CALL文の実行によってプログラムを呼び出した場合、EXIT PROGRAM文を実行すると、CALL文の直後に制御が戻ります。
2. EXIT PROGRAM文を実行した後の、呼び出したプログラムの状態は、呼び出したプログラムと呼び出されたプログラムで共有しているデータ項目およびファイルの内容を除いて、CALL文を実行したときの状態と同じです。
3. EXIT PROGRAM文を実行した後の、呼び出されたプログラム(EXIT PROGRAM文を実行したプログラム)の状態は、以下のとおりです。
    a. そのプログラムが初期化属性を持たない場合、PERFORM文の範囲が決定されていることを除いて、呼び出される前の状態と同じです。
    b. そのプログラムが初期化属性を持つ場合、そのプログラムに対してCANCEL文を実行した結果と同じです。

6.4.21 GENERATE文（報告書作成）

報告書記述に従って報告書を作成します。

【書き方】

GENERATE { データ名-1 / 報告書名-1 }

注意

[WinIPF][Winx64][LinuxIPF][Linux64][.NET]では報告書作成機能は使用できません。

構文規則

1. データ名-1は、明細報告集団の名前でなければなりません。データ名-1は、報告書名で修飾することができます。
2. 報告書名-1を書けるのは、その報告書記述項が以下の条件をすべて満たす場合だけです。
    a. CONTROL句を含む。
    b. 2つ以上の明細報告集団の記述を含まない。
    c. 本体集団の記述を1つ以上含む。

一般規則

1. 報告書名-1を指定したGENERATE文に対して、報告書作成管理システムは合計報告の処理を行います。報告書に対するGENERATE文が、すべて報告書名-1を指定したGENERATE文の形式であれば、作成表示される報告書を「合計報告書」といいます。合計報告書は、明細報告集団が表示されない報告書です。
2. データ名-1を指定したGENERATE文に対して、報告書作成管理システムは、指定された明細報告集団に特有の処理を含めて明細処理を行います。通常、データ名-1を指定したGENERATE文を実行すると、報告書作成管理システムは、指定された明細報告集団を表示します。
3. 報告書作成管理システムは、報告書に対する最初のGENERATE文の実行中に制御データ項目の値を保存します。報告書作成管理システムは、同じ報告書に対する2回目以降のGENERATE文の実行中に制御切れを検出するまで、この保存した制御データ項目の値を使って制御切れが起こったかどうかを決めます。報告書作成管理システムは、制御切れが起こったときに新しい値を保存し、その後別の制御切れが起こるまで、その値を制御切れの検出に用います。
4. 報告書の表示中に、本体集団を表示するために改ページしなければならないとき、ページ頭書き報告集団およびページ脚書き報告集団が定義されていれば、報告書作成管理システムは、自動的にそれらの処理を行います。
5. 報告書作成管理システムは、報告書に対する最初のGENERATE文を実行するときに、以下に示す各報告集団が報告書の記述に定義されていれば、その順に処理します。報告書作成管理システムは、4.に従ってページ頭書き報告集団およびページ脚書き報告集団の処理も行います。
    a. 報告書頭書き報告集団の処理
    b. ページ頭書き報告集団の処理
    c. 高いレベルから低いレベルへのすべての制御頭書き報告集団の処理
    d. データ名-1を指定したGENERATE文を実行する場合は、指定された明細報告集団の処理
       報告書名-1を指定したGENERATE文を実行する場合は、明細報告処理の一部
6. 報告書作成管理システムは、報告書に対する2回目以降のGENERATE文を実行するときに、以下に示す処理を行います。報告書作成管理システムは、4.に従ってページ頭書き報告集団およびページ脚書き報告集団の処理も行います。
    a. 制御切れを検出します。制御データ項目が等しいかどうかを決める規則は、比較条件の規則と同じとします。制御切れが起こった場合に、以下の処理を行います。
        - 報告書作成管理システムが制御切れの検出に使用した制御データ項目の値を、制御脚書き報告集団に関連するUSE手続きおよび制御脚書き報告集団中のSOURCE句で参照できるようにします。
        - 制御脚書き報告集団を低いレベルから高いレベルへの順に処理します。制御切れが起こったうちで最も高いレベル以下のすべての制御脚書き報告集団だけを処理します。
        - 制御頭書き報告集団を高いレベルから低いレベルへの順に処理します。制御切れが起こったうちで最も高いレベル以下のすべての制御頭書き報告集団だけを処理します。
    b. データ名-1を指定したGENERATE文を実行する場合は、指定した明細報告集団を処理します。報告書名-1を指定したGENERATE文を実行する場合は、明細報告処理の一部を処理します。
7. 報告書に対するGENERATE文は、INITIATE文が実行された後で、かつ、TERMINATE文が実行される前でだけ実行できます。
8. 報告書の表示規則については、“5.7.8 TYPE句”および“5.8 報告集団の表示規則”を参照してください。

6.4.22 GO TO文（中核）

手続き部のある部分から別の部分に制御を移します。書き方1のGO TO文で手続き名-1を省略する書き方は、廃要素です。

【書き方1】特定の部分へ制御を移す

GO TO [手続き名-1]

【書き方2】データ項目の値に従って制御を移す

GO TO {手続き名-1} … DEPENDING ON 一意名-1

構文規則

書き方1の規則

1. ALTER文でGO TO文の行き先を変更する場合、ALTER文に指定した段落は、書き方1のGO TO文1個からなる1つの完結文だけで構成しなければなりません。
2. 手続き名-1を省略する場合、GO TO文は1つの完結文でなければなりません。この文を含む段落には、この文しか書くことができません。
3. 複数の無条件文からなる文の中にGO TO文を書く場合、GO TO文は無条件文の最後の文でなければなりません。

書き方2の規則

一意名-1は、整数項目でなければなりません。

一般規則

書き方1の規則

1. 書き方1のGO TO文は、制御を手続き名-1に移します。
2. 手続き名-1を省略する場合、GO TO文を実行する前に、GO TO文を参照するALTER文を実行しなければなりません。

書き方2の規則

手続き名-1の並びには、一意名-1の値が1、2、3、…、nであるときの制御の移行先を、左から順に指定します。書き方2のGO TO文は、一意名-1の値が1の場合は1番目の手続き名-1、2の場合は2番目の手続き名-1というように、制御を移します。一意名-1の値が整数1、2、…、n以外の場合、何もしないで次の文に制御を移します。

6.4.23 IF文（中核）

条件を評価し、評価結果に対応する文を実行します。

【書き方】

IF 条件-1

THEN { {文-1} … | NEXT SENTENCE }

```
{ ELSE {文-2} … [END-IF]          }
{ ELSE NEXT SENTENCE              }
{ END-IF                          }
```

構文規則

1. ELSE NEXT SENTENCE指定の直後に分離符の終止符を書く場合、ELSE NEXT SENTENCE指定を省略することができます。
2. NEXT SENTENCE指定とEND-IF指定の両方を書くことはできません。

一般規則

1. IF文は、条件-1を評価し、真の場合はTHEN指定の処理、偽の場合はELSE指定の処理を行います。
2. IF文の終わりは、以下のいずれかの方法で指定します。
   a. 同じ水準にあるEND-IF指定
   b. 分離符の終止符
   c. IF文が入れ子になっている場合、外側のIF文に対応するELSE指定

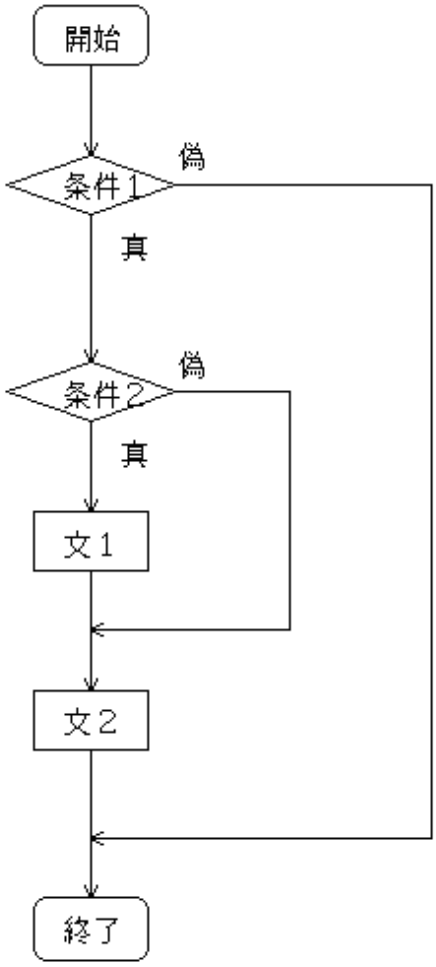
   IF文の範囲については、“6.1 手続き部の構成（PROCEDURE DIVISION）”の“文の範囲”を参照してください。
3. 条件-1の評価結果が真の場合、以下のように制御が移ります。
   a. 文-1を書いた場合、文-1の並びに指定した文を実行します。文-1の実行が終了した後、IF文の終わりに制御が移ります。ただし、文-1の中で、制御を明に移行する文を実行した場合は、その文の規則に従って制御が移ります。
   b. NEXT SENTENCEを書いた場合、IF文の次の実行完結文に、制御が移ります。
4. 条件-1の評価結果が偽の場合、THEN指定が無視され、ELSE指定に制御が移ります。その後、ELSE指定の記述に従って以下のように制御が移ります。
   a. 文-2を書いた場合、文-2の並びに指定した文を実行します。文-2の実行が終了した後、IF文の終わりに制御が移ります。ただし、文-2の中で、制御を明に移行する文を実行した場合は、その文の規則に従って制御が移ります。
   b. ELSE指定を省略した場合、IF文の終わりに制御が移ります。
   c. NEXT SENTENCE指定を書いた場合、IF文の次の実行完結文に、制御が移ります。
5. IF文が入れ子になっている場合、IF文の中のIF文は、左から右に向かってIF、ELSEおよびEND-IFの組を作っているものとみなされます。ELSEおよびEND-IFに対応するIFは、それらの左側にあってほかのELSEおよびEND-IFと組になっていないIFのうち、最も右側にあるIFです。
6. END-IF指定は、IF文の範囲を区切ります。
7. 条件式および比較の規則については、“6.3 文に関する共通の規則”を参照してください。

例

入れ子のIF文の例

```
IF  条件１ THEN
  IF  条件２ THEN
    文１
  END-IF
  文２
END-IF.
```

〔入れ子のIF文の論理的な流れ〕



6.4.24 INITIALIZE文（中核）

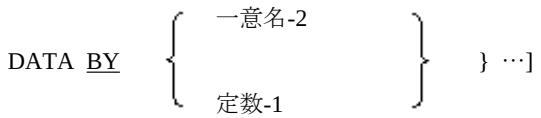
データ項目を初期化します。

使い方の例については、“付録H サンプル集”の“H.37 INITIALIZE文”を参照してください。

【書き方】

INITIALIZE {一意名-1}…

[REPLACING{ ALPHABETIC / ALPHANUMERIC / NUMERIC / ALPHANUMERIC-EDITED / NUMERIC-EDITED / NATIONAL / NATIONAL-EDITED / BOOLEAN }



構文規則

1. 一意名-2または定数-1の項類とREPLACING指定に書く項類の組合せは、前者を送出し側の項類、後者を受取り側の項類とするときの転記の規則に従うものでなければなりません。
2. REPLACING指定の並びで、同じ項類のREPLACING指定を繰り返すことはできません。
3. 一意名-1または一意名-1に従属するデータ項目のデータ記述項に、DEPENDING指定付きのOCCURS句を書くことはできません。
4. 一意名-1および一意名-2に、指標データ項目を指定することはできません。
5. 一意名-1に、RENAMES句を持つデータ項目を指定することはできません。

一般規則

1. INITIALIZE文は、一意名-1のデータ項目を初期化します。任意の項類を持つデータ項目を定められた値で初期化する場合、REPLACING指定を省略します。特定の項類を持つ基本項目を特定の値で初期化する場合、REPLACING指定を書きます。REPLACINGの後に初期化の対象とする項類を指定し、BYの後に初期値を指定します。
2. REPLACINGの後の語は、初期化の対象とする項類を表します。ALPHABETは英字、ALPHANUMERICは英数字、NUMERICは数字、ALPHANUMERIC-EDITEDは英数字編集、NUMERIC-EDITEDは数字編集を表します。

   NATIONALは日本語、NATIONAL-EDITEDは日本語編集、BOOLEANはブールを表します。
3. 一意名-1に以下を指定した場合、INITIALIZE文を実行しても、一意名-1の内容は変更されません。
   - 指標データ項目
   - ポインタデータ項目
4. 一意名-1に集団項目を指定した場合、集団項目に従属する基本項目のうち、以下の基本項目は初期化の対象になりません。
   - 指標データ項目
   - FILLER項目
   - ポインタデータ項目
   - REDEFINES句を指定したデータ項目、またはそのような項目に従属するデータ項目
5. REPLACING指定を書いた場合、以下のように初期化されます。
   a. 一意名-1に基本項目を指定した場合、その基本項目がREPLACING指定に書いた項類に属する場合にだけ、BY指定の作用対象の値で初期化されます。ただし、3.の場合を除きます。
   b. 一意名-1に集団項目を指定した場合、その集団項目に従属する基本項目のうち、REPLACING指定に書いた項類に属する基本項目だけが、BY指定の作用対象の値で初期化されます。集団項目が表を含む場合、その表の表要素である基本項目も初期化の対象になります。ただし、4.の場合を除きます。
   c. 初期化は、BY指定の作用対象を送出し側とし、初期化の対象となる基本項目を受取り側とするMOVE文を実行した場合と同じように行われます。
6. REPLACING指定を省略した場合、以下のように初期化されます。
   a. 一意名-1に基本項目を指定した場合、その基本項目が下記のc.に示す値で初期化されます。

b. 一意名-1に集団項目を指定した場合、その集団項目に従属する基本項目が、下記のc.に示す値で初期化されます。集団項目が表を含む場合、その表の表要素である基本項目も初期化の対象になります。ただし、4.の場合を除きます。

c. 初期化は、以下の値を持つ定数を送出し側とし、対象となる基本項目を受取り側とするMOVE文を実行した場合と同じように行われます。

- 対象となる基本項目の項類が英字、英数字または英数字編集の場合、送出し側の値は、英数字の空白です。
- 対象となる基本項目の項類が日本語項目または日本語編集項目の場合、送出し側の値は、日本語の空白です。
- 対象となる基本項目の項類が数字または数字編集の場合、送出し側の値は、数字のゼロです。
- 対象となる基本項目の項類がブールの場合、送出し側の値は、ブールのゼロです。

7. 一意名-1が集団項目の場合、対象となる基本項目への転記は、その集団項目の中で定義した順に初期化されます。
8. 一意名-1が一意名-2と同じ記憶領域を占有する場合、INITIALIZE文の実行結果は規定されません。

6.4.25 INITIATE文（報告書作成）

報告書の処理を開始します。

【書き方】

INITIATE［報告書名-1］…

注意

[WinIPF][Winx64][LinuxIPF][Linux64][.NET]では報告書作成機能は使用できません。

構文規則

報告書名-1は、データ部の報告書節の報告書記述項で指定しなければなりません。

一般規則

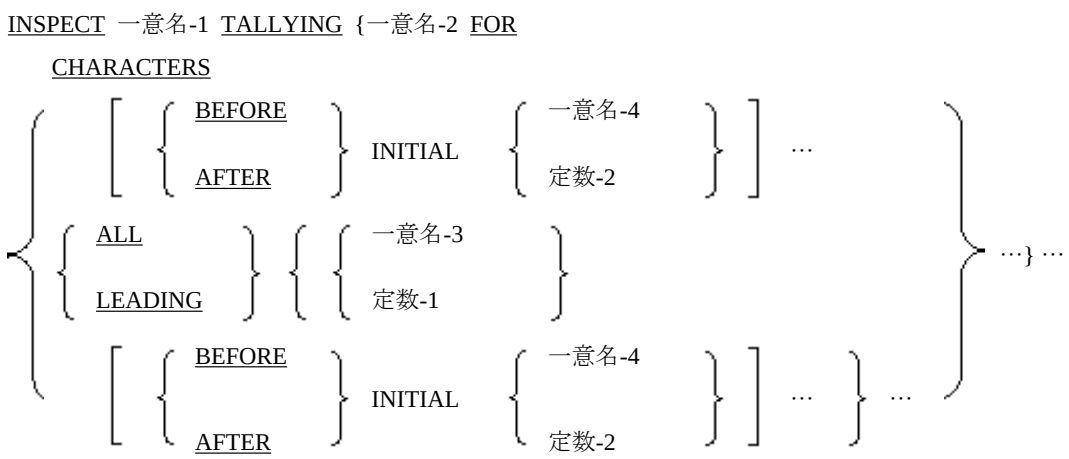
1. INITIATE文は、指定された各報告書について以下に示す初期化を行います。
   a. すべての合計カウンタに0を設定します。
   b. 行カウンタに0を設定します。
   c. ページカウンタに1を設定します。
2. INITIATE文は、この報告書に関するファイルを開いた状態にする機能は持ちません。INITIATE文の実行に先立ち、そのファイルを開くためにはOUTPUT指定またはEXTEND指定のOPEN文を実行しなければなりません。
3. 報告書名-1についてINITIATE文を実行した後は、TERMINATE文を実行する前に重ねてINITIATE文を実行することはできません。
4. 1つのINITIATE文に2つ以上の報告書名を書いたとき、このINITIATE文の実行結果は、各報告書名をこのINITIATE文に書いた順序で別々のINITIATE文に書いた場合と同じとします。

6.4.26 INSPECT文

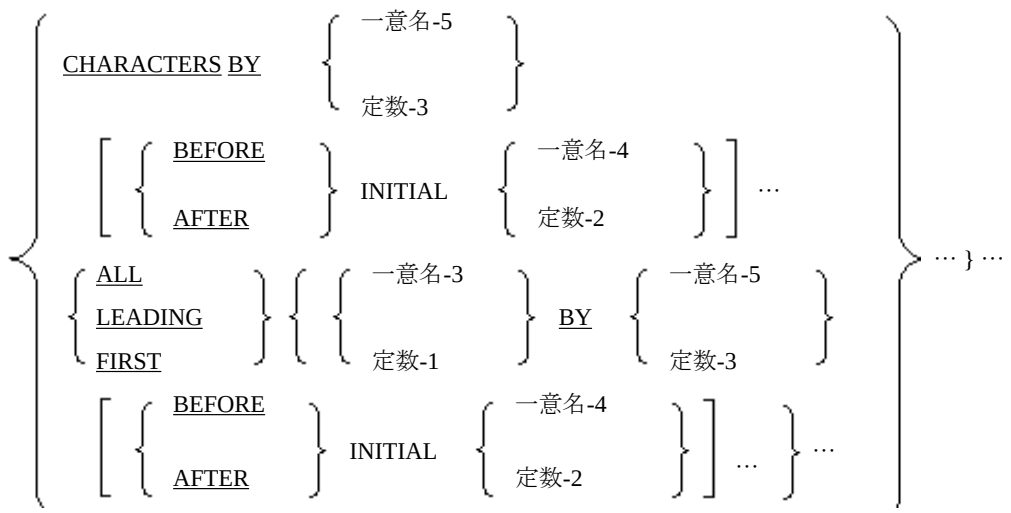
文字列の出現回数のカウントおよび文字列の置き換えを行います。

使い方の例については、“付録H サンプル集”の“H.38 INSPECT文”を参照してください。

【書き方1】文字列の出現回数のカウント

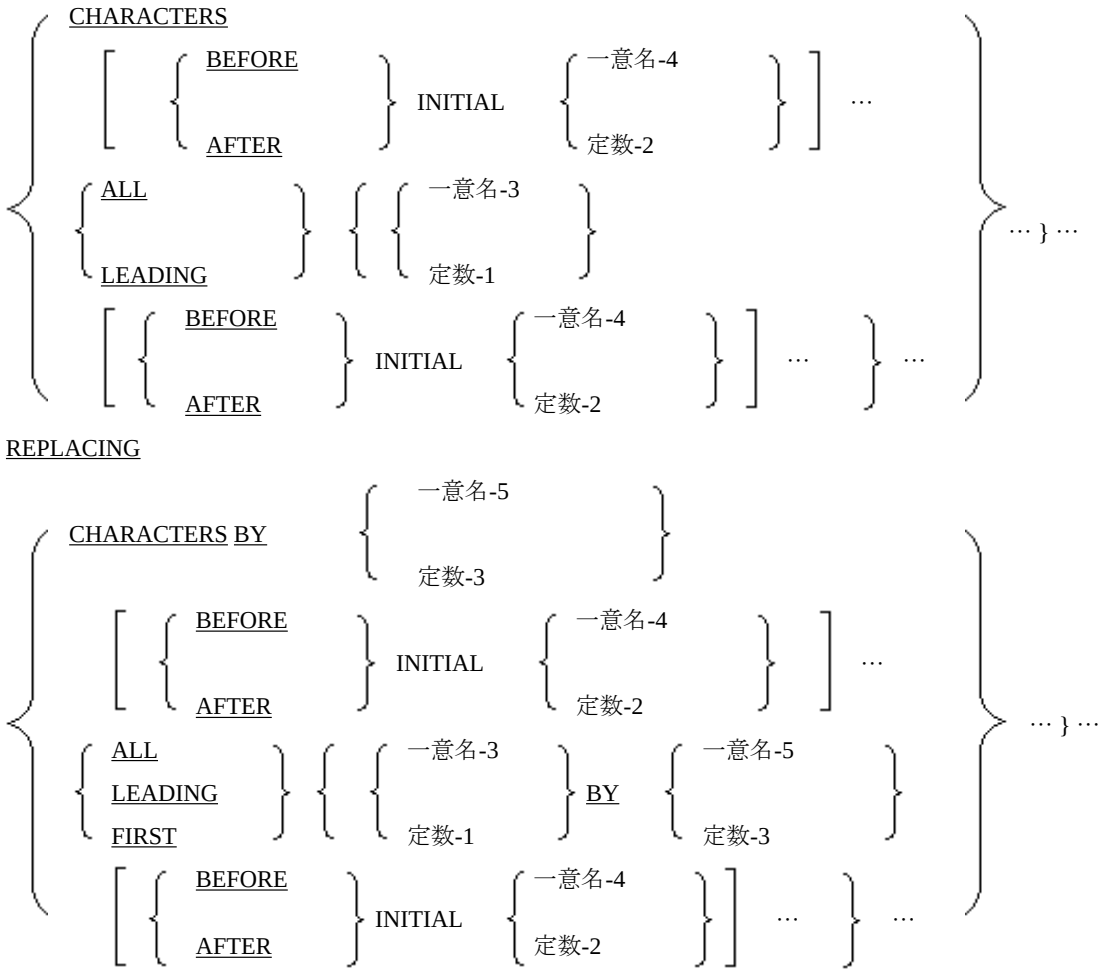


【書き方2】文字列の置き換え

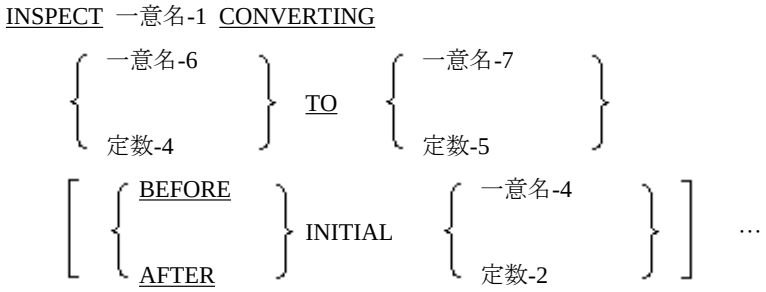


【書き方3】文字列の出現回数のカウントおよび文字列の置き換え

INSPECT 一意名-1 TALLYING {一意名-2 FOR



【書き方4】置き換える文字列をまとめて指定



構文規則

書き方1～書き方4に共通する規則

1. 一意名-1は、集団項目または用途が表示用の基本項目でなければなりません。
2. 一意名-2は、数字項目でなければなりません。
3. 一意名-3～一意名-7は、集団項目または用途が表示用の基本項目でなければなりません。
4. 定数-1～定数-5は、文字定数、日本語定数、またはALLで始まらない表意定数でなければなりません。定数-1、定数-2または定数-4に表意定数を指定した場合、表意定数の長さは1文字であるとみなされます。

5. 1つのALL指定、LEADING指定、FIRST指定、CHARACTERS指定およびCONVERTING指定の後には、1つのBEFORE指定と1つのAFTER指定しか書くことができません。
6. 一意名-3～一意名-7および定数-1～定数-5の組合せは、以下の規則に従わなければなりません。
   a. 一意名-1が日本語項目または日本語編集項目の場合、一意名-3～一意名-7は日本語項目または日本語編集項目でなければなりません。定数-1～定数-5は、日本語定数でなければなりません。
   b. 一意名-1が日本語項目および日本語編集項目以外の場合、一意名-3～一意名-7は日本語項目および日本語編集項目以外でなければなりません。定数-1～定数-5は、日本語定数以外でなければなりません。

書き方2および書き方3に共通する規則

1. CHARACTERS指定の中の定数-3および一意名-5の文字数は、1文字でなければなりません。
2. ALL指定、LEADING指定またはFIRST指定の中の定数-3および一意名-5の文字数は、定数-1または一意名-3の文字数と同じでなければなりません。定数-3に表意定数を指定した場合、表意定数の文字数は、定数-1または一意名-3の文字数と同じであるとみなされます。

書き方4の規則

1. 定数-5および一意名-7の文字数は、定数-4または一意名-6の文字数と同じでなければなりません。定数-5に表意定数を指定した場合、表意定数の文字数は、定数-4または一意名-6の文字数と同じであるとみなされます。
2. 定数-4および一意名-6の文字列の中に、同じ文字を2回以上指定してはいけません。
3. 一意名-6および一意名-7の文字数の最大値については、“付録B システムの定量制限”を参照してください。

一般規則

書き方1の規則

1. 書き方1のINSPECT文は、FOR指定で指定した文字列(検査文字列)が一意名-1に含まれるかどうかを検査し、検査文字列の出現回数を数えます。一意名-1の文字列検査の範囲は、BEFORE指定またはAFTER指定で指定します。検査文字列の出現回数の数え方は、CHARACTERS指定、ALL指定またはLEADING指定で指定します。検査文字列は、一意名-3または定数-1で指定します。CHARACTERS指定の場合は、一意名-1の文字列検査の範囲にある文字の個数を数えます。
2. FOR指定に2つ以上の検査文字列を指定した場合、それらの文字列は、指定した順に検査されます。
3. 一意名-1の中の文字列は左から順に1回だけ検査されます。
4. 1つのALL指定またはLEADING指定の中に2つ以上の検査文字列を書いた場合、ALLまたはLEADINGが各検査文字列に適用されます。
5. INSPECT文の実行前に、一意名-2に初期値を設定しなければなりません。
6. 一意名-1、一意名-3または一意名-4が、一意名-2と同じ記憶領域を占有する場合、INSPECT文の実行結果は規定されません。

書き方1のINSPECT文の処理の流れ

以下のINSPECT文を例に、処理の流れを説明します。

[書き方1のINSPECT文の例]

```
INSPECT AN-1 TALLYING ED-2 FOR ALL "F"
             BEFORE INITIAL "-".
```

[実行前のAN-1の内容]

F	I	F	T	Y	-	F	I	F	T	H
[1]	[2]	[3]	[4]	[5]	[6]					

a. 検査文字列は、“F”です。

b. AN-1の文字列が、最左端から1文字ずつ検査文字列と比較されます。[1]から[6]は、それぞれの比較における最左端の文字位置を示します。[6]の比較のときに、文字列検査の範囲を超えたことがわかり、比較が終了します。

c. [1]および[3]の文字が検査文字列と一致するので、[1]および[3]の比較のときに、ED-2の値が1ずつ増やされます。INSPECT文の実行前のED-2の値が0の場合、INSPECT文の実行後のED-2の値は2になります。

書き方2の規則

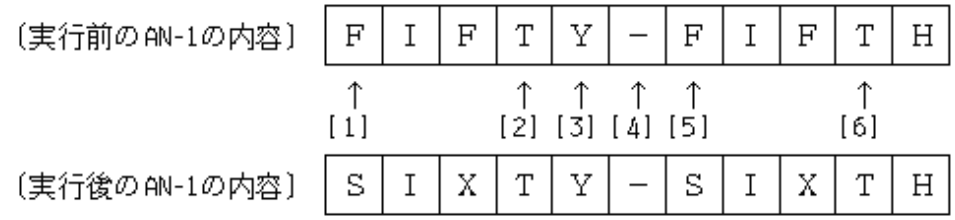
1. 書き方2のINSPECT文は、REPLACING指定で指定した文字列(検査文字列)が一意名-1に含まれるかどうかを検査し、検査文字列と一致した文字列を一意名-5または定数-3で置き換えます。一意名-1の文字列検査の範囲は、BEFORE指定またはAFTER指定で指定します。文字列の置き換え方は、CHARACTERS指定、ALL指定、LEADING指定またはFIRST指定で指定します。検査文字列は、一意名-3または定数-1で指定します。CHARACTERS指定の場合は、一意名-1の文字列検査の範囲にあるすべての文字を置き換えます。
2. REPLACING指定に2つ以上の検査文字列を書いた場合、それらの文字列は、指定した順に検査されます。
3. 一意名-1の文字列は左から順に1回だけ検査されます。
4. 1つのALL指定、LEADING指定またはFIRST指定の中に2つ以上の検査文字列を書いた場合、ALL、LEADINGまたはFIRSTが各検査文字列に適用されます。
5. 一意名-3、一意名-4または一意名-5が、一意名-1と同じ記憶領域を占有する場合、INSPECT文の実行結果は規定されません。

書き方2のINSPECT文の処理の流れ

以下のINSPECT文を例に、処理の流れを説明します。

〔書き方2 のINSPECT 文の例〕

```
    INSPECT AN-1 REPLACING ALL "FIF" BY "SIX".
```



a. 検査文字列は、“FIF”です。

b. AN-1の文字列が、最左端から3文字ずつ検査文字列と比較されます。[1]から[6]は、それぞれの比較における最左端の文字位置を示します。

c. [1]および[5]で始まる3文字が検査文字列と一致するので、[1]および[5]の比較のときに、“FIF”が“SIX”に置き換えられます。

書き方3の規則

1. 書き方3のINSPECT文は、最初にTALLYING指定だけのINSPECT文(書き方1)を書き、次にREPLACING指定だけのINSPECT文(書き方2)を書いたものと同じです。例えば、以下の(a)のINSPECT文の実行結果は、(b)の2つのINSPECT文を順に実行した結果と同じです。

```
INSPECT AN-1 TALLYING ED-2 FOR ALL "F"        … (a)
             BEFORE  INITIAL "-"
             REPLACING ALL "FIF" BY "SIX".
INSPECT AN-1 TALLYING ED-2 FOR ALL "F"        … (b)
             BEFORE  INITIAL "-".
INSPECT AN-1 REPLACING ALL "FIF" BY "SIX".
```

2. 出現回数の計算のための規則は、書き方1の規則と同じであり、置換えのための規則は、書き方2の規則と同じです。

書き方4の規則

1. 書き方4のINSPECT文は、REPLACINGの後にALL指定だけを2つ以上書いた書き方2のINSPECT文と同じです。書き方4の作用対象と書き方2の作用対象は、それぞれ以下のように対応します。

   ー 書き方4の一意名-1:書き方2の一意名-1

   ー 書き方4の一意名-6または定数-4の1文字ずつ:書き方2のALL指定の一意名-3または定数-1

   ー 書き方4の一意名-7または定数-5の1文字ずつ:書き方2のALL指定の一意名-5または定数-3

   ー 書き方4の一意名-4または定数-2:書き方2のALL指定の一意名-4または定数-2

   例えば、以下の(a)のINSPECT文の実行結果は、(b)のINSPECT文の実行結果と同じです。

```
INSPECT ITEM CONVERTING                          …  (a)
    "ABCD" TO "XYZX" AFTER QUOTE BEFORE "@".
INSPECT ITEM REPLACING                           …  (b)
    ALL "A" BY "X" AFTER QUOTE BEFORE "@"
    ALL "B" BY "Y" AFTER QUOTE BEFORE "@"
    ALL "C" BY "Z" AFTER QUOTE BEFORE "@"
    ALL "D" BY "X" AFTER QUOTE BEFORE "@".
```

2. 一意名-4、一意名-6または一意名-7が、一意名-1と同じ記憶領域を占有する場合、INSPECT文の実行結果は保証されません。

書き方1～書き方4に共通する規則

各一意名に関する規則

一意名-2以外の一意名は、以下の内容であるとみなされます。

1. 一意名に英数字項目、英字項目または日本語項目を指定した場合、一意名の内容は文字列とみなされます。
2. 一意名に英数字編集項目、数字編集項目、符号なし数字項目または外部ブール項目を指定した場合、一意名の内容は、英数字項目によって再定義された文字列とみなされます。
3. 一意名に日本語編集項目を指定した場合、一意名の内容は、日本語項目によって再定義された文字列とみなされます。
4. 一意名に符号付き数字項目を指定した場合、一意名の内容は、同じ長さの符号なし数字項目に転記され、文字列とみなされます。一意名の符号は、INSPECT文が完了するまで保存されます。

添字および部分参照子の評価に関する規則

一意名-1～一意名-7に添字または部分参照子を付けた場合、または一意名-2以外の一意名に関数一意名を指定した場合、添字、部分参照子および関数一意名は、INSPECT文の実行の最初に1回だけ評価されます。

文字列検査の範囲

1. BEFORE指定とAFTER指定の両方を省略した場合、一意名-1のすべての文字列が、文字列検査の範囲になります。
2. BEFORE指定を書いた場合、一意名-1の文字列のうち、最左端文字位置から始まり、一意名-4または定数-2と最初に一致する文字列の直前までの文字列が、文字列検査の範囲になります。文字列検査の範囲の最右端文字の決定は、検査文字列との比較が行われる前に行われます。一意名-1の文字列中に一意名-4または定数-2と一致する文字が出現しなかった場合、BEFORE指定を書かなかったものとみなされます。
3. AFTER指定を書いた場合、一意名-1の文字列のうち、一意名-4または定数-2と最初に一致する文字の直後の文字列から始まり、最右端文字までが、文字列検査の範囲になります。文字列検査の範囲の最左端文字の決定は、検査文字列との比較が行われる前に行われます。一意名-1の文字列中に一意名-4または定数-2と一致する文字が出現しなかった場合、検査文字列との比較は行われません。

検査文字列の比較の手順

1. 検査文字列の比較は、以下の手順で行われます。以下で、最初の「現在の最左端位置」は、一意名-1の文字列検査の範囲の最左端の文字位置です。

   a. 「現在の最左端位置」から始まる文字列が、検査文字列と同じ文字数だけ、検査文字列と比較されます。

b. a.の比較で文字列が一致した場合、後述の「出現回数の数え方と文字列の置き換え方」の規則に従って、出現回数の加算と文字列の置換えが行われます。そして、「現在の最左端文字位置」が、比較された文字列の最右端文字のすぐ右側の文字位置に移ります。

c. a.の比較で文字列が一致しなかった場合、検査文字列を2つ以上指定したときは、それらを指定した順に、「現在の最左端位置」から始まる文字列が検査文字列と比較されます。この比較は、文字列が一致するかまたは検査文字列がなくなるまで繰り返されます。文字列が一致した場合は、b.と同じ処理が行われます。検査文字列がなくなった場合は、“現在の最左端文字位置”が、比較された文字列の最右端文字のすぐ右側の文字位置に移ります。

d. 「現在の最左端位置」が一意名-1の文字列検査の範囲を超えるまで、a.～c.の処理が繰り返されます。

2. CHARACTERS指定の場合、検査文字列として暗黙の1文字が仮定され、検査文字列の比較では、すべての文字が検査文字列と常に一致するものとみなされます。

出現回数の数え方と文字列の置き換え方

1. ALL指定の場合、文字列検査の範囲にある、検査文字列と一致するすべての文字列が、出現回数の加算と文字列の置換えの対象になります。
2. LEADING指定の場合、文字列検査の範囲の最左端位置から始まる文字列が検査文字列と一致した場合だけ、出現回数の加算と文字列の置換えが行われます。文字列検査の範囲の最左端位置から始まり、検査文字列と異なる文字列が現れるまでの文字列が、出現回数の加算と文字列の置換えの対象になります。
3. FIRST指定の場合、文字列検査の範囲にある、最初に検査文字列と一致した文字列だけが、出現回数の加算と文字列の置換えの対象になります。
4. CHARACTERS指定の場合、文字列検査の範囲にあるすべての文字列が、出現回数の加算と文字列の置換えの対象になります。

例

INSPECT文の例(その1)

```
INSPECT ITEM TALLYING                 …  (a)
    COUNT-0 FOR ALL "AB", ALL "D"
    COUNT-1 FOR ALL "BC"
    COUNT-2 FOR LEADING "EF"
    COUNT-3 FOR LEADING "B"
    COUNT-4 FOR CHARACTERS.
INSPECT ITEM REPLACING                …  (b)
    ALL "AB" BY "XY", "D" BY "X"
    ALL "BC" BY "VW"
    LEADING "EF" BY "TU"
    LEADING "B" BY "S"
    FIRST "G" BY "R"
    FIRST "G" BY "P"
    CHARACTERS BY "Z".
```

(a)および(b)のINSPECT文の実行結果を下表に示します。

ITEMの初期値	(a)の実行結果 (*1)					(b)の実行結果
	COUNT-0	COUNT-1	COUNT-2	COUNT-3	COUNT-4	ITEMの終了値
EFABDBCGABCFGG	3	1	1	0	5	TUXYXVWRXYZZPZ
BABABC	2	0	0	1	1	SXYXYZ
BBBC	0	1	0	2	0	SSVW

*1 ： COUNT-0 ～COUNT-4 の初期値はゼロであるとします。

例

INSPECT文の例(その2)

```
INSPECT ITEM TALLYING          … (a)
    COUNT-0 FOR CHARACTERS
    COUNT-1 FOR ALL "A".
INSPECT ITEM REPLACING         … (b)
    CHARACTERS BY "Z"
    ALL "A" BY "X".
```

(a)および(b)のINSPECT文の実行結果を下表に示します。

ITEMの初期値	(a)の実行結果 (*1)		(b)の実行結果
	COUNT-0	COUNT-1	ITEMの終了値
BBB	3	0	ZZZ
ABA	3	0	ZZZ

*1 ： COUNT-0 およびCOUNT-1 の初期値はゼロであるとします。

例

INSPECT文の例(その3)

```
INSPECT ITEM TALLYING                          … (a)
    COUNT-0 FOR ALL "AB" BEFORE "BC"
    COUNT-1 FOR LEADING "B" AFTER "D"
    COUNT-2 FOR CHARACTERS AFTER "A" BEFORE "C".
INSPECT ITEM REPLACING                         … (b)
    ALL "AB" BY "XY" BEFORE "BC"
    LEADING "B" BY "W" AFTER "D"
    FIRST "E" BY "V" AFTER "D"
    CHARACTERS BY "Z" AFTER "A" BEFORE "C".
```

(a)および(b)のINSPECT文の実行結果を下表に示します。

ITEMの初期値	(a)の実行結果 (*1)			(b)の実行結果
	COUNT-0	COUNT-1	COUNT-2	ITEMの終了値
BBEABDABABBCABEE	3	0	2	BBEXYZXYXYZCABVE
ADDDDC	0	0	4	AZZZZC
ADDDDA	0	0	5	AZZZZZ
CDDDDC	0	0	0	CDDDDC
BDBBBDB	0	3	0	BDWWWDB

*1 ： COUNT-0 ～COUNT-2 の初期値はゼロであるとします。

例

INSPECT文の例(その4)

```
INSPECT ITEM TALLYING                          … (a)
    COUNT-0 FOR ALL "AB" AFTER "BA" BEFORE "BC".
```

```
    INSPECT ITEM REPLACING                                  …  (b)
       ALL "AB" BY "XY" AFTER "BA" BEFORE "BC".
```

(a)および(b)のINSPECT文の実行結果を下表に示します。

ITEMの初期値	(a)の実行結果 (*1)	(b)の実行結果
	COUNT-0	ITEMの終了値
ABABABABC	1	ABABXYABC

*1 ： COUNT-0 の初期値はゼロであるとします。

例

INSPECT文の例(その5)

```
    INSPECT ITEM CONVERTING
       "ABCD" TO "XYZX" AFTER QUOTE BEFORE "@".
```

このINSPECT文の実行結果を下表に示します。

ITEMの初期値	ITEMの終了値
AC"AEBDFBCD@AB"D	AC"XEYXFYZX@AB"D

6.4.27 MERGE文（整列併合）

2つ以上のファイルを併合します。

【書き方】

MERGE ファイル名-1
{ON { ASCENDING / DESCENDING } KEY {データ名-1} … } …
[COLLATING SEQUENCE IS 符号系名-1]
USING ファイル名-2 {ファイル名-3} …
{ OUTPUT PROCEDURE IS 手続き名-1 [{ THROUGH / THRU } 手続き名-2] / GIVING {ファイル名-4} … }

構文規則

1. MERGE文は、宣言部分を除く手続き部のどこでも書くことができます。
2. ファイル名-1は、データ部の整列併合用ファイル記述項で定義しなければなりません。
3. ファイル名-2およびファイル名-3は、以下の規則に従わなければなりません。
   - ファイル名-1のレコード形式が可変長の場合、ファイル名-2およびファイル名-3のレコードの長さは、ファイル名-1の最小レコード長以上かつ最大レコード長以下でなければなりません。

- ファイル名-1のレコード形式が固定長の場合、ファイル名-2およびファイル名-3のレコードの長さは、ファイル名-1のレコード長以下でなければなりません。

4. データ名-1は、以下の規則に従わなければなりません。
   a. データ名-1は、ファイル名-1のレコード記述項で定義したデータ項目でなければなりません。
   b. データ名-1は、修飾することができます。
   c. データ名-1に、可変反復データ項目を従属する集団項目を指定してはいけません。
   d. ファイル名-1に2つ以上のレコード記述項を書いた場合、データ名-1の並びには、1つのレコード記述項に書いたデータ項目を指定しなければなりません。1つのレコード記述項のデータ名-1のデータ項目と同じ文字位置が、そのファイルのすべてのレコード中のキーとみなされます。
   e. データ名-1に、OCCURS句を指定したデータ項目またはOCCURS句を指定したデータ項目に従属するデータ項目を指定することはできません。
   f. ファイル名-1のレコード形式が可変長の場合、データ名-1のデータ項目は、ファイル名-1の最小レコードの大きさの範囲内に含まれていなければなりません。
   g. データ名-1は、ブール項目であってはいけません。
   h. データ名-1の個数の最大値については、“付録B システムの定量制限”を参照してください。
5. ファイル名-2、ファイル名-3およびファイル名-4は、データ部中の整列併合用ファイル記述項以外のファイル記述項で定義しなければなりません。
6. 1つのMERGE文で指定した2つのファイルが、同一の複数ファイルリール上にあってはいけません。
7. ファイル名-1～ファイル名-4に、同じファイルを指定することはできません。
8. ファイル名-1～ファイル名-4のいくつかの組を、1つのSAME AREA句、または1つのSAME SORT/SORT-MERGE AREA句に指定することはできません。しかし、1つのSAME RECORD AREA句には、ファイル名-4のいくつかの組を指定することができます。
9. THROUGHとTHRUは同義語です。
10. ファイル名-4に索引ファイルを指定する場合、以下の規則に従わなければなりません。
    - 最初のKEY指定は、ASCENDING KEY指定でなければなりません。
    - 最初のデータ名-1に指定したデータ項目のレコード内文字位置は、索引ファイルの主レコードキーに関連付けたデータ項目のレコード内文字位置と同じでなければなりません。また、2つのデータ項目は同じ長さでなければなりません。
11. ファイル名-4は、以下の規則に従わなければなりません。
    - ファイル名-4のレコード形式が可変長の場合、ファイル名-1のレコードの長さは、ファイル名-4の最小レコード長以上かつ最大レコード長以下でなければなりません。
    - ファイル名-4のレコード形式が固定長の場合、ファイル名-1のレコードの長さは、ファイル名-4のレコード長以下でなければなりません。
12. 符号系名-1は、特殊名段落のALPHABET句で機能名に対応付けた符号系名であってはいけません。
13. ファイル名-1～ファイル名-4のレコード記述項に、CHARACTER TYPE句またはPRINTING POSITION句を書くことはできません。
14. ファイル名-2～ファイル名-4は、以下のファイルでなければなりません。
    - 大記憶装置またはフロッピーディスク装置の順ファイル
    - 順呼出し法の相対ファイル
    - 順呼出し法の索引ファイル
15. ファイル名-2およびファイル名-3に指定できるファイルの個数の最大値は、“付録B システムの定量制限”を参照してください。

一般規則

MERGE文の全般規則

1. ファイル名-2およびファイル名-3のレコードは、ASCENDING KEY指定またはDESCENDING KEY指定に従って並んでいなければなりません。その順序で並んでいない場合、MERGE文の結果は規定されません。

2. MERGE文を実行すると、以下の処理が行われます。

   a. ファイル名-2およびファイル名-3に対してREAD文が実行され、READ文によって読み込まれたレコードが、併合操作に引き渡されます。この段階を始める前に、ファイル名-2およびファイル名-3のファイルは開いた状態であってはいけません。この段階の中で、ファイル名-2およびファイル名-3のファイルは自動的にオープンされ、クローズされます。ここでの処理を、「併合操作の最初の段階」といいます。

   b. ファイル名-2およびファイル名-3の中のすべてのレコードが併合されます。ここでの処理を、「併合操作」といいます。

   c. 以下の処理によって、併合されたレコードがファイル名-1のレコードとして使用可能になります。ここでの処理を、「併合操作の最後の段階」といいます。

      OUTPUT PROCEDURE指定を書いた場合、出力手続きが実行されます。「出力手続き」とは、手続き名-1に書いた手続き、または手続き名-1から手続き名-2までに書いた手続きのことです。出力手続きの中でファイル名-1のファイルに対するRETURN文を実行すると、ファイル名-1のレコードが使用可能になります。

      GIVING指定を書いた場合、併合されたレコードがファイル名-1のレコードとして使用可能になり、ファイル名-4のファイルに書き出されます。この段階を始める前に、ファイル名-4のファイルは、開いた状態であってはいけません。この段階の中で、ファイル名-4のファイルは自動的にオープンされ、クローズされます。

3. 併合操作の最初の段階では、以下の処理が順に行われます。

   a. 初期処理が行われます。これは、ファイル名-2およびファイル名-3のファイルに対して、INPUT指定のOPEN文を実行したことと同じです。OUTPUT PROCEDURE指定を書いた場合、初期処理は、出力手続きに制御が移る前に行われます。

   b. ファイル名-2およびファイル名-3のレコードが、併合操作に引き渡されます。これは、ファイル名-2およびファイル名-3に対して、NEXT指定およびAT END指定のREAD文を、レコードがなくなるまで実行したことと同じです。

   c. 終了処理が行われます。これは、ファイル名-2またはファイル名-3だけを指定したCLOSE文を実行したことと同じです。OUTPUT PROCEDURE指定を書いた場合、終了処理は、出力手続きの最後の文に制御が渡った後まで行われません。

   上記のa.～c.の処理の中で、ファイル名-2～ファイル名-4のファイルに関連するUSE AFTER STANDARD EXCEPTION手続きが実行されることがあります。そのUSE手続きの中では、ファイル名-2～ファイル名-4のファイルを操作する文、またはそれらのファイル中のレコードを参照する文を実行することはできません。

4. ファイル名-1のレコード形式が固定長で、ファイル名-2およびファイル名-3のレコードがファイル名-1のレコード長より短い場合、そのレコードがファイル名-1のファイルに引き渡されるとき、レコードの右側の余りの部分には空白が詰められます。

5. データ名-1のデータ項目を、「キー項目」といいます。キー項目を2つ以上指定する場合、キーの強さの順に左から右へ並べます。

   － ASCENDING指定を書いた場合

      ファイル名-2およびファイル名-3のレコードのキー項目の値が比較の規則に従って比較され、キー項目の値が小さいレコードから大きいレコードの順に併合されます。

   － DESCENDING指定を書いた場合

      ファイル名-2およびファイル名-3のレコードのキー項目の値が比較の規則に従って比較され、キー項目の値が大きいレコードから小さいレコードの順に併合されます。

6. 5.の比較の結果、すべてのキー項目の内容が同じであるレコードが2つ以上存在する場合には、ファイル名-2およびファイル名-3の並びを書いた順に各ファイル中のレコードが引き取られます。1つのファイル名-2またはファイル名-3のファイル中にそのような複数のレコードが存在する場合、他のファイルからレコードが引き取られる前にレコードが引き取られます。

7. 文字比較の規則が適用される場合、キー項目の値の大小順序は、以下の規則に従って決定されます。

   ー COLLATING SEQUENCE指定を書いた場合

   符号系名-1に対応付けた文字の大小順序に従います。

   ー COLLATING SEQUENCE指定を省略した場合

   見出し部の翻訳用計算機段落にPROGRAM COLLATING SEQUENCE句を書いた場合、その句で指定した文字の大小順序に従います。この句を省略した場合、計算機固有の文字の大小順序に従います。

OUTPUT PROCEDURE指定の規則

1. OUTPUT PROCEDURE指定を書いた場合、併合操作が終了した後、出力手続きに制御が移ります。出力手続きの最後の文を実行した後、MERGE文の次の実行文に制御が移ります。
2. 出力手続きに制御が移ったとき、ファイル名-1のレコードは完全に併合されていて、RETURN文によって順番に引き取れるような状態になっています。出力手続きの中には、ファイル名-1のファイルに対するRETURN文を書きます。RETURN文は、併合操作の最後の段階から、1つのレコードを引き取ります。
3. 出力手続きの中で実行される文を、出力手続きの範囲といいます。出力手続きの範囲は、以下のとおりです。

   ー 出力手続きの中に書いた文

   ー 出力手続きの中に書いた文(たとえばCALL文など)によって制御が移行した結果、実行されるすべての文

   ー 出力手続きの中に書いた文によって実行される宣言手続きの中のすべての文
4. 出力手続きの範囲で、MERGE文、RELEASE文またはSORT文を実行することはできません。
5. 出力手続きの範囲で、MERGE文と同じ手続き部内に書かれたEXIT PROGRAM文を実行することはできません。

GIVING指定の規則

1. GIVING指定を書いた場合、併合操作が終了した後、ファイル名-1のすべてのレコードが、ファイル名-4のファイルに移されます。
2. MERGE文を実行する前に、ファイル名-4のファイルは開かれた状態であってはいけません。
3. ファイル名-4の各ファイルに対して、以下の処理が順に行われます。

   a. 初期処理が行われます。これは、ファイル名-4のファイルに対して、OUTPUT指定のOPEN文を実行したことと同じです。

   b. ファイル-1のレコードを併合操作から引き取ります。これは、ファイル名-1のファイルに対して、レコード名だけを指定したWRITE文を、ファイル名-1のレコードがなくなるまで実行したことと同じです。

      ファイル名-4に相対ファイルを指定した場合、引き取られる相対レコード番号は、最初のレコードから順に、1、2、3、…というように設定されます。ファイル名-4のファイルに相対キー項目を指定した場合、レコードが引き取られるごとに、相対レコード番号が相対キー項目に設定されます。MERGE文の実行後の相対キー項目の内容は、ファイル名-4のファイルに引き取られた最後のレコードを表します。

   c. 終了処理が行われます。これは、ファイル名-4だけを指定したCLOSE文を実行したことと同じです。

   上記のa.～c.の処理の中で、ファイル名-4に関連するUSE AFTER STANDARD EXCEPTION手続きが実行されることがあります。そのUSE手続きの中では、ファイル名-4のファイルを操作する文またはファイル名-4のレコードを参照する文を実行することはできません。
4. ファイル名-4のファイルの外部で定義された区域を超えて、最初にレコードを書き出そうとすると、以下の処理が行われます。

   ー ファイル名-4に関連するUSE AFTER STANDARD EXCEPTION手続きを書いた場合

   そのUSE手続きが実行されます。そして、USE手続きから制御が戻った後、ファイル名-4だけを指定したCLOSE文を実行したかのように、終了処理が行われます。

   ー ファイル名-4に関連するUSE AFTER STANDARD EXCEPTION手続きを書かなかった場合

   ファイル名-4だけを指定したCLOSE文を実行したかのように、終了処理が行われます。

5. ファイル名-4のレコード形式が固定長で、ファイル名-1のレコードの長さがファイル名-4のレコード長より短い場合、そのレコードがファイル名-4のファイルに引き取られるとき、レコードの右側の余りの部分には空白が詰められます。

6.4.28 MOVE文（中核）

データを別のデータ項目に転記します。

使い方の例については、“付録H サンプル集”の“H.39 MOVE文”を参照してください。

【書き方1】1つのデータを1つ以上のデータ項目に転記する

```
MOVE { 一意名-1 } TO {一意名-2} …
     { 定数-1   }
```

【書き方2】2つの集団項目に従属するデータ項目の対応をとって転記する

```
MOVE { CORRESPONDING } 一意名-1 TO 一意名-2
     { CORR          }
```

構文規則

1. CORRESPONDINGとCORRは同義語です。
2. 書き方2の一意名-1および一意名-2は、集団項目でなければなりません。
3. 一意名-1および一意名-2は、指標データ項目であってはいけません。
4. 一意名-2が強く型付けされた集団項目である場合、定数-1は指定できません。
5. 一意名-2が強く型付けされた集団項目である場合、一意名-1は同じ型で強く型付けされた集団項目でなければなりません。

一般規則

書き方1の規則

1. 書き方1のMOVE文は、TOの前の作用対象を、TOの後の各作用対象に転記します。TOの前の作用対象を「送出し側作用対象」、TOの後の各作用対象を「受取り側作用対象」といいます。
2. 一意名-1に部分参照子または添字を付けた場合、または一意名-1に関数一意名を指定した場合、部分参照子、添字および関数一意名は、送出し側作用対象を先頭の受取り側作用対象に転記する直前に1回だけ評価されます。
3. 一意名-2に部分参照子または添字を付けた場合、部分参照子および添字はそれぞれの受取り側作用対象への転記を行う直前に評価されます。
4. 一意名-1のデータ項目の長さは、送出し側作用対象を受取り側作用対象に転記する直前に1回だけ評価されます。
5. 一意名-2のデータ項目の長さは、それぞれの受取り側作用対象への転記を行う直前に評価されます。

書き方2の規則

書き方2のMOVE文は、一意名-1に従属するデータ項目と一意名-2に従属するデータ項目のうち、名前の修飾語の系列が同じものどうしを転記します。一意名-1に従属するデータ項目を送出し側作用対象、一意名-2に従属するデータ項目を受取り側作用対象として転記します。この結果は、対応する一意名ごとに、別々のMOVE文を書いた結果と同じです。

書き方1および書き方2に共通する規則

1. 一意名-1または一意名-2に可変反復データ項目を指定した場合、可変反復データ項目の長さの評価はDEPENDING ON指定のデータ項目の値によって影響されます。可変反復データ項目の長さの評価については、“5.4.6 OCCURS句”を参照してください。
2. 送出し側作用対象と受取り側作用対象の組合せ、およびMOVE文の動作については、“6.3.5 転記の規則”を参照してください。

例

添字の評価の例

```
MOVE a (b) TO b c (b)
```

このMOVE文の実行結果は、以下のMOVE文の実行結果と同じです。tempは一時的なデータ項目を示します。

```
MOVE a (b) TO temp
MOVE temp TO b
MOVE temp TO c (b)
```

6.4.29 MULTIPLY文（中核）

乗算の結果を求めます。

【書き方1】乗算の結果を乗数と置き換える

MULTIPLY { 一意名-1 / 定数-1 } BY {一意名-2 ROUNDED]} …
[ON SIZE ERROR 無条件文-1]
[NOT ON SIZE ERROR 無条件文-2]
[END-MULTIPLY]

【書き方2】乗算の結果を乗数とは異なるデータ項目に格納する

MULTIPLY { 一意名-1 / 定数-1 } BY { 一意名-2 / 定数-2 }
GIVING {一意名-3 [ROUNDED]} …
[ON SIZE ERROR 無条件文-1]
[NOT ON SIZE ERROR 無条件文-2]
[END-MULTIPLY]

構文規則

1. 一意名-1および一意名-2は、数字項目でなければなりません。
2. 一意名-3は、数字項目または数字編集項目でなければなりません。
3. 定数-1および定数-2は、数字定数でなければなりません。

一般規則

書き方1の規則

書き方1のMULTIPLY文は、BYの前の作用対象に一意名-2を掛け、一意名-2に格納します。一意名-2の並びを書いた順に、この乗算を行います。

書き方2の規則

書き方2のMULTIPLY文は、BYの前の作用対象にBYの後の作用対象を掛け、一意名-3に格納します。一意名-3の並びを書いた順に、この乗算の結果を格納します。

書き方1および書き方2に共通する規則

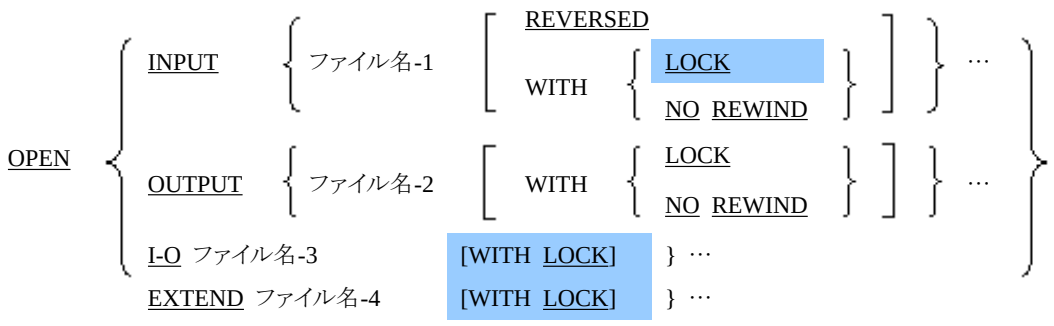
1. END-MULTIPLY指定は、MULTIPLY文の範囲を区切ります。
2. ROUNDED指定、ON SIZE ERROR指定、演算および転記の規則については、“6.3 文に関する共通の規則”を参照してください。

6.4.30 OPEN文（順ファイル・相対ファイル・索引ファイル）

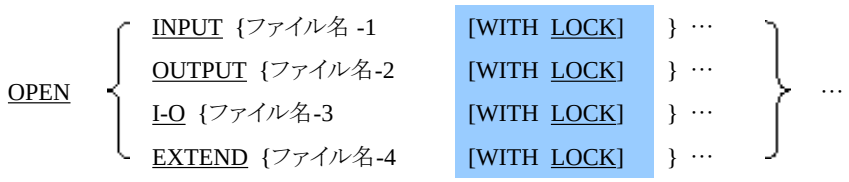
ファイルを使用可能な状態にします。

順ファイルのOPEN文のREVERSED指定は、廃要素です。

【書き方1】順ファイル



【書き方2】相対ファイル・索引ファイル



構文規則

各ファイルに共通する規則

ファイル名-1～ファイル名-4の並びに、ORGANIZATION句の指定が異なるファイルまたはACCESS MODE句の指定が異なるファイルを指定することもできます。

順ファイルの規則

1. REVERSED指定またはNO REWIND指定を書く場合、ファイル名-1はレコード順ファイルでなければなりません。
2. NO REWIND指定は注釈とみなされます。

3. EXTEND指定を書く場合、ファイル名-4は、LINAGE句の指定のないファイルでなければなりません。ただし、EXTEND指定を、複数ファイルリールに書くことはできません。
4. INPUT指定を書く場合、ファイル名-1は印刷ファイルであってはいけません。
5. I-O指定を書く場合、ファイル名-3は行順ファイル、または印刷ファイルであってはいけません。

相対ファイルおよび索引ファイルの規則

EXTEND指定を書く場合、ファイル名-4は順呼出し法のファイルでなければなりません。

一般規則

各ファイルに共通する規則

以下で、“ファイル名-n”は、ファイル名-1～ファイル名-4に指定したファイルを表します。

1. OPEN文の実行が成功すると、ファイル名-nのファイルはプログラムで使用可能になり、開いた状態になります。OPEN文の実行が成功すると、そのファイルは、ファイル結合子を通してファイル名と関連付けられます。
2. ファイル名-nのファイルが、物理的に存在し入出力管理システムに認識された状態を「ファイルが使用可能」といいます。認識されていない状態を、「ファイルが使用不可能」といいます。それぞれの状態でOPEN文を実行したときの結果を、下表に示します。

OPEN文の指定	ファイルが使用可能	ファイルが使用不可能
INPUT	ファイルが開かれます。	OPEN文は不成功になります。
INPUT （不定ファイルの場合）	ファイルが開かれます。	ファイルが開かれます。最初の読込みでファイル終了条件または無効キー条件になります。
I-O	ファイルが開かれます。	OPEN文は不成功になります。
I-O （不定ファイルの場合）	ファイルが開かれます。	OPEN文の実行で、ファイルが生成されます。
OUTPUT	ファイルが開かれます。ただし、そのファイルにはレコードがない状態です。	OPEN文の実行で、ファイルが生成されます。
EXTEND	ファイルが開かれます。	OPEN文は不成功になります。
EXTEND （不定ファイルの場合）	ファイルが開かれます。	OPEN文の実行で、ファイルが生成されます。

3. OPEN文の実行が成功すると、ファイル名-nのレコード領域が使用可能になります。
4. ファイルを開く前に、そのファイルに対する入出力文を実行することはできません。
5. INPUT指定、OUTPUT指定、I-O指定およびEXTEND指定は、ファイルのオープンモードを指定します。ファイルを開いた後に実行できる入出力文は、オープンモードによって異なります。オープンモードと入出力文の関係については、“2.2.3 入出力文の動作”を参照してください。
6. OPEN文は、最初のレコードの読み書きを行いません。
7. 1つの実行単位中で、1つのファイルに対するOPEN文を2回以上実行する場合、次のOPEN文を実行する前に、REEL指定、UNIT指定(順ファイルだけ)およびLOCK指定のいずれの指定もないCLOSE文を実行しなければなりません。
8. OPEN文の実行中にファイル属性不整合条件が発生すると、OPEN文の実行は不成功になり、以下の処理が順に行われます。
    a. ファイル名-nの入出力状態が設定されます。
    b. ファイル名-nに関連するUSE AFTER STANDARD EXCEPTION手続きを書いた場合、その手続きが実行されます。そして、その手続きの実行後、USE文の規則に従って制御が渡されます。USE AFTER STANDARD EXCEPTION手続きを書かなかった場合、ファイル名-nにFILE STATUS句を指定したときは、OPEN文の最

後に制御が移ります。USE AFTER STANDARD EXCEPTION手続きもFILE STATUS句も書かなかったときの実行結果は、規定されません。

9. OPEN文を実行すると、ファイル名-nの入出力状態の値が更新されます。
10. I-O指定を書く場合、ファイル名-3のファイルは、入力文と出力文の両方を実行することができるファイルでなければなりません。I-O指定のOPEN文の実行が成功すると、入力文と出力文の両方の実行が可能になります。
11. OCK指定は、ファイルのロックのモードを指定します。ファイルのロックのモードは、ファイル管理記述項のLOCK MODE句とOPEN文の記述によって決まります。ファイルのロックのモードについては、“2.2.6 ファイルの共用と排他”を参照してください。
12. LOCK指定を書いた場合、OPEN文の実行によって、ファイル名-nのファイルに対するロックが取得されます。これは、ファイル名-nのLOCK MODE句でEXCLUSIVEを書いたことと同じです。
13. 排他モードのOPEN文の実行が成功すると、ファイル名-nのファイル結合子に関連するファイルが、排他的にロックされます。ファイル名-nのファイルがすでに他のファイル結合子によって開かれている場合は、OPEN文の実行は不成功になります。
14. 共用モードのOPEN文の実行が成功した後、ファイル名-nのファイルを1つ以上のファイル結合子によって開くことができます。ファイル名-nのファイルがすでに他のファイル結合子によって排他的に開かれている場合は、OPEN文の実行は不成功になります。
15. 不定ファイルに対して、実行単位中で最初にINPUT指定のOPEN文を実行すると、ファイル位置指示子は不定ファイルが存在しないことを指示するように設定されます。
16. まだ生成されていない不定ファイルに対するEXTEND指定またはI-O指定のOPEN文の実行が成功すると、ファイルが生成されます。ファイルの生成は、以下の順に2つの文を実行したかのように行われます。

    ― OPEN OUTPUT ファイル名-n

    ― CLOSE ファイル名-n

    生成されたファイルは、レコードが存在しない状態になっています。
17. ファイル名-nを2つ以上書いた場合のOPEN文の実行結果は、書いた順に、各ファイルに対して別々のOPEN文を実行した結果と同じです。
18. ファイルの最小のレコードの大きさおよび最大のレコードの大きさは、利用者の指定に従って、ファイルを生成したときに決定されます。その後は変更できません。

順ファイルの規則

1. INPUT指定またはI-O指定のOPEN文の実行が成功すると、ファイル名-nのファイル位置指示子に1が設定されます。
2. EXTEND指定のOPEN文の実行が成功すると、ファイルはその最後のレコードの直後に位置付けられます。順ファイルの最後のレコードは、そのファイルで最後に書き出されたレコードです。
3. ファイル名-nに、ラベルレコードの指定のあるレコード順ファイルを指定した場合、始めラベルは以下のように処理されます。

    a. OPEN INPUT文を実行すると、標準のラベル規約に従ってラベルの検査が行われます。

    b. OPEN OUTPUT文を実行すると、標準のラベル規約に従ってラベルの書出しが行われます。

    ラベルレコードの指定があってラベルがない場合、またはラベルレコードの指定がなくてラベルがある場合、OPEN文の実行結果は規定されません。
4. EXTEND指定のOPEN文を、ラベルレコードの指定のあるレコード順ファイルに対して実行した場合、ラベルは以下の手順で処理されます。

    a. 順単一リール/ユニットファイルの場合に限って、始めファイルラベルが処理されます。

    b. 順複数リール/ユニットファイルの場合、標準のラベル規約に従って最後のリール/ユニットの始めボリュームラベルの検査が行われます。

    c. 標準のラベル規約に従って終わりファイルラベルの検査が行われます。その後、このラベルは削除されます。

d. 続いて、標準のラベル規約に従ってラベルの書出しが行われます。

5. I-O指定のOPEN文を、ラベルレコードの指定のあるレコード順ファイルに対して実行した場合、ラベルは以下の手順で処理されます。

   a. 標準のラベル規約に従ってラベルの検査が行われます。

   b. 標準のラベル規約に従ってラベルの書出しが行われます。

6. 複数ファイルテープ環境下のファイルは、単一ファイルテープ環境下の順ファイルと同じものとみなします。

7. ファイル名-nに、複数ファイルリール上にあるファイルを指定する場合、以下の規則が適用されます。

   a. 同一テープリール上のファイルを、同時に2つ以上開いてはいけません。

   b. INPUT指定のOPEN文を実行する場合、ファイル名-nの位置番号は、それ以前に開いたファイルの位置番号以上である必要はありません。

   c. OUTPUT指定のOPEN文を実行する場合、ファイル名-nは、関連するテープリール上に存在するファイルの中で最大の位置番号を持たなければなりません。

   d. ファイル名-nは、順ファイルでなければなりません。

8. NO REWIND指定またはREVERSED指定を書いたOPEN文は、以下のファイルに対してだけ実行できます。

   a. 順単一リール/ユニットファイル。

   b. 1つのテープリール内に完全に含まれる、複数ファイルテープ環境下の順ファイル。複数ファイルテープ環境については、“4.3.2.3 MULTIPLE FILE TAPE句（順ファイル・報告書作成）”を参照してください。

9. NO REWIND指定またはREVERSED指定を書いた場合、ファイル名-nのファイルがこれらの機能を適用できないファイル(例えば大記憶ファイル)のときは、これらの指定は無視されます。

10. ファイル名-nのファイルに巻き戻し可能な装置を割り当てた場合、以下の規則が適用されます。

    a. REVERSED指定、EXTEND指定またはNO REWIND指定のいずれも書かなかった場合、OPEN文を実行すると、ファイルは先頭に位置付けられます。

    b. NO REWIND指定を書いた場合、OPEN文を実行する前にファイルの始めの位置を指定しておかなければなりません。OPEN文の実行が成功すると、ファイルは指定した位置に位置付けられます。

    c. REVERSED指定を書いた場合、OPEN文を実行すると、ファイルは終わりに位置付けられます。

11. REVERSED指定を書いた場合、OPEN文を実行した後にファイル名-nのファイルに対するREAD文を実行すると、最後のレコードから逆順にレコードが使用可能になります。

12. 行順ファイルおよび印刷ファイルに対して、ラベル処理は行われません。

13. 印刷ファイルをINPUT指定のOPEN文またはI-O指定のOPEN文で開いてはいけません。

14. 順ファイルに対するOPEN文の実行が成功すると、ボリューム指示子は以下のように設定されます。

    a. 使用可能なファイルに対してINPUT指定またはI-O指定のOPEN文を実行した場合、最初のリール/ユニットまたは唯一のリール/ユニットを指すように設定されます。

    b. EXTEND指定のOPEN文を実行した場合、最後のレコードを含むリール/ユニットを指すように設定されます。

    c. 使用不可能なファイルに対してOUTPUT指定またはI-O指定のOPEN文を実行した場合、新しいリール/ユニットを指すように設定されます。

相対ファイルの規則

1. INPUT指定またはI-O指定のOPEN文の実行が成功すると、ファイル名-nのファイル位置指子は1に設定されます。

2. EXTEND指定のOPEN文の実行が成功すると、ファイルはその最後のレコードの直後に位置付けられます。相対ファイルの最後のレコードは、すでに存在するレコードのうち最高の相対レコード番号を持つものです。

索引ファイルの規則

1. INPUT指定またはI-O指定のOPEN文の実行が成功すると、ファイル名-nのファイル位置指示子は、そのファイルの主レコードキーが取りうる最小の値に設定されます。また、主レコードキーが参照キーとして設定されます。

2. EXTEND指定のOPEN文の実行が成功すると、ファイルはその最後のレコードの直後に位置付けられます。索引ファイルの最後のレコードは、すでに存在するレコードのうち最大の主レコードキーの値を持つレコードです。

   ただし、RECORD KEY句にDUPLICATES指定を書いた場合、索引ファイルの最後のレコードは、すでに存在するレコードで最大の主レコードキーの値を持つレコードのうち、最後に生成されたレコードです。

6.4.31 OPEN文（表示ファイル）

ファイルを使用可能な状態にします。

【書き方】

```
OPEN { INPUT  {ファイル名-1} …
     { OUTPUT {ファイル名-2} …   } …
     { I-O    {ファイル名-3} …
```

構文規則

ファイル名-1～ファイル名-3の並びには、表示ファイルの他に順ファイル、相対ファイルまたは索引ファイルも指定でき、ORGANIZATION句の指定が異なるファイルまたはACCESS MODE句の指定が異なるファイルを指定することもできます。

一般規則

以下で、“ファイル名-n”は“ファイル名-1”、“ファイル名-2”または“ファイル名-3”に指定したファイルを表します。

1. OPEN文の実行が成功すると、ファイル名-nのファイルはプログラムで使用可能になり、開いた状態になります。OPEN文の実行が成功すると、そのファイルは、ファイル結合子を通してファイル名と関連付けられます。
2. OPEN文の実行が成功すると、ファイル名-nのレコード領域が使用可能になります。
3. ファイルを開く前に、ファイルに対する入出力文を実行することはできません。
4. INPUT指定、OUTPUT指定およびI-O指定は、ファイルのオープンモードを指定します。ファイルを開いた後に実行できる入出力文は、オープンモードによって異なります。オープンモードと入出力文の関係については、“2.2.3 入出力文の動作”を参照してください。
5. ファイル名-nに対して指定できるオープンモードは、ファイル名-nのSYMBOLIC DESTINATION句で指定したあて先種別によって異なります。下表に、あて先種別とオープンモードの関係を示します。

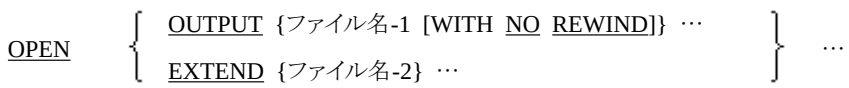
あて先種別	OPEN文の指定		
	INPUT	OUTPUT	I-O
ACM	実行可能	実行可能	実行可能
DSP	実行不可能	実行不可能	実行可能
PRT	実行不可能	実行可能	実行可能
APL	実行可能	実行可能	実行可能
TRM	実行不可能	実行可能	実行可能

6. OPEN文の実行は、最初のレコードの読み書きを行いません。
7. OPEN文を実行すると、ファイル名-nの入出力状態の値が更新されます。
8. ファイル名-nを2つ以上指定した場合のOPEN文の実行結果は、ファイル名-nを書いた順に、各ファイルに対して別々のOPEN文を実行した結果と同じです。

6.4.32 OPEN文（報告書作成）

ファイルを使用可能な状態にします。

【書き方】



注意

[WinIPF][Winx64][LinuxIPF][Linux64][.NET]では報告書作成機能は使用できません。

構文規則

報告書ファイルに対するOPEN文では、OUTPUT指定またはEXTEND指定だけを書くことができます。

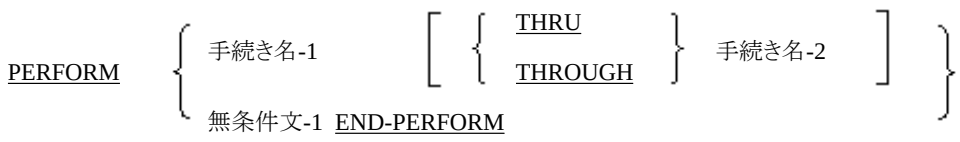
一般規則

1. 報告書ファイルに対するOPEN文は、そのファイルに関連する報告書に対してINITIALIZE文を実行する前に実行しなければなりません。
2. 報告書ファイルのOPEN文の各句の規則は、順ファイルのOPEN文の規則と同じです。“6.4.30 OPEN文（順ファイル・相対ファイル・索引ファイル）”を参照してください。

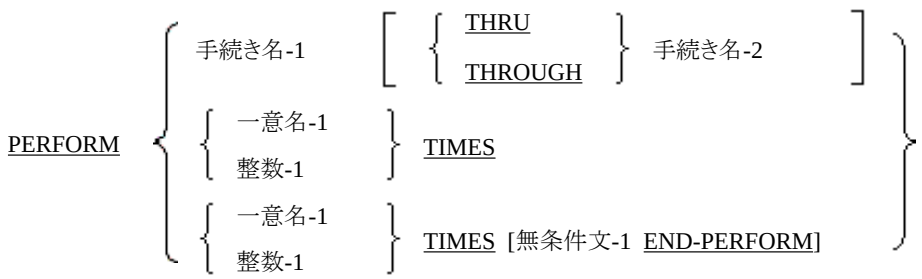
6.4.33 PERFORM文（中核）

一連の文の組を繰り返し実行します。

【書き方1】文の組を1回だけ実行する



【書き方2】文の組の実行を、特定の回数だけ繰り返す



【書き方3】文の組の実行を、条件を満足するまで繰り返す

```
PERFORM [ 手続き名-1 [ { THRU     } 手続き名-2 ] ]
                      { THROUGH }
        [ WITH TEST { BEFORE } ] UNTIL 条件-1
                    { AFTER  }
        [無条件文-1 END-PERFORM]
```

【書き方4】文の組の実行を条件を満足するまで繰り返すと同時に、繰り返し回数に応じてデータ項目または指標名の値を変化させる

```
PERFORM [ 手続き名-1 [ { THRU     } 手続き名-2 ] ]
                       { THROUGH }
    [ WITH TEST { BEFORE } ]
                { AFTER  }
    VARYING { 一意名-2 }
            { 指標名-1 }
    FROM { 一意名-3 } BY { 一意名-4 } UNTIL 条件-1
         { 指標名-2 }    {        }
         { 整数-1   }    { 整数-2 }
    [ AFTER { 一意名-5 } ]
            { 指標名-3 }
    FROM { 一意名-6 } BY { 一意名-7 } UNTIL 条件-2 ] …
         { 指標名-4 }    {        }
         { 整数-3   }    { 整数-4 }
    [無条件文-1 END-PERFORM]
```

【書き方5】文の組の実行を無条件に繰り返す

```
PERFORM WITH NO LIMIT
        無条件文-1 END-PERFORM
```

構文規則

書き方1～書き方5に共通する規則

1. 書き方1～書き方4では、手続き名-1と無条件文-1のいずれか一方を書かなければなりません。また、手続き名-1と無条件文-1の両方を同時に書くことはできません。
2. THROUGHとTHRUは同義語です。
3. 手続き名-1と手続き名-2の両方を書く場合、一方の手続き名が手続き部の宣言部分の中の手続き名のとき、他方の手続き名も同一の宣言節の中になければなりません。
4. 書き方3と書き方4で、TEST BEFORE指定もTEST AFTER指定も書かない場合は、TEST BEFORE指定を書いたものとみなされます。
5. 書き方3と書き方4で、条件-1および条件-2には、“6.3.3 条件式”で説明した条件式を書きます。

書き方2の規則

1. 一意名-1は、整数項目でなければなりません。
2. 整数-1および一意名-1に設定できる値の最大値については、“付録B システムの定量制限”を参照してください。

書き方4の規則

1. 無条件文-1を書く場合、AFTER指定を書くことはできません。
2. 一意名-2～一意名-7は、数字項目でなければなりません。
3. 定数-1～定数-4は、数字定数でなければなりません。
4. 定数-2および定数-4は、ゼロであってはいけません。
5. VARYING指定に指標名-1を書く場合、FROM指定およびBY指定の作用対象は、以下の規則に従わなければなりません。
    a. 一意名-3および一意名-4は、整数項目でなければなりません。
    b. 定数-1は、正の整数でなければなりません。
    c. 定数-2は、ゼロでない整数でなければなりません。
6. AFTER指定に指標名-3を書く場合、FROM指定およびBY指定の作用対象は、以下の規則に従わなければなりません。
    a. 一意名-6および一意名-7は、整数項目でなければなりません。
    b. 定数-3は、正の整数でなければなりません。
    c. 定数-4は、ゼロでない整数でなければなりません。
7. VARYING指定の中のFROM指定に指標名-2を書く場合、VARYING指定およびBY指定の作用対象は、以下の規則に従わなければなりません。
    a. 一意名-2および一意名-4は、整数項目でなければなりません。
    b. 定数-2は、ゼロでない整数でなければなりません。
8. AFTER指定の中のFROM指定に指標名-4を書く場合、VARYING指定およびBY指定の作用対象は、以下の規則に従わなければなりません。
    a. 一意名-5および一意名-7は、整数項目でなければなりません。
    b. 定数-4は、ゼロでない整数でなければなりません。
9. AFTER指定は、最大6個まで書くことができます。

一般規則

PERFORM文の全般規則

1. 手続き名-1を書いたPERFORM文を「そとPERFORM文」といい、無条件文-1を書いたPERFORM文を「うちPERFORM文」といいます。PERFORM文によって実行される一連の文を「文の組」といいます。
2. そとPERFORM文は、別の節または段落に書いた文の組を、指定した回数または条件に従って繰り返し実行します。
3. うちPERFORM文は、うちPERFORM文に書いた文の組(無条件文-1)を、指定した回数または条件に従って繰り返し実行します。
4. そとPERFORM文とうちPERFORM文では、実行される文の組をPERFORM文の外に書くか中に書くかを除いて、同じ規則が適用されます。

PERFORM文で実行される文の組の規則

1. PERFORM文は、PERFORM文の終わりに制御が戻るまでに実行される文によって論理的に構成されます。文の組の中にCALL文、EXIT文、GO TO文またはPERFORM文を書いた場合、それらの文によって実行されるすべての文が、PERFORM文の範囲に含まれます。文の組の中で宣言手続きが実行された場合、宣言手続きで実行さ

れた文もPERFORM文の範囲に含まれます。PERFORM文の範囲に含まれる文は、プログラムの中で連続して書く必要はありません。

2. PERFORM文の範囲の中で、そのPERFORM文と同じプログラムの中に書いたEXIT PROGRAM文を実行した場合、EXIT PROGRAM文の実行による制御の移行の結果として実行される文は、PERFORM文の範囲とはみなされません。

うちPERFORM文で実行される文の組の規則

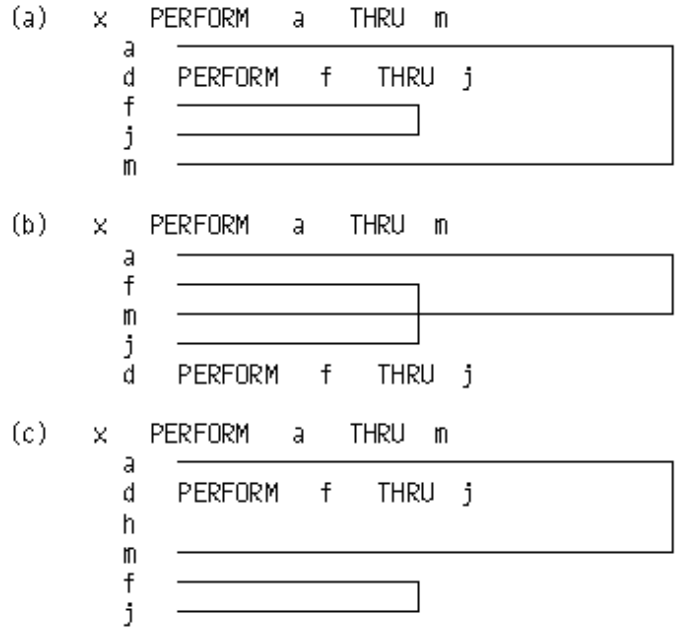
1. うちPERFORM文は、無条件文-1に指定した文の組を繰り返し実行します。
2. END-PERFORM指定は、うちPERFORM文の範囲を区切ります。
3. うちPERFORM文で実行される文の組は、無条件文-1の最初の文からEND-PERFORM指定の直前の文までに実行される文です。

そとPERFORM文で実行される文の組の規則

1. そとPERFORM文は、別の節または段落に書いた文の組を繰り返し実行します。手続き名-1の最初の文から以下の文までを、繰り返し実行します。
    a. 手続き名-2を省略し、手続き名-1に段落名を指定した場合、その段落の最後の文まで
    b. 手続き名-2を省略し、手続き名-1に節名を指定した場合、その節の最後の段落の最後の文まで
    c. 手続き名-2に段落名を指定した場合、その段落の最後の文まで
    d. 手続き名-2に節名を指定した場合、その節の最後の段落の最後の文まで
2. 手続き名-1と手続き名-2の関係は、制御の流れが手続き名-1から始まって手続き名-2に至ることを指定します。手続き名-1から手続き名-2の終わりまでの間に、GO TO文またはPERFORM文を書くこともできます。文の組の中の最後の文以外からPERFORM文に制御を戻す場合、EXIT文だけの段落を定義し、EXIT文だけの段落からPERFORM文に制御を戻さなければなりません。このとき、手続き名-2を書かなければなりません。手続き名-2は、EXIT文だけの段落の段落名、またはEXIT文だけの段落を含む節の節名でなければなりません。
3. 文の組は、PERFORM文以外で実行することができます。PERFORM文以外から、文の組に制御が移った場合、文の組の最後の文からPERFORM文に制御が戻ることはありません。文の組の最後の文の次の実行文に、制御が移ります。
4. 手続き名-1および手続き名-2は、PERFORM文を書いたプログラム中の節名または段落名でなければなりません。PERFORM文を書いたプログラムを含むプログラム中の手続き名も、PERFORM文を書いたプログラムに含まれるプログラム中の手続き名も、手続き名-1および手続き名-2に指定することはできません。
5. PERFORM文の範囲の中に別のPERFORM文を書く場合、以下の規則に従わなければなりません。前者のPERFORM文を「含むPERFORM文」といい、後者のPERFORM文を「含まれるPERFORM文」といいます。
    a. 含まれるPERFORM文の範囲は、含むPERFORM文の論理的な実行範囲の内側に完全に含まれているか、または完全に外側でなければなりません。すなわち、含まれるPERFORM文の実行中に、含むPERFORM文の出口を通ることはできません。
    b. 含むPERFORM文の範囲の中にある1つ以上のPERFORM文は、互いに共通の出口を持つことはできません。

        ただし、このコンパイラでは、含むPERFORM文の範囲の中にある1つ以上のPERFORM文が、互いに共通の出口を持つことができます。

6. 以下に、そとPERFORM文の正しい使い方の例を示します。



書き方1の規則

書き方1のPERFORM文は、文の組を一回だけ実行します。その後、PERFORM文の終わりに制御が戻ります。

書き方2の規則

1. 書き方2のPERFORM文は、整数-1または一意名-1の初期値で指定した回数だけ、文の組を繰り返し実行します。その後、PERFORM文の終わりに制御が移ります。
2. PERFORM文を実行する前の一意名-1の値がゼロまたは負の場合、文の組を実行しないでPERFORM文の終わりに制御が移ります。
3. 文の組の実行中に一意名-1の値を変更しても、文の組の実行回数は変わりません。

書き方3の規則

1. 書き方3のPERFORM文は、条件-1を満足するまで、文の組を繰り返し実行します。その後、PERFORM文の終わりに制御が移ります。
2. 文の組を実行する前に条件-1を検査する場合、TEST BEFORE指定を書くか、またはTEST指定を省略します。文の組を実行した後に条件-1を検査する場合、TEST AFTER指定を書きます。
3. 条件-1の中の作用対象に添字または部分参照子を付けた場合、それらは、条件-1が検査されるごとに毎回評価されます。

書き方4の規則

1. 書き方4のPERFORM文は、特定の条件を満足するまで文の組を繰り返し実行し、繰り返しに応じてデータ項目の値を変化させます。その後、PERFORM文の終わりに制御が移ります。
   そとPERFORM文の場合、繰返しに応じて変化させるデータ項目は、VARYING指定とAFTER指定に1つ以上指定することができます。うちPERFORM文の場合、繰返しに応じて変化させるデータ項目は、VARYING指定に1つだけ指定することができます。
   繰返しの終了条件はUNTILの後に書きます。文の組を実行する前に、終了条件を検査する場合、TEST BEFORE指定を書くか、またはTEST指定を省略します。文の組を実行した後に、終了条件を検査する場合、TEST AFTER指定を書きます。
2. 一意名-4および一意名-7の値は、ゼロであってはいけません。
3. VARYING指定に指標名-1を書いた場合、FROM指定の作用対象の値は、指標名-1を持つ表の出現番号に対応する値でなければなりません。また、BY指定の作用対象の値は、指標名-1を持つ表の出現番号に対応する値の増分値でなければなりません。同様に、AFTER指定に指標名-3を書いた場合、FROM指定の作用対象の値は、

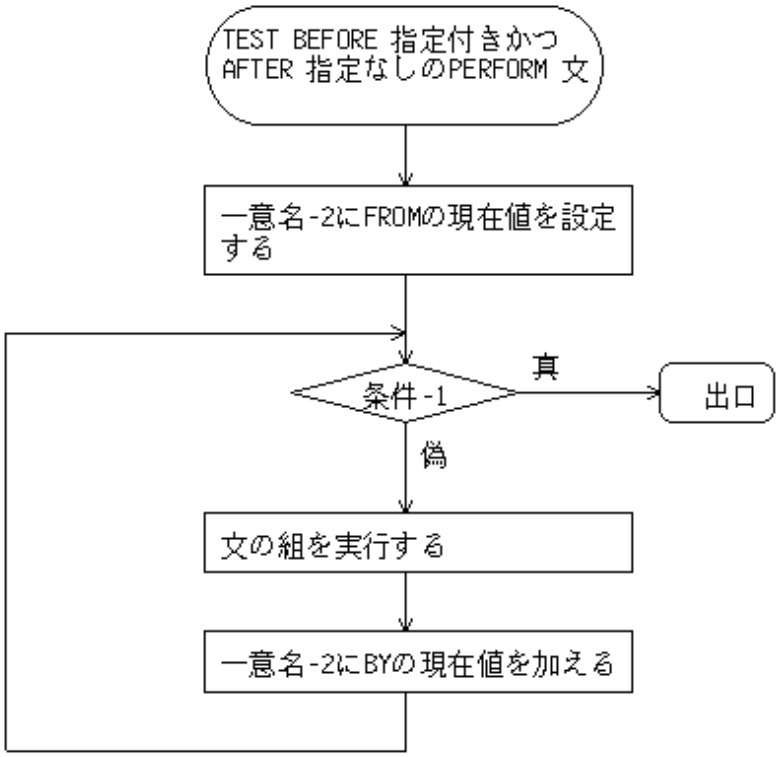
指標名-3を持つ表の出現番号に対応する値でなければなりません。また、BY指定の作用対象の値は、指標名-3を持つ表の出現番号に対応する値の増分値でなければなりません。

4. 条件-1を満足するまでに、指標名-1に、指標名-1を持つ表の範囲外の値を設定することはできません。同様に、条件-2を満足するまでに、指標名-3に、指標名-3を持つ表の範囲外の値を設定することはできません。ただし、PERFORM文の実行が終わったときの指標名-1の値が、1つの増分値または減分値だけ、指標名-1を持つ表の範囲外の値になることがあります。

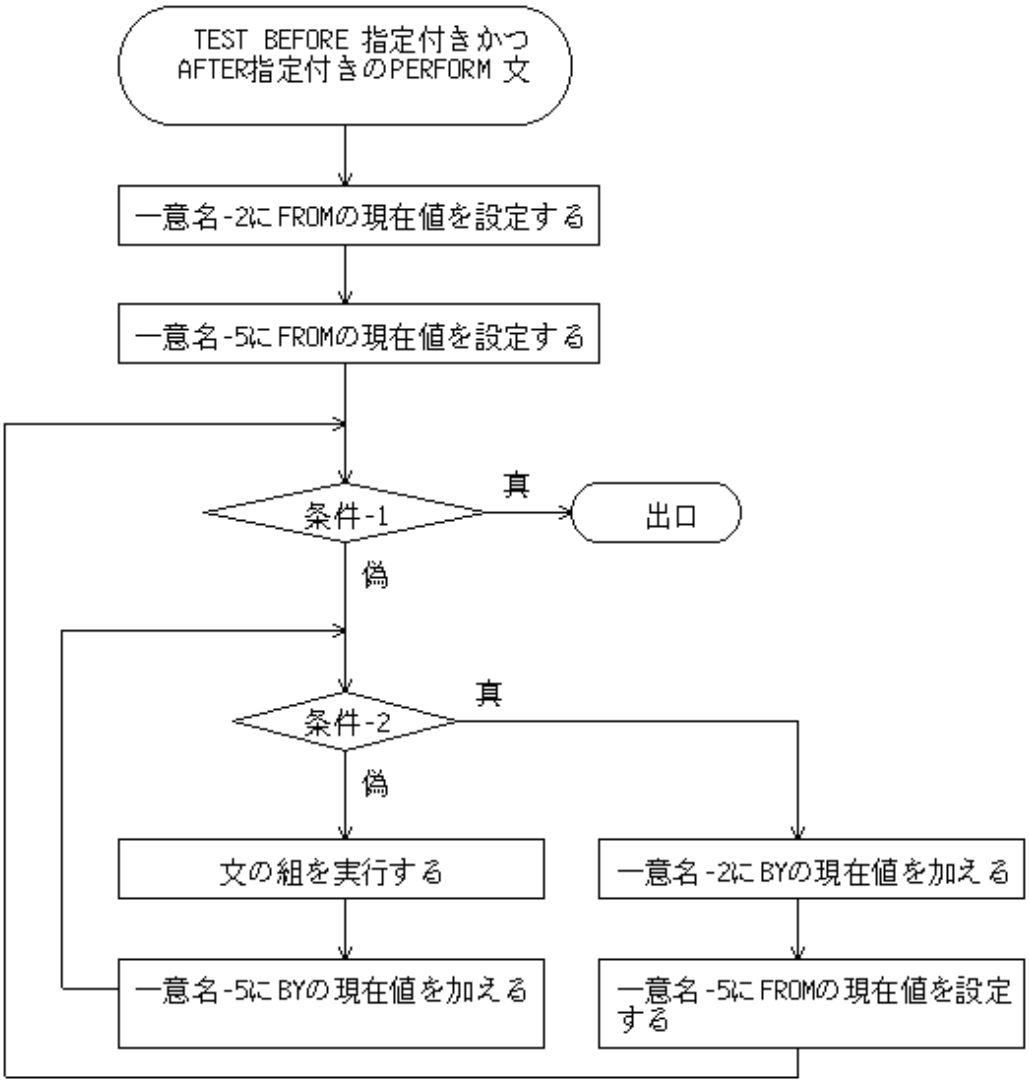
5. 一意名またはUNTIL指定の条件の作用対象に添字または部分参照子を付けた場合、それらは以下の時点に評価されます。

   - 一意名-2および一意名-5の添字は、これらの一意名に初期値を設定するときおよび増分値を加算するとき、毎回評価されます。
   - 一意名-3および一意名-6の添字は、これらの一意名に初期値を設定するとき評価されます。
   - 一意名-4および一意名-7の添字は、一意名-2または一意名-5に増分値を加算するとき評価されます。
   - 条件-1および条件-2の中の一意名に付けた添字および部分参照子は、条件-1または条件-2が検査されるとき、毎回評価されます。

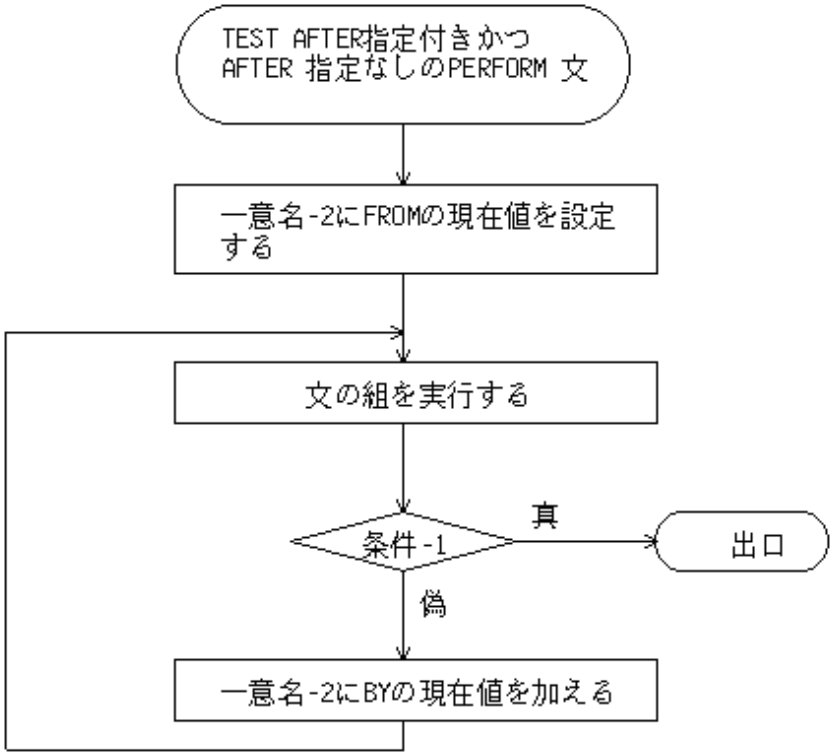
6. TEST BEFORE指定を明にまたは暗に指定し、AFTER指定を省略した場合、下図で示すように処理が行われます。



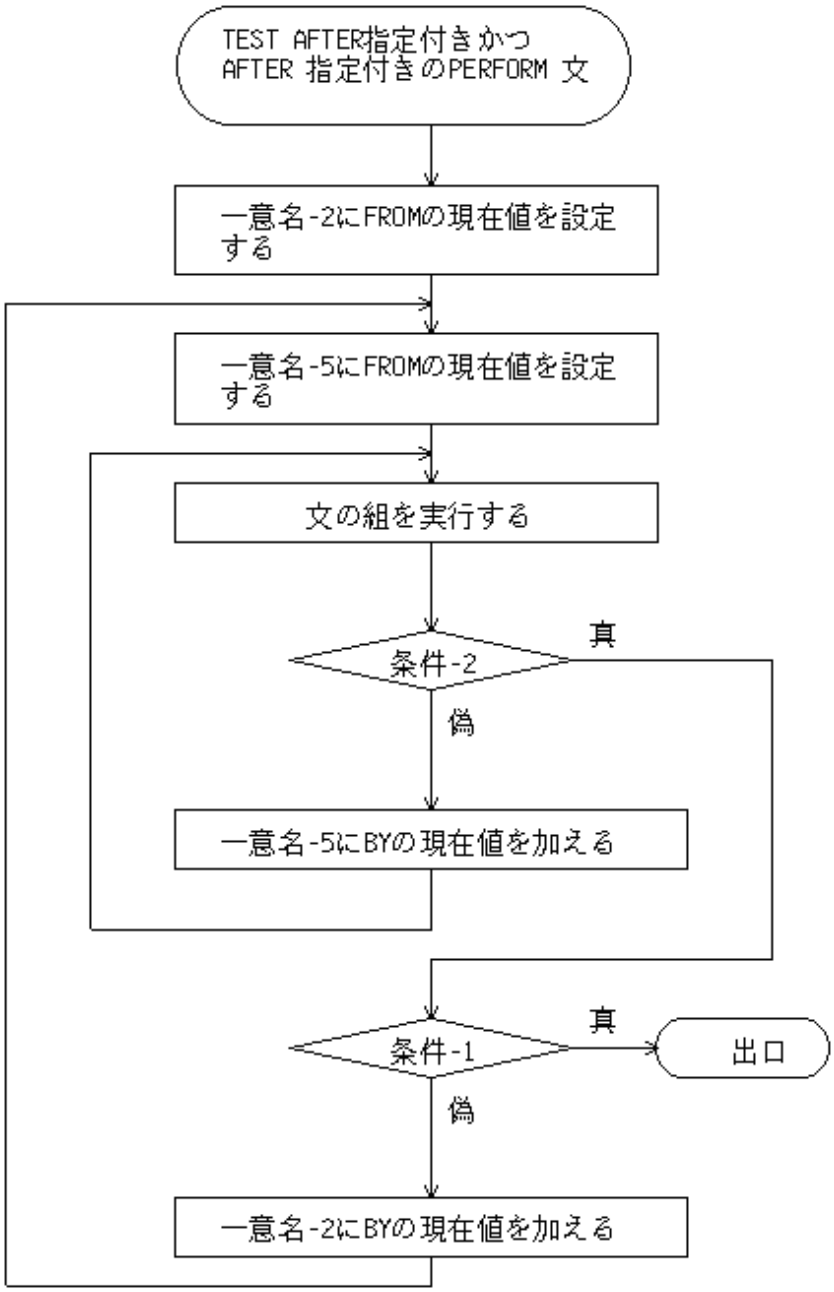
7. TEST BEFORE指定を明にまたは暗に指定し、AFTER指定を書いた場合、下図で示すように処理が行われます。



8. TEST AFTER指定を書き、AFTER指定を省略した場合、下図で示すように処理が行われます。



9. TEST AFTER指定を書き、AFTER指定を1つ書いた場合、下図で示すように処理が行われます。



10. AFTER指定を2つ以上書いた場合、AFTER指定の検査機構は、AFTER指定を1つしか書かなかった場合と同じです。ただし、先行するAFTER指定の作用対象の値が変更されるごとに各AFTER指定の作用対象の処理が一巡します。

書き方5の規則

書き方5のPERFORM文は、文の組を繰り返し実行します。この繰り返しは、制御の明示移行を示す分岐文またはEXIT PERFORM文を実行したときに終了します。

6.4.34 READ文（順ファイル・相対ファイル・索引ファイル）

ファイルからレコードを読みます。

【書き方1】レコードを順に読む（順ファイル・相対ファイル・索引ファイル）

READ ファイル名-1 [NEXT] RECORD
[INTO 一意名-1]
[WITH [NO] LOCK]
[AT END 無条件文-1]
[NOT AT END 無条件文-2]
[END-READ]

【書き方2】レコードを乱に読む（相対ファイル）

READ ファイル名-1 RECORD
[INTO 一意名-1]
[WITH [NO] LOCK]
[INVALID KEY 無条件文-3]
[NOT INVALID KEY 無条件文-4]
[END-READ]

【書き方3】レコードを乱に読む（索引ファイル）

READ ファイル名-1 RECORD
[INTO 一意名-1]
[WITH [NO] LOCK]
[KEY IS {データ名-1} …]
[INVALID KEY 無条件文-3]
[NOT INVALID KEY 無条件文-4]
[END-READ]

構文規則

各ファイルに共通する規則

1. 一意名-1の記憶領域とファイル名-1のレコード領域は、異なる領域でなければなりません。
2. ファイル名-1に関連するUSE AFTER STANDARD EXCEPTION手続きを書かない場合、AT END指定またはINVALID KEY指定を書かなければなりません。

   ただし、このコンパイラでは、USE STANDARD EXCEPTION手続きとAT END指定またはINVALID KEY指定の両方を省略することができます。
3. 一意名-1に、定数節で定義したデータ項目を指定することはできません。
4. 一意名-1が強く型付けされた集団項目である場合、ファイル名-1のレコード記述項には、同じ型で強く型付けされた1つの集団項目だけを指定できます。

相対ファイルの規則

1. 順呼出し法のファイルからレコードを読む場合、書き方1を使います。
2. 動的呼出し法のファイルから順にレコードを読む場合、書き方1を使います。NEXT指定は書かなければなりません。

3. 乱呼出し法のファイルからレコードを読む場合、または動的呼出し法のファイルから乱にレコードを読む場合、書き方2を使います。

索引ファイルの規則

1. 順呼出し法のファイルからレコードを読む場合、書き方1を使います。
2. 動的呼出し法のファイルから順にレコードを読む場合、書き方1を使います。NEXT指定は書かなければなりません。
3. 乱呼出し法のファイルからレコードを読む場合、または動的呼出し法のファイルから乱にレコードを読む場合、書き方3を使います。
4. データ名-1は、ファイル名-1のRECORD KEY句またはALTERNATE RECORD KEY句に指定したデータ名でなければなりません。RECORD KEY句のデータ名を指定する場合、RECORD KEY句にデータ名を2つ以上指定しているならば、データ名-1の並びは、RECORD KEY句に指定したデータ名の並びと、指定順序および個数とも同じでなければなりません。また、ALTERNATE RECORD KEY句のデータ名を指定する場合、ALTERNATE RECORD KEY句にデータ名を2つ以上指定しているならば、データ名-1の並びはALTERNATE RECORD KEY句に指定したデータ名の並びと、指定順序および個数とも同じでなければなりません。
5. データ名-1は、修飾することができます。

一般規則

書き方1～書き方3に共通する規則

各ファイルに共通する規則

1. ファイル名-1のファイルは、READ文を実行する前に入力モードまたは入出力両用モードで開いておかなければなりません。
2. READ文を実行すると、ファイルから読み込むレコードが選択され、ファイル名-1のファイル位置指示子の値が設定されます。また、ファイル名-1の入出力状態の値が更新されます。ファイル位置指示子の値によって次にとる動作が決定されます。
3. READ文の実行が不成功になった場合、ファイル名-1のレコード領域の内容は規定されません。ファイル位置指示子は、次の有効なレコードがないことを示すように設定されます。
4. ファイル位置指示子の値が有効なレコードがあることを示している場合、レコードは、レコード領域で使用可能になります。INTO指定を書いた場合、レコード領域から一意名-1へレコードが転送されます。
5. 読み込んだレコードの文字位置の個数がファイル名-1のレコード記述項で指定した最小レコードの大きさより小さい場合、ファイル名-1のレコード領域のうち、読み込んだレコードの文字位置の個数を超える部分の内容は規定されません。読み込んだレコードの文字位置の個数がファイル名-1のレコード記述項で指定した最大レコードの大きさより大きい場合、読み込んだレコードの右側が最大レコードの大きさに合わせて切り捨てられ、ファイル名-1のレコード領域に設定されます。どちらの場合も、READ文は成功します。ファイル名-1の入出力状態には、レコード長矛盾を示す値が設定されます。
6. END-READ指定は、READ文の範囲を区切ります。

INTO指定の規則

1. INTO指定は、以下の場合に書くことができます。
    a. ファイル名-1のレコード記述項を1つだけ書いた場合
    b. ファイル名-1に関連するすべてのレコード名および一意名-1が、集団項目または英数字項目の場合
2. INTO指定を書いた場合、以下の順に処理が行われます。
    a. INTO指定のない同じREAD文が実行されます。
    b. a.のREAD文の実行が成功した場合、現在のレコードが、CORRESPONDING指定なしのMOVE文の規則に従って、ファイル名-1のレコード領域から一意名-1の領域へ転記されます。読み込まれたレコードは、ファイル名-1のレコード領域と一意名-1のデータ項目の両方で使用可能になります。
    現在のレコードの大きさは、ファイル名-1のRECORD句の規則に従って決定されます。ファイル名-1のファイル記述項に書き方3以外のRECORD句を書いた場合、b.では集団項目転記が行われます。一意

名-1に付けた添字は、a.の後b.の前に評価されます。a.のREAD文が不成功の場合、暗黙のMOVE文は実行されません。

レコードのロックの規則

1. LOCK MODE IS AUTOMATIC句を指定したファイルに対して以下のREAD文を実行すると、読み込まれたレコードがロックされます。
    a. WITH LOCK指定を書いたREAD文
    b. WITH LOCK指定もWITH NO LOCK指定も書いていないREAD文
2. LOCK MODE IS MANUAL句を指定したファイル(相対ファイルまたは索引ファイル)に対して、WITH LOCK指定を書いたREAD文を実行すると、読み込まれたレコードがロックされます。
3. 入力モードで開いたファイルに対してREAD文を実行した場合、レコードはロックされません。WITH LOCK指定を書いても書かなくても同じです。
4. 明にまたは暗にロックを伴うREAD文の実行が開始されるとき、そのREAD文によって読み込まれるレコードが、他のファイル結合子によってロックされているならば、そのREAD文の実行は不成功になります。ファイル名-1の入出力状態には、レコードのロックを示す値が、設定されます。このとき、ファイル位置指示子の値は変更されません。

書き方1の規則

各ファイルに共通する規則

1. ファイル名-1に順ファイル、順呼出し法の相対ファイルまたは順呼出し法の索引ファイルを指定した場合、NEXT指定は省略することができます。NEXT指定はREAD文の実行には影響を与えません。
2. ファイル名-1に順ファイル、順呼出し法の相対ファイルまたは順呼出し法の索引ファイルを指定したREAD文は、ファイル名-1のファイルから次のレコードを呼び出します。
3. ファイル名-1に動的呼び出し法の相対ファイルまたは動的呼び出し法の索引ファイルを指定し、NEXT指定を書いたREAD文は、ファイル名-1から次のレコードを呼び出します。
4. READ文を実行すると、使用可能なレコードを定めるための規則に従って、ファイル位置指示子の値が決定されます。ファイル位置指示子の値によって、次にとる動作が決定されます。

使用可能なレコードを定めるための規則(順ファイルの場合)

1. READ文の実行が開始されるときのファイル位置指示子の値は、以下の規則に従って使用可能なレコードを定めるために使われます。順ファイルでのレコードの比較は、レコード番号によって行われます。
    a. ファイル位置指示子が次の有効なレコードが存在しないことを示している場合、READ文の実行は不成功になります。
    b. ファイル位置指示子が不定ファイルが存在しないことを示している場合、ファイル終了条件が発生します。
    c. 先に実行したOPEN文によりファイル位置指示子が設定されている場合、ファイル位置指示子より大きいかまたは等しいレコード番号を持つ、ファイル中の最初のレコードが選択されます。
    d. 先に実行したREAD文によりファイル位置指示子が設定されている場合、ファイル位置指示子より大きいレコード番号を持つ、ファイル中の最初のレコードが選択されます。
2. 1.のc.またはd.の条件を満足するレコードが見つからなかった場合、ファイル終了条件が発生します。ファイル位置指示子には、次のレコードが存在しないことを示す値が設定されます。
3. 1.のc.またはd.の条件を満足するレコードが見つかった場合、そのレコードがファイル名-1に関連するレコード領域で使用可能になります。そして、ファイル位置指示子が、使用可能になったレコードのレコード番号に設定されます。

使用可能なレコードを定めるための規則(相対ファイルの場合)

1. READ文の実行が開始されるときのファイル位置指示子の値は、以下の規則に従って使用可能なレコードを定めるために使われます。相対ファイルでのレコードの比較は、相対キー番号によって行われます。

    a. ファイル位置指示子が有効な次のレコードがないことを示しているとき、READ文の実行は不成功になります。

    b. ファイル位置指示子が不定ファイルが存在しないことを示している場合、ファイル終了条件が発生します。

    c. 先に実行されたOPEN文によりファイル位置指示子が設定されている場合、ファイル位置指示子と等しいかまたは大きい相対レコード番号を持つ最初のレコードが選択されます。

    d. 先に実行されたSTART文によりファイル位置指示子が設定されている場合、そのファイル位置指示子と同じ相対レコード番号を持つレコードが選択されます。

    e. 先に実行されたREAD文によりファイル位置指示子が設定されている場合、そのファイル位置指示子より大きい相対レコード番号を持つ最初のレコードが選択されます。

2. 1.のc.またはd.の条件を満足するレコードが見つからなかった場合、ファイル終了条件が発生します。ファイル位置指示子には、次のレコードが存在しないことを示す値が設定されます。

3. 1.のc.またはd.の条件を満足するレコード(「選択されたレコード」という)が見つかった場合、以下の処理が行われます。

    a. ファイル名-1に対してRELATIVE KEY指定を書いた場合、選択されたレコードの相対レコード番号の有効桁数と相対キー項目の大きさが比較されます。

    相対レコード番号の有効桁数が相対キー項目の大きさより大きい場合、ファイル終了条件が発生します。

    このとき、ファイル位置指示子には、相対レコード番号の有効桁数が相対キー項目の大きさを超えたことを示す値が設定されます。

    相対レコード番号の有効桁数が相対キー項目の大きさ以下の場合、選択されたレコードがファイル名-1のレコード領域で使用可能になります。このとき、ファイル位置指示子には、使用可能になったレコードの相対レコード番号が設定されます。また、転記の規則に従って、使用可能になったレコードの相対レコード番号が相対キー項目に転記されます。

    b. ファイル名-1に対してRELATIVE KEY指定を書かなかった場合、選択されたレコードが、ファイル名-1のレコード領域で使用可能になります。このとき、ファイル位置指示子には、使用可能になったレコードの相対レコード番号が設定されます。

使用可能なレコードを定めるための規則(索引ファイルの場合)

1. READ文の実行が開始されるときのファイル位置指示子の値は、以下の規則に従って、使用可能なレコードを定めるために使われます。索引ファイルでのレコードの比較は、現在の参照キーの値の大小順序によって行われます。

    a. ファイル位置指示子が次のレコードが存在しないことを示している場合、READ文の実行は不成功になります。

    b. ファイル位置指示子が不定ファイルが存在しないことを示している場合、ファイル終了条件が発生します。

    c. 先に実行されたOPEN文によりファイル位置指示子が設定されている場合、現在の参照キーによるファイル中の先頭レコードが選択されます。

    d. 先に実行されたSTART文によりファイル位置指示子が設定されている場合、そのファイル位置指示子と同じキー値を持つレコードが選択されます。

    e. 先に実行されたREAD文によりファイル位置指示子が設定されている場合、以下のいずれかの条件を満足するレコードが選択されます。

    - 先に実行されたREAD文により使用可能となったレコードの、論理的に直後のレコード

    - REVERSED ORDER指定付きのSTART文の実行により検索方向が逆順であることが指定されている場合、先に実行されたREAD文により使用可能となったレコードの論理的に直前のレコード。すなわち、ファイルの末尾のレコードから先頭のレコードへ向かう方向において、論理的に次のレコード

2. 1.のc.、d.およびe.で、条件を満足するレコードが見つからなかった場合、ファイル終了条件が発生します。ファイル位置指示子には、次のレコードが存在しないことを示す値が設定されます。
3. 1.のc.、d.およびe.で、条件を満足するレコードが見つかった場合、そのレコードがファイル名-1のレコード領域で使用可能になります。そして、ファイル位置指示子には、使用可能になったレコードの現在の参照キーの値が設定されます。
4. READ文の実行が不成功になった場合、参照キーの値は規定されません。
5. 主レコードキーまたは副レコードキーを参照キーとしてREAD文を実行した場合、参照キーの値を持つレコードが2つ以上存在することがあります。その場合、以下の順にレコードが読み込まれます。
    a. 先に実行したREVERSED ORDER指定付きのSTART文によりファイル位置指示子が設定されている場合
    WRITE文またはREWRITE文で、参照キーの値が重複するレコードを生成した順の逆の順に、レコードが読み込まれます。
    b. 上記以外の場合
    WRITE文またはREWRITE文で、参照キーの値が重複するレコードを生成した順に、レコードが読み込まれます。
6. 動的呼出し法のファイルに対して、書き方3のREAD文に続いて書き方1のREAD文を実行すると、書き方1のREAD文では以下に示す参照キーが使われ、かつ以下に示す検索方向でレコードが読み込まれます。
    a. 参照キーが変更されるまで、書き方3のREAD文の実行によって設定された参照キーが使用されます。
    b. REVERSED ORDER指定付きのSTART文が実行されるまで、正順で検索されます。

例外条件の発生の有無による制御の移行

1. READ文の実行中にファイル終了条件が発生すると、READ文の実行が不成功になります。ファイル名-1の入出力状態にファイル終了条件を示す値が設定された後、“6.3.13 AT END指定”の規則に従って制御が移ります。
2. READ文の実行中にファイル終了条件以外の例外条件が発生すると、ファイル位置指示子およびファイル名-1の入出力状態が設定された後、“6.3.13 AT END指定”の規則に従って制御が移ります。
3. READ文の実行中にファイル終了条件もその他の例外条件も発生しなかった場合、以下の処理が順に行われます。
    a. ファイル位置指示子およびファイル名-1の入出力状態が設定されます。
    b. 読み込んだレコードがレコード領域で使用可能になります。INTO指定を書いた場合、暗黙の転記も実行されます。
    c. “6.3.13 AT END指定”の規則に従って制御が移ります。

順ファイルのリール/ユニットファイルに関する規則

順ファイルに対するREAD文の実行中に、リール/ユニットの終わりが検出された場合、またはリール/ユニットにレコードが存在しなかった場合、ファイルの論理的な終わりに到達していないならば、以下の処理が順に行われます。

a. 標準の終わりリール/ユニットラベル手続きが実行されます。
b. リール/ユニットの交換が行われます。そして、ボリューム指示子がそのファイルの次のリール/ユニットを指すように更新されます。
c. 標準の始めリール/ユニットラベル手続きが実行されます。

行順ファイルの規則

行順ファイルに対するREAD文を実行すると、レコードのデータは以下のように設定されます。以降の説明で、「レコードの区切り文字までの長さ」は、レコードの区切り文字を含みません。

a. レコードの区切り文字までの長さが最大レコード長と同じ場合、レコードの区切り文字までの部分がレコード領域に設定されます。
b. レコードの区切り文字までの長さが最大レコード長より短い場合、レコードの区切り文字までの部分がレコード領域に設定され、レコード領域の残りの部分には空白が設定されます。

c. レコードの区切り文字までの長さが最大レコード長より長い場合、2回以上のREAD文を繰り返し実行することによって、レコードのデータ全体が読み込まれます。1回目のREAD文では、レコードの先頭から始まる、最大レコード長と同じ長さのデータが読み込まれます。2回目以降のREAD文では、先行するREAD文によって読み込まれた部分の次の文字位置から始まるデータが読み込まれます。READ文の繰返しでは、a.～c.の規則が適用されます。

書き方2の規則

1. READ文の実行が開始されるときのファイル位置指示子が、入力の不定ファイルが存在しないことを示している場合、無効キー条件が発生します。READ文の実行は不成功になります。
2. READ文を実行する前に、ファイル名-1の相対キー項目に、読み込みたいレコードの相対レコード番号を設定しておく必要があります。READ文を実行すると、ファイル位置指示子に相対キー項目の値が設定されます。そして、ファイル位置指示子の値と等しい相対レコード番号を持つレコードが見つかった場合、そのレコードがファイル名-1に関連するレコード領域で使用可能になります。ファイル位置指示子の値と等しい相対レコード番号を持つレコードが見つからなかった場合、無効キー条件が発生し、READ文の実行は不成功になります。

書き方3の規則

1. READ文の実行が開始されるときのファイル位置指示子が、入力の不定ファイルが存在しないことを示している場合、無効キー条件が発生します。READ文の実行は不成功になります。
2. KEY指定を書いた場合、データ名-1がREAD文の参照キーになります。
3. KEY指定を省略した場合、主レコードキーがREAD文の参照キーになります。
4. READ文の実行によって、ファイル位置指示子に参照キーの値が設定されます。そして、ファイルから読み込むレコードを選択するために、レコードの参照キーに対応する部分がファイル位置指示子の値と比較されます。この比較は、参照キーに対応する部分がファイル位置指示子の値と一致する最初のレコードが見つかるまで行われます。主レコードキーまたは副レコードキーが重複した値を持つ場合は、重複した参照キーを持つ一連のレコードの中の最初のレコードが見つかるまで、比較が行われます。
   この比較の結果、条件を満足するレコードが見つかった場合、そのレコードがファイル名-1に関連するレコード領域で使用可能になります。条件を満足するレコードが見つからなかった場合、無効キー条件が発生します。この場合、READ文の実行が不成功になります。
5. READ文の実行が不成功になった場合、参照キーの値は規定されません。

書き方2および書き方3に共通する規則

1. READ文の実行中に無効キー条件が発生すると、ファイル名-1の入出力状態に無効キー条件を示す値が設定された後、“6.3.12 INVALID KEY指定”の規則に従って制御が移ります。
2. READ文の実行中に無効キー条件以外の例外条件が発生すると、ファイル位置指示子およびファイル名-1の入出力状態が設定された後、“6.3.12 INVALID KEY指定”の規則に従って制御が移ります。
3. READ文の実行中に無効キー条件もその他の例外条件も発生しなかった場合、以下の処理が順に行われます。
   a. ファイル位置指示子およびファイル名-1の入出力状態が設定されます。
   b. 読み込んだレコードがレコード領域で使用可能になります。INTO指定を書いた場合、暗黙の転記も実行されます。
   c. “6.3.12 INVALID KEY指定”の規則に従って制御が移ります。

6.4.35 READ文（表示ファイル）

ファイルからレコードを読みます。

【書き方】

READ ファイル名-1 [NEXT] RECORD
[INTO 一意名-1]
[AT END 無条件文-1]

[NOT AT END 無条件文-2]

[END-READ]

構文規則

1. 一意名-1の記憶領域とファイル名-1のレコード領域は、異なる領域でなければなりません。
2. ファイル名-1に関連するUSE AFTER STANDARD EXCEPTION手続きを書かない場合、AT END指定を書かなければなりません。ただし、このコンパイラでは、USE AFTER STANDARD EXCEPTION手続きとAT END指定の両方を省略することができます。
3. 一意名-1に、定数節で定義したデータ項目を指定することはできません。

一般規則

1. ファイル名-1のファイルは、READ文を実行する前に入力モードまたは入出力両用モードで開いておかなければなりません。
2. READ文を実行すると、ファイル名-1の入出力状態の値が更新されます。
3. NEXT指定は省略することができます。NEXT指定は、READ文の実行には影響を与えません。
4. READ文の実行が開始されるとき、ファイル中に次の有効なレコードが存在するかが検査されます。次の有効なレコードが存在すれば、そのレコードがファイル名-1に関連するレコード領域で使用可能になります。次の有効なレコードが存在しない場合、ファイル終了条件が発生します。
5. INTO指定は、以下の場合に書くことができます。
   - ファイル名-1のレコード記述項を1つだけ書いた場合
   - ファイル名-1に関連するすべてのレコード名および一意名-1が、集団項目または英数字項目の場合
6. INTO指定を書いた場合、以下の順に処理が行われます。
   a. INTO指定のない同じREAD文が実行されます。
   b. a.のREAD文の実行が成功した場合、現在のレコードが、CORRESPONDING指定なしのMOVE文の規則に従って、ファイル名-1のレコード領域から一意名-1の領域へ転記されます。読み込まれたレコードは、ファイル名-1のレコード領域と一意名-1のデータ項目の両方で使用可能になります。

   現在のレコードの大きさは、ファイル名-1のRECORD句の規則に従って決定されます。ファイル名-1のファイル記述項にRECORD VARYING句を書いた場合、b.では集団項目転記が行われます。一意名-1に付けた添字は、a.の後b.の前に評価されます。a.のREAD文の実行が不成功の場合、暗黙のMOVE文は実行されません。
7. READ文の実行中にファイル条件が発生すると、READ文の実行が不成功になります。ファイル名-1の入出力状態にファイル終了条件を示す値が設定された後、“6.3.13 AT END指定”の規則に従って制御が移ります。
8. READ文の実行中にファイル終了条件以外の例外条件が発生すると、ファイル名-1の入出力状態が設定された後、“6.3.13 AT END指定”の規則に従って制御が移ります。
9. READ文の実行中にファイル終了条件もその他の例外条件も発生しなかった場合、以下の処理が順に行われます。
   a. ファイル名-1の入出力状態が設定されます。
   b. 読み込んだレコードがレコード領域で使用可能になります。INTO指定を書いた場合、暗黙の転記も実行されます。
   c. “6.3.13 AT END指定”の規則に従って制御が移ります。
10. READ文の実行が不成功になった場合、ファイル名-1のレコード領域の内容は規定されません。
11. 読み込んだレコードの文字位置の個数がファイル名-1のレコード記述項で指定した最小レコードの大きさより小さい場合、ファイル名-1のレコード領域のうち、読み込んだレコードの文字位置の個数を超える部分の内容は規定されません。読み込んだレコードの文字位置の個数がファイル名-1のレコード記述項で指定した最大レコードの大き

さより大きい場合、読み込んだレコードの右側が最大レコードの大きさに合わせて切り捨てられ、ファイル名-1のレコード領域に設定されます。どちらの場合も、READ文は成功します。ファイル名-1の入出力状態には、レコード長矛盾を示す値が設定されます。

12. END-READ指定は、READ文の範囲を区切ります。

6.4.36 RELEASE文（整列併合）

整列操作の最初の段階にレコードを引き渡します。

【書き方】

RELEASE レコード名-1 [FROM 一意名-1]

構文規則

1. RELEASE文は、レコード名-1のファイルに対するSORT文で指定した入力手続きの中にだけ書くことができます。
2. レコード名-1は、整列併合用ファイル記述項に関連するレコード記述項で定義しなければなりません。
3. レコード名-1は、ファイル名で修飾することができます。
4. 一意名-1にデータ項目を指定する場合、レコード名-1の記憶領域と一意名-1の記憶領域は、同じであってはいけません。一意名-1に関数一意名を指定する場合、一意名-1は英数字関数でなければなりません。

一般規則

1. RELEASE文は、レコード名-1のレコードを整列操作の最初の段階に引き渡します。
2. RELEASE文を実行すると、レコード名-1のレコードはレコード領域中で使用不可能になります。ただし、入出力管理段落のSAME RECORD AREA句に、レコード名-1に関連するファイルを指定した場合、レコード名-1のレコードは、レコード領域中で使用不可能になりません。そのレコードは、同じSAME RECORD AREA句に指定した他のファイルの出力レコードとして使用可能です。
3. FROM指定付きのRELEASE文の実行結果は、以下の順に2つの文を実行した結果と同じです。
    a. MOVE 一意名-1 TO レコード名-1
    b. FROM指定のない同じRELEASE文

6.4.37 RETURN文（整列併合）

整列または併合操作の最後の段階から、整列または併合されたレコードを引き取ります。

【書き方】

RETURN ファイル名-1 RECORD [INTO 一意名-1]
AT END 無条件文-1
[NOT AT END 無条件文-2]
[END-RETURN]

構文規則

1. RETURN文は、ファイル名-1のファイルに対するSORT文またはMERGE文に指定した出力手続きの中にだけ書くことができます。
2. ファイル名-1は、整列併合用ファイル記述項で定義しなければなりません。
3. 一意名-1の記憶領域とファイル名-1のレコード領域は、同じであってはいけません。
4. 一意名-1が強く型付けされた集団項目である場合、ファイル名-1のレコード記述項には、同じ型で強く型付けされた1つの集団項目だけを指定できます。

一般規則

1. RETURN文を実行すると、ファイル名-1のファイルからSORT文またはMERGE文に指定したキーの並びによって決められた次のレコードが引き取られます。次のレコードがない場合、ファイル終了条件が発生します。
2. ファイル名-1のレコード記述項に2つ以上のレコードを定義した場合、それらのレコードは再定義の規則に従って同じ記憶領域を共用します。
3. INTO指定は、以下の場合に書くことができます。
   - ファイル名-1のレコード記述項を1つだけ書いた場合
   - ファイル名-1に関連するレコード名および一意名-1のデータ項目が、すべて集団項目または英数字項目の場合
4. INTO指定を書いた場合、以下の順に処理が行われます。
   a. INTO指定のない同じRETURN文が実行されます。
   b. a.のRETURN文の実行が成功した場合、現在のレコードが、CORRESPONDING指定なしのMOVE文の規則に従って、ファイル名-1のレコード領域から一意名-1の領域に転記されます。読み込まれたレコードは、ファイル名-1のレコード領域と一意名-1のデータ項目の両方で使用可能になります。

   現在のレコードの大きさは、ファイル名-1のRECORD句の規則に従って決定されます。ファイル名-1のファイル記述項にRECORD VARYING句を書いた場合、b.では集団項目転記が行われます。一意名-1に付けた添字は、a.の後b.の前に評価されます。a.のRETURN文の実行が不成功の場合、暗黙のMOVE文は実行されません。
5. ファイル終了条件が発生した場合、RETURN文の実行は不成功になり、ファイル名-1のレコード領域の内容は不定になります。そして、無条件文-1に制御が移り、無条件文-1の実行後、RETURN文の最後に制御が移ります。ただし、無条件文-1で制御の明示移行を起こす手続き分岐または条件文を実行した場合、その文の規則に従って制御が移ります。無条件文-1の実行後は、実行中の出力手続きの一部として、RETURN文を実行することはできません。
6. RETURN文の実行中にファイル終了条件が発生しなかった場合、以下の処理が順に行われます。
   a. 引き取られたレコードが、ファイル名-1のレコード領域で使用可能になります。INTO指定を書いた場合、ファイル名-1のレコード領域から一意名-1へレコードが転記されます。
   b. NOT AT END指定を書いた場合、無条件文-2に制御が移ります。そして、無条件文-2の実行後、RETURN文の最後に制御が移ります。ただし、無条件文-2で制御の明示移行を起こす手続き分岐または条件文を実行した場合、その文の規則に従って制御が移ります。NOT AT END指定を省略した場合は、RETURN文の最後に制御が移ります。
7. END-RETURN指定は、RETURN文の範囲を区切ります。

6.4.38 REWRITE文（順ファイル・相対ファイル・索引ファイル）

大記憶ファイル中のレコードを論理的に書き換えます。

【書き方1】順に読んだレコードを書き換える（順ファイル・相対ファイル）

REWRITE レコード名-1 [FROM 一意名-1]

[END-REWRITE]

【書き方2】指定したレコードを書き換える（相対ファイル・索引ファイル）

REWRITE レコード名-1
[FROM 一意名-1]
[INVALID KEY 無条件文-1]
[NOT INVALID KEY 無条件文-2]
[END-REWRITE]

構文規則

各ファイルに共通する規則

1. レコード名-1は、データ部のファイル節で定義したレコードの名前でなければなりません。レコード名-1は、ファイル名で修飾することができます。
2. 一意名-1にデータ項目を指定する場合、レコード名-1の領域と一意名-1の領域は同じであってはいけません。一意名-1に関数一意名を指定する場合、英数字関数でなければなりません。

相対ファイルの規則

1. レコード名-1に順呼出し法のファイルを指定する場合、書き方1のREWRITE文を書かなければなりません。
2. レコード名-1に乱呼出し法または動的呼出し法のファイルを指定する場合、書き方2のREWRITE文を書かなければなりません。この場合、関連するUSE AFTER STANDARD EXCEPTION手続きを書かないときは、INVALID KEY指定を書かなければなりません。

   ただし、このコンパイラでは、USE AFTER STANDARD EXCEPTION手続きとINVALID KEY指定の両方を省略することができます。

索引ファイルの規則

1. レコード名-1に関連するファイルは、DUPLICATES指定付きのRECORD KEY句を指定した乱呼出し法のファイルであってはいけません。
2. レコード名-1に関連するファイルに関連するUSE AFTER STANDARD EXCEPTION手続きを書かない場合、INVALID KEY指定を書かなければなりません。

   ただし、このコンパイラでは、関連するUSE AFTER STANDARD EXCEPTION手続きとINVALID KEY指定の両方を省略することができます。

一般規則

各ファイルに共通する規則

1. レコード名-1に関連するファイルは、大記憶ファイルでなければなりません。
2. レコード名-1に関連するファイルは、REWRITE文を実行する前に入出力両用モードで開いておかなければなりません。
3. REWRITE文の実行が成功すると、レコード名-1のレコードはレコード領域で使用不可能になります。ただし、入出力管理段落のSAME RECORD AREA句にレコード名-1に関連するファイルを指定した場合、REWRITE文によってレコードが使用不可能になることはありません。そのレコードは、そのプログラムで使用可能です。
4. FROM指定を書いたREWRITE文の実行結果は、以下に示す文をこの順で実行した結果と同じです。
   a. MOVE 一意名-1 TO レコード名-1
   b. FROM指定のない同じREWRITE文

5. REWRITE文の実行が完了した後、SAME RECORD AREA句にレコード名-1に関連するファイルを指定している場合を除いて、レコード名-1の領域の情報は使用不可能になります。ただし、一意名-1の領域の情報は使用可能です。
6. REWRITE文を実行しても、ファイル位置指示子は変更されません。
7. REWRITE文を実行すると、レコード名-1に関連するファイルの入出力状態の値が更新されます。
8. REWRITE文の実行によって、レコードが入出力管理システムに渡されます。
9. 他のファイル結合子によってロックされているレコードを書き換えようとすると、REWRITE文の実行は不成功になります。そして、レコードのロックを示す値が、レコード名-1に関連する入出力状態に設定されます。
10. レコード名-1に関連するファイルのLOCK MODE句にAUTOMATICを指定した場合、REWRITE文の実行が成功すると、存在するレコードのロックが解除されます。
11. END-REWRITE指定は、REWRITE文の範囲を区切ります。

順ファイルの規則

1. REWRITE文を実行する前に、レコード名-1に関連するファイルに対してREAD文を実行しなければなりません。READ文の実行が成功した場合、REWRITE文を実行すると、READ文によって読み込まれたレコードが論理的に書き換えられます。
2. レコード名-1のレコードの文字位置の個数と書き換えられるレコードの長さが異なる場合、REWRITE文の実行は不成功になり、レコードは書き換えられません。レコード名-1のレコード領域の内容は変更されません。このとき、以下の処理が順に行われます。
    a. レコード名-1に関連するファイルの入出力状態に、この状態の原因を示す値が設定されます。
    b. レコード名-1に関連するファイルのUSE AFTER STANDARD EXCEPTION手続きを書いた場合、その手続きが実行されます。そして、その手続きの実行後、USE文の規則に従って制御が渡されます。USE AFTER STANDARD EXCEPTION手続きを書かなかった場合、レコード名-1に関連するファイルにFILE STATUS句を指定したときは、REWRITE文の最後に制御が移ります。USE AFTER STANDARD EXCEPTION手続きもFILE STATUS句も書かなかったときの実行処理は、規定されません。

相対ファイルの規則

1. 順呼出し法のファイルに対するREWRITE文(書き方1のREWRITE文)の場合、REWRITE文を実行する前に、レコード名-1に関連するファイルに対してREAD文を実行しなければなりません。READ文の実行が成功した場合、REWRITE文を実行すると、READ文によって読み込まれたレコードが書き換えられます。
2. 乱呼出し法または動的呼出し法のファイルに対してREWRITE文(書き方2のREWRITE文)を実行すると、相対キー項目の値と同じ相対レコード番号を持つレコードが書き換えられます。相対キー項目の値と同じ相対レコード番号を持つレコードがファイルに存在しない場合は、無効キー条件が発生します。

索引ファイルの規則

1. 以下の場合、REWRITE文を実行する前に実行される最後の入出力文は、レコード名-1に関連するファイルに対するREAD文でなければなりません。READ文の実行が成功した場合、REWRITE文を実行すると、READ文によって読み込まれたレコードが書き換えられます。
    - 順呼出し法のファイルに対するREWRITE文
    - DUPLICATES指定付きのRECORD KEY句を指定したファイルに対するREWRITE文
    - REVERSED指定付きのSTART文によりファイル位置指示子が確定したファイルに対するREWRITE文
2. 順呼出し法のファイルに対するREWRITE文の場合、書き換えるレコードを主レコードキーの値で指定します。REWRITE文を実行するとき、書き換えるレコードの主レコードキーの値は、このファイルから最後に読み込まれたレコードの主レコードキーの値と等しくなければなりません。
3. 各DUPLICATES指定付きのRECORD KEY句を指定したファイルまたはREVERSED指定付きのSTART文によりファイル位置指示子が確定したファイルに対するREWRITE文の場合、書き換えるレコードを主レコードキーで指定します。REWRITE文を実行するとき、書き換えるレコードのレコードキーの値は、このファイルから最後に読み込まれたレコードのレコードキーの値と等しくなければなりません。
4. 乱呼出し法または動的呼出し法のファイルの場合、書き換えるレコードを主レコードキーの値で指定します。

5. レコード名-1に関連するファイルにALTERNATE RECORD KEY句を指定した場合、REWRITE文の実行は、副レコードキーの値により以下のように行われます。

   ― 副レコードキーの値が変わらないとき、そのキーが参照キーであっても、レコードの検索の順序は変わりません。

   ― 副レコードキーの値が変わるとき、その副レコードキーが参照キーであれば、レコードの検索の順序が変わることがあります。重複したキー値が許されているとき、そのレコードは、設定された副レコードキーの値と同じ副レコードキーの値を持つレコードの組の中の論理的に最後の位置を占めます。

6. 以下の場合、無効キー条件が発生します。

   ― 以下の条件を満足するREWRITE文で、書き換えようとするレコードの主レコードキーの値が、同じファイルから最後に読み込まれたレコードの主レコードキーの値と等しくない場合

   - 順呼出し法のファイルに対するREWRITE文
   - DUPLICATES指定付きのRECORD KEY句を指定したファイルに対するREWRITE文
   - REVERSED ORDER指定付きのSTART文によりファイル位置指示子が確定したファイルに対するREWRITE文

   ― 動的呼出し法または乱呼出し法のファイルに対するREWRITE文で、書き換えようとするレコードの主レコードキーの値に等しいレコードが、ファイル中に存在しない場合

   ― DUPLICATES指定なしのALTERNATE RECORD KEY句を指定したファイルに対するREWRITE文で、書き換えようとするレコードの副レコードキーの値と等しい副レコードキーの値を持つレコードが、ファイル中にすでに存在する場合

相対ファイルおよび索引ファイルに共通する規則

1. レコード名-1のレコードの文字位置の個数は、書き換えられるレコードの長さと同じである必要はありません。
2. レコード名-1のレコードの文字位置の個数は、レコード名-1に関連するファイルの最大レコード長より大きくてはいけません。この場合、REWRITE文の実行は不成功になり、レコードは書き換えられません。レコード名-1に関連するファイルのレコード領域の内容は変更されません。レコード名-1に関連するファイルの入出力状態には、この状態の原因を示す値が設定されます。
3. REWRITE文の実行中に無効キー条件が発生すると、REWRITE文の実行は不成功になり、レコードは書き換えられません。レコード名-1に関連するファイルの入出力状態に無効キー条件を示す値が設定された後、“6.3.12 INVALID KEY指定”の規則に従って制御が移ります。
4. REWRITE文の実行中に無効キー条件以外の例外条件が発生すると、レコード名-1に関連するファイルの入出力状態が設定された後、“6.3.12 INVALID KEY指定”の規則に従って制御が移ります。
5. REWRITE文の実行中に無効キー条件もその他の例外条件も発生しなかった場合、以下の処理が順に行われます。

   ― レコード名-1に関連するファイルの入出力状態が設定されます。

   ― READ文によって読み込まれたレコードが書き換えられます。

   ― “6.3.12 INVALID KEY指定”の規則に従って制御が移ります。

6.4.39 SEARCH文（中核）

表要素を検索します。

使い方の例については、“付録H サンプル集”の“H.40 SEARCH文”を参照してください。

【書き方1】表要素を順に検索する

[AT END 無条件文-1]

{ WHEN 条件名-1 { 無条件文-2 / NEXT SENTENCE } } …

[END-SEARCH]

【書き方2】昇順または降順に並んでいる表要素を非逐次に検索する

SEARCH ALL 一意名-1 [AT END 無条件文-1]

WHEN { データ名-1 / 条件名-1 } { IS EQUAL TO / IS = } { 一意名-3 / 定数-1 / 算術式-1 }

[AND { データ名-2 / 条件名-2 } { IS EQUAL TO / IS = } { 一意名-4 / 定数-2 / 算術式-2 }] …

{ 無条件文-2 / NEXT SENTENCE }

[END-SEARCH]

注意

“＝”は必要語です。ここでは、他の記号との混同を避けるために下線を付けていません。

構文規則

書き方1の規則

1. 一意名-1に、添字または部分参照子を付けてはいけません。一意名-1は、INDEXED BY指定付きのOCCURS句を指定したデータ項目でなければなりません。
2. 一意名-2は、指標データ項目または整数項目でなければなりません。一意名-1のOCCURS句のINDEXED BY指定に書いた指標名のうち、最初または唯一の指標名を使って、一意名-2を添字付けしてはいけません。

書き方2の規則

1. 一意名-1は、INDEXED BY指定とKEY IS指定の両方を持つOCCURS句を指定したデータ項目でなければなりません。一意名-1に、添字または部分参照子を付けてはいけません。
2. データ名-1およびデータ名-2は、一意名-1のOCCURS句のKEY IS指定に書いたデータ名でなければなりません。
3. 条件名-1および条件名-2に対する条件変数は、一意名-1のOCCURS句のKEY IS指定に書いたデータ名でなければなりません。条件名-1および条件名-2のデータ記述項のVALUE句に、THRU指定を書くことはできません。
4. データ名-1、データ名-2、条件名-1または条件名-2に添字を付ける場合、1つの添字は、一意名-1のOCCURS句のINDEXED BY指定の中の最初の指標名でなければなりません。
5. 一意名-3、一意名-4、算術式-1の中の一意名、および算術式-2の中の一意名に、一意名-1のOCCURS句のKEY IS指定に書いたデータ名を指定することはできません。これらを添字付けしなければならない場合、添字は、一意名-1のOCCURS句のINDEXED BY指定の中の最初の指標名であってはいけません。
6. 一意名-1のOCCURS句のKEY IS指定に2つ以上のデータ名を書いた場合、WHEN指定のデータ名または条件名には、KEY IS指定の中の最初のデータ名から第n番目(最後でなくてもよい)までのすべてのデータ名、または

それらのデータ名に関係付けたすべての条件名を書かなければなりません。データ名と条件名は混在して書くこともできます。

7. データ名-1、データ名-2、一意名-3、一意名-4、定数-1および定数-2は、ブール項目であってはいけません。

書き方1および書き方2に共通する規則

NEXT SENTENCE指定とEND-SEARCH指定の両方を書いてはいけません。

例

書き方2のSEARCH文の例

データ部で以下に示す表を定義した場合、手続き部では(a)および(b)のSEARCH文を書くことができます。

```
01  TBL.
    02  Y OCCURS 50 TIMES
          ASCENDING KEY IS K1, K2, K3, K4
          INDEXED BY  IX1.
        03  K1 PIC 9.
            88  T1 VALUE 3.
        03  K2 PIC 9.
            88  T2 VALUE 2.
        03  K3 PIC 9.
            88  T3 VALUE 1.
        03  K4 PIC 9.
77  X PIC 9.
```

```
   SEARCH ALL Y WHEN K1(IX1) = 3     …  (a)
                 AND K2(IX1) = 2
                 AND K3(IX1) = 1
       :
   END-SEARCH
   SEARCH ALL Y WHEN T1(IX1)         …  (b)
                 AND T2(IX1)
                 AND T3(IX1)
                 AND K4(IX1) = X
       :
   END-SEARCH
```

一般規則

書き方1の規則

1. 書き方1のSEARCH文は、一意名-1の表を表引き用の指標名が示す表要素から順に検索し、条件-1を満足する表要素が見つかった場合、条件-1に対応する無条件文-2の処理を実行します。
2. 表要素の検索には、以下の指標名が使われます。この指標名を「表引き用の指標名」といいます。
   - 以下の場合、一意名-1のOCCURS句のINDEXED BY指定の中で最初に書いた指標名が使われます。
     - VARYING指定を省略した場合
     - VARYING指定に一意名-1の表以外の表の指標名を書いた場合
     - VARYING指定に一意名-2を書いた場合
   - VARYING指定に一意名-1のOCCURS句のINDEXED BY指定に書いた指標名を書いた場合、その指標名が使われます。
3. 表要素の検索は、以下の手順で行われます。
   - 表引き用の指標名の値が一意名-1の最大の出現番号に対応する値より大きい場合、表の検索は行われません。このとき、AT END指定を書いた場合は無条件文-1に制御が移り、AT END指定を省略した場合はSEARCH文の終わりに制御が移ります。

ー 表引き用の指標名の値が一意名-1の最大の出現番号に対応する値以下の場合、以下の処理が行われます。

1. 表引き用の指標名の値が示す表要素が条件-1を満足するかどうかが検査されます。検査は、WHEN指定の並びを書いた順に行われます。
2. どの条件も満足しなかった場合、表引き用の指標名の値が増やされます。そして、[1]の処理が繰り返されます。
3. 表引き用の指標名の新しい値が表の範囲外の出現番号になった場合、a.と同じ方法でSEARCH文の処理が終了します。
4. 条件の1つを満足した場合、表要素の検査が終了します。そして、無条件文-2を書いた場合は無条件文-2に制御が移り、NEXT SENTENCEを書いた場合は、次の実行完結文に制御が移ります。このとき、表引き用の指標名は、条件を満足したときの出現番号に対応する値になっています。

4. 表引き用の指標名、一意名-2および指標名-1の値は、以下の規則に従って増加します。

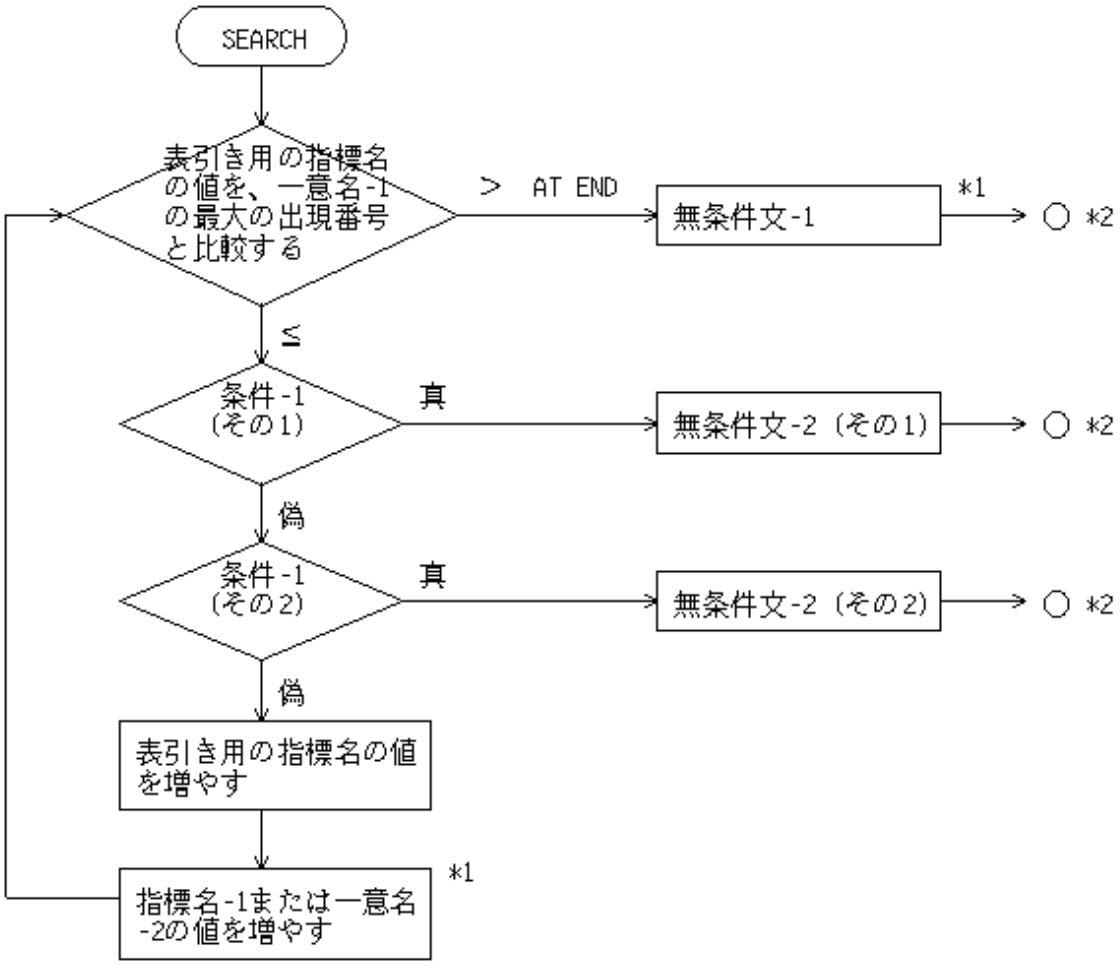
   a. VARYING指定を省略した場合、またはVARYING指定に一意名-1のOCCURS句のINDEXED BY指定に書いた指標名を書いた場合、表引き用の指標名だけが増加します。

   b. VARYING指定の指標名-1に一意名-1の表以外の表の指標名を書いた場合、表引き用の指標名が増加するごとに、その増分値が示す出現番号に対応する値だけ、指標名-1の値が増加します。

   c. VARYING指定に指標データ項目を書いた場合、表引き用の指標名が増加するごとに、その増分値だけ、指標データ項目の値が増加します。

   d. VARYING指定に整数項目を書いた場合、表引き用の指標名が増加するごとに、1だけ整数項目の値が増加します。

5. WHEN指定を2つ書いた場合、表要素の検査は以下の流れ図に従って行われます。



*1 ： これらの操作は、SEARCH文に書いた場合だけ行われます。
*2 ： 無条件文の実行後、SEARCH分の終わりに制御が移ります。ただし、無条件文の中に制御の明示移行を起こす手続き分岐文を書いた場合、その文の規則に従って制御が移ります。

書き方2の規則

1. 書き方2のSEARCH文は、一意名-1の表の中からWHEN指定の条件を満足する表要素を非逐次に検索し、条件を満足する表要素が見つかった場合、無条件文-2を実行します。
2. 表要素の検索には、一意名-1のOCCURS句のINDEXED BY指定の中で最初に書いた指標名が使われます。この指標名を「表引き用の指標名」といいます。表引き用の指標名以外の指標名の値は、変更されません。
3. 書き方2のSEARCH文の実行結果は、以下のすべての条件を満足する場合だけ規定されます。
   a. 一意名-1の表の中のデータが、一意名-1のOCCURS句のKEY IS指定の記述に従って、昇順または降順に並んでいる。
   b. データ名-1およびデータ名-2、または条件名-1および条件名-2の値が、表要素を一意に識別できる値である。
4. 表要素の検索は、表引き用の指標名の値とは関係なく、以下の手順で行われます。
   a. WHEN指定に書いたすべての条件を満足する表要素が検索されます。
   b. WHEN指定に書いたすべての条件を満足する表要素が見つからなかった場合、AT END指定を書いたときは無条件文-1に制御が移り、AT END指定を省略したときはSEARCH文の終わりに制御が移ります。表引き用の指標名の値は規定されません。
   c. WHEN指定に書いたすべての条件を満足する表要素が見つかった場合、無条件文-2を書いたときは無条件文-2に制御が移り、NEXT SENTENCEを書いたときは次の実行完結文に制御が移ります。表引き用の指標名には、条件を満足する表要素の出現番号に対応する値が設定されます。

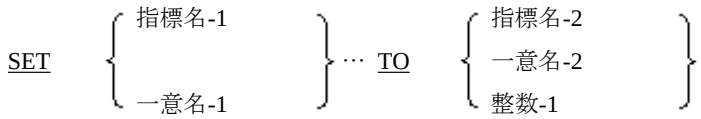
書き方1および書き方2に共通する規則

1. SEARCH文の終わりは、以下のいずれかの方法で指定します。
   ― END-SEARCH指定
   ― 分離符の終止符
   ― IF文の無条件文の中にSEARCH文を書いた場合、IF文に対応するELSE指定またはEND-IF指定
   IF文の範囲については、“6.1 手続き部の構成（PROCEDURE DIVISION）”の“文の範囲”を参照してください。
2. 無条件文-1または無条件文-2の実行後、SEARCH文の終わりに制御が移ります。ただし、無条件文の中に制御の明示移行を起こす手続き分岐文を書いた場合、その文の規則に従って制御が移ります。
3. END-SEARCH指定は、SEARCH文の範囲を区切ります。

6.4.40 SET文（中核）

表要素の指標の設定、外部スイッチの状態の設定、条件変数の値の設定を行います。

【書き方1】表要素の指標を設定する



【書き方2】指標名の値を増減する

```
SET {指標名-3} …    { UP  BY   }  { 一意名-3 }
                    { DOWN BY  }  { 整数-2   }
```

【書き方3】外部スイッチの状態を設定する

```
SET    { {呼び名-1} … TO … { ON  } }
                           { OFF }
```

【書き方4】条件変数の値を設定する

```
SET {条件名-1} … TO TRUE
```

構文規則

1. 一意名-1および一意名-2は、指標データ項目または整数項目でなければなりません。
2. 一意名-3は、整数項目でなければなりません。
3. 整数-1には、正号付きのゼロ以上の整数を指定することもできます。整数-2には、正号付きのゼロ以上の整数または負号付きの整数を指定することもできます。
4. 呼び名-1は、環境部の特殊名段落で、SWITCH-0～SWITCH-7またはSWITCH-1～SWITCH-8に関係付けなければなりません。

一般規則

書き方1の規則

1. 書き方1のSET文は、TOの後の作用対象が示す出現番号に対応する値を、TOの前の各作用対象に設定します。TOの後の作用対象を「送出し側作用対象」、TOの前の各作用対象を「受取り側作用対象」といいます。
2. 書き方1のSET文における作用対象の組合せの可否を下表に示します。

送出し側作用対象		受取り側作用対象		
		指標名-1	一意名-1	
			指標データ項目	整数項目
指標名-2		○ (5)(a)	○ (5)(b)	○ (5)(c)
一意名-2	指標データ項目	○ (5)(a)	○ (5)(b)	－
	整数項目	○ (5)(a)	○ (5)(b)	－
整数-1		○ (5)(a)	－	－

○ ： 組合せができる
－ ： 組合せができない
(n) ： 一般規則の番号

3. 指標名-1を書く場合、TOの後の作用対象の値は、指標名-1を関係付けた表の出現番号に対応しなければなりません。
4. 指標名-2を書く場合、指標名-2の値は、指標名-1を関係付けた表の出現番号に対応しなければなりません。

5. 指標の設定は、以下の規則に従って行われます。

   a. 指標名-1を書いた場合、指標名-2、一意名-2または整数-1の値に対応する出現番号の値が、指標名-1に設定されます。一意名-2に指標データ項目を指定した場合、または指標名-2に指標名-1と同じ表に関係付けた指標名を指定した場合は、出現番号の値の変換は行われません。

   b. 一意名-1に指標データ項目を指定した場合、指標名-2または一意名-2の内容と同じ値が、一意名-1に変換されずに設定されます。

   c. 一意名-1に整数項目を指定した場合、指標名-2の値に対応する出現番号の値が一意名-1に、設定されます。

6. 受取り側作用対象を2つ以上書いた場合、指標の設定が繰り返されます。繰返しにおいて、指標名-2および一意名-2の値として、SET文を実行する前の値が使われます。一意名-1に添字を付けた場合、添字は繰返しごとに1つずつ順番に評価されます。

書き方2の規則

1. 書き方2のSET文は、UP BYの前の各作用対象の値を、UP BYの後の作用対象が示す出現番号に対応する値だけ増やすか、またはDOWN BYの前の各作用対象の値を、DOWN BYの後の作用対象が示す出現番号に対応する値だけ減らします。
2. SET文の実行前および実行後の指標名-3の値は、指標名-3を関係付けた表の出現番号に対応しなければなりません。
3. 指標名-3を2つ以上書いた場合、指標名の値の増減が繰り返されます。繰返しにおいて、一意名-3の値として、SET文を実行する前の値が使われます。

書き方3の規則

1. 書き方3のSET文は、呼び名-1に対応付けた外部スイッチの状態を、オン状態またはオフ状態にします。呼び名-1の並びを書いた順に、この処理を繰り返します。
2. 呼び名-1に関係付けた外部スイッチの状態は、以下のように設定されます。

   ー ON指定を書いた場合、外部スイッチに関係付けた条件名を評価したときの真理値の結果がオン状態になるように、外部スイッチの状態が設定されます。

   ー OFF指定を書いた場合、外部スイッチに関係付けた条件名を評価したときの真理値の結果がオフ状態になるように、外部スイッチの状態が設定されます。

書き方4の規則

1. 書き方4のSET文は、条件名-1の条件名記述項に書いたVALUE句の値を、条件名-1に関係付けた条件変数に設定します。条件名-1の並びを書いた順に、この処理を繰り返します。
2. 条件名-1のVALUE句に2つ以上の定数を書いた場合、VALUE句に書いた最初の定数の値が、条件変数に設定されます。

6.4.41 SORT文（整列併合）

キー項目の値の順にレコードを整列します。

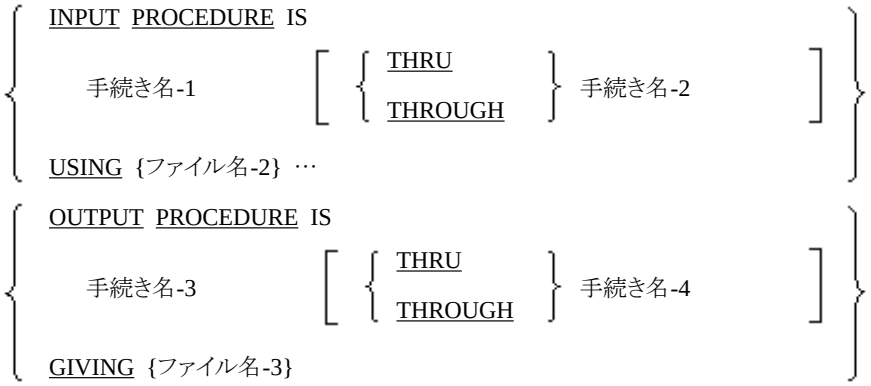
【書き方】

SORT ファイル名-1

{ ON { ASCENDING / DESCENDING } KEY {データ名-1} … } …

[WITH DUPLICATES IN ORDER]

[COLLATING SEQUENCE IS 符号系名-1]



構文規則

1. SORT文は、宣言部分を除く手続き部のどこにでも書くことができます。
2. ファイル名-1は、データ部の整列併合用ファイル記述項で定義しなければなりません。
3. ファイル名-2は、以下の規則に従わなければなりません。
   a. ファイル名-1のレコード形式が可変長の場合、ファイル名-2のレコードの長さは、ファイル名-1の最小レコード長以上かつ最大レコード長以下でなければなりません。
   b. ファイル名-1のレコード形式が固定長の場合、ファイル名-2のレコードの長さは、ファイル名-1のレコード長以下でなければなりません。
4. データ名-1は、以下の規則に従わなければなりません。
   a. データ名-1は、ファイル名-1のレコード記述項で定義したデータ項目でなければなりません。
   b. データ名-1は、修飾することができます。
   c. データ名-1に、可変反復データ項目を従属する集団項目を指定することはできません。
   d. ファイル名-1に2つ以上のレコード記述項を書いた場合、データ名-1の並びには、1つのレコード記述項に書いたデータ項目を指定しなければなりません。1つのレコード記述項中のデータ名-1のデータ項目と同じ文字位置が、そのファイルのすべてのレコード中のキーとみなされます。
   e. データ名-1に、OCCURS句を指定したデータ項目またはOCCURS句を指定したデータ項目に従属するデータ項目を指定することはできません。
   f. ファイル名-1のレコード形式が可変長の場合、データ名-1のデータ項目は、ファイル名-1の最小レコードの大きさの範囲内に含まれていなければなりません。
   g. データ名-1は、ブール項目であってはいけません。
   h. データ名-1の個数の最大については、“付録B システムの定量制限”を参照してください。
5. ファイル名-2およびファイル名-3は、データ部の整列併合用ファイル以外のファイル記述項で定義しなければなりません。
6. ファイル名-2およびファイル名-3に、同一の複数ファイルリール上のファイルを指定することもできます。
7. ファイル名-1～ファイル名-3のいくつかの組を、1つのSAME AREA句または1つのSAME SORT/SORT-MERGE AREA句に指定することはできません。また、ファイル名-3のいくつかの組を、1つのSAME句に指定することはできません。
8. THROUGHとTHRUは同義語です。
9. ファイル名-3に索引ファイルを指定する場合、以下の規則に従わなければなりません。
   a. 最初のKEY指定は、ASCENDING KEY指定でなければなりません。

b. 最初のデータ名-1に指定したデータ項目のレコード内文字位置は、索引ファイルの主レコードキーに関連付けたデータ項目のレコード内文字位置と同じでなければなりません。また、2つのデータ項目は同じ長さでなければなりません。

10. ファイル名-3は、以下の規則に従わなければなりません。

    a. ファイル名-3のレコード形式が可変長の場合、ファイル名-1のレコードの長さは、ファイル名-3の最小レコード長以上かつ最大レコード長以下でなければなりません。

    b. ファイル名-3のレコード形式が固定長の場合、ファイル名-1のレコードの長さは、ファイル名-3のレコード長以下でなければなりません。

11. 符号系名-1は、特殊名段落のALPHABET句で機能名に対応付けた符号系名であってはいけません。

12. ファイル名-1～ファイル名-3のレコード記述項に、CHARACTER TYPE句またはPRINTING POSITION句を書くことはできません。

13. ファイル名-2およびファイル名-3は、以下のファイルでなければなりません。

    a. 大記憶装置またはフロッピーディスク装置の順ファイル

    b. 順呼出し法の相対ファイル

    c. 順呼出し法の索引ファイル

14. ファイル名-2に指定できるファイルの個数の最大については、“付録B システムの定量制限”を参照してください。

一般規則

SORT文の全般規則

1. SORT文を実行すると、以下の処理が順に行われます。

    a. 以下の処理によって、レコードが整列操作に引き渡されます。ここでの処理を、「整列操作の最初の段階」といいます。
    INPUT PROCEDURE指定を書いた場合、入力手続きが実行されます。「入力手続き」とは、手続き名-1に書いた手続き、または手続き名-1から手続き名-2までに書いた手続きのことです。入力手続きの中でファイル名-1のレコードに対するRELEASE文を実行すると、レコードが整列操作に引き渡されます。
    USING指定を書いた場合、ファイル名-2に対してREAD文が実行され、READ文によって読み込まれたレコードが、整列操作に引き渡されます。この段階を始める前に、ファイル名-2のファイルは開いた状態であってはいけません。この段階の中で、ファイル名-2のファイルは自動的にオープンされ、クローズされます。

    b. ファイル名-1のレコードが、KEY指定で指定した順に整列されます。ここでの処理を、「整列操作」といいます。この段階の中で、ファイル名-2およびファイル名-3のファイルが処理されることはありません。

    c. 以下の処理によって、整列されたレコードがファイル名-1のレコードとして使用可能になります。ここでの処理を、「整列操作の最後の段階」といいます。
    OUTPUT PROCEDURE指定を書いた場合、出力手続きが実行されます。「出力手続き」とは、手続き名-3に書いた手続き、または手続き名-3から手続き名-4までに書いた手続きのことです。出力手続きの中でファイル名-1のファイルに対するRETURN文を実行すると、ファイル名-1のレコードが使用可能になります。
    GIVING指定を書いた場合、整列されたレコードがファイル名-1のレコードとて使用可能になり、ファイル名-3のファイルに書き出されます。この段階を始める前に、ファイル名-3のファイルは、開いた状態であってはいけません。この段階の中で、ファイル名-3のファイルは自動的にオープンされ、クローズされます。

2. データ名-1のデータ項目を、「キー項目」といいます。キー項目を2つ以上指定する場合、キーの強さの順に左から右へ並べます。

    ― ASCENDING指定を書いた場合

    整列操作に引き渡されたレコードのキー項目が比較の規則に従って比較され、キー項目の値が小さいレコードから大きいレコードの順に並べ替えられます。

    ― DESCENDING指定を書いた場合

    整列操作に引き渡されたレコードのキー項目が比較の規則に従って比較され、キー項目の値が大きいレコードから小さいレコードの順に並べ替えられます。

3. 2.の比較の結果、すべてのキー項目の内容が同じであるレコードが2つ以上存在する場合には、整列されたレコードが以下の順に引き取られます。

   ー DUPLICATES指定を書いた場合

   USING指定を書いた場合、ファイル名-2の並びを書いた順に各ファイル中のレコードが引き取られます。1つのファイル名-2のファイル中にそのような複数のレコードが存在する場合、レコードが読み込まれた順に、レコードが引き取られます。

   INPUT PROCEDURE指定を書いた場合、入力手続きで引き渡された順にレコードが引き取られます。

   ー DUPLICATES指定を省略した場合

   レコードを引き取る順序は、規定されません。

4. 文字比較の規則が適用される場合、キー項目の値の大小順序は、以下の規則に従って決定されます。

   ー COLLATING SEQUENCE指定を書いた場合

   符号系名-1に対応付けた文字の大小順序に従います。

   ー COLLATING SEQUENCE指定を省略した場合

   見出し部の翻訳用計算機段落にPROGRAM COLLATING SEQUENCE句を書いた場合は、その句で指定した文字の大小順序に従います。PROGRAM COLLATING SEQUENCE句を省略した場合は、計算機固有の文字の大小順序に従います。

INPUT PROCEDURE指定の規則

1. INPUT PROCEDURE指定を書いた場合、整列操作が開始される前に、入力手続きに制御が移ります。入力手続きの最後の文を実行した後、整列操作が開始されます。
2. 入力手続きの中には、ファイル名-1のレコードに対するRELEASE文を書きます。RELEASE文は、整列操作の最初の段階に、1つのレコードを引き渡します。
3. 入力手続きの中で実行される文を、「入力手続きの範囲」といいます。入力手続きの範囲は、以下のとおりです。

   ー 入力手続きの中に書いた文

   ー 入力手続きの中に書いた文(たとえばCALL文など)によって制御が移行した結果、実行されるすべての文

   ー 入力手続きの中に書いた文によって実行される宣言手続きの中のすべての文

4. 入力手続きの範囲で、MERGE文、RETURN文またはSORT文を実行することはできません。
5. 入力手続きの範囲で、SORT文と同じ手続き部内に書かれたEXIT PROGRAM文を実行することはできません。

USING指定の規則

1. USING指定を書いた場合、ファイル名-2のファイル中のすべてのレコードが、ファイル名-1のファイルに移されます。
2. ファイル名-2の各ファイルに対して、以下の処理が順に行われます。

   a. 初期処理が行われます。これは、ファイル名-2のファイルに対して、INPUT指定のOPEN文を実行したことと同じです。

   b. ファイル名-2のレコードが整列操作に引き渡されます。これは、ファイル名-2のファイルに対して、NEXT指定およびAT END指定付きのREAD文を、レコードがなくなるまで実行したことと同じです。ファイル名-2に相対ファイルを指定した場合、SORT文を実行した後の相対キー項目の内容は規定されません。

   c. 終了処理が行われます。これは、ファイル名-2だけを指定したCLOSE文を実行したことと同じです。この処理は、整列操作が開始される前に行われます。

   上記のa.～c.の処理の中で、ファイル名-2に関連するUSE AFTER STANDARD EXCEPTION手続きが実行されることがあります。そのUSE手続きの中では、ファイル名-2のファイルを操作する文またはファイル名-2のレコードを参照する文を実行することはできません。

3. ファイル名-1のレコード形式が固定長で、ファイル名-2のレコードの長さがファイル名-1のレコード長より短い場合、そのレコードがファイル名-1のファイルに引き渡されるとき、レコードの右側の余りの部分には空白が詰められます。

OUTPUT PROCEDURE指定の規則

1. OUTPUT PROCEDURE指定を書いた場合、整列操作が終了した後、出力手続きに制御が移ります。出力手続きの最後の文を実行した後、SORT文の次の実行文に制御が移ります。
2. 出力手続きに制御が移ったとき、ファイル名-1のレコードは完全に整列されていて、RETURN文によって順番に引き取れるような状態になっています。出力手続きの中には、ファイル名-1のファイルに対するRETURN文を書きます。RETURN文は、整列操作の最後の段階から、1つのレコードを引き取ります。
3. 出力手続きの中で実行される文を、「出力手続きの範囲」といいます。出力手続きの範囲は、以下のとおりです。

   ー 出力手続きの中に書いた文

   ー 出力手続きの中に書いた文(たとえばCALL文など)によって制御が移行した結果、実行されるすべての文

   ー 出力手続きの中に書いた文によって実行される宣言手続きの中のすべての文
4. 出力手続きの範囲で、MERGE文、RELEASE文またはSORT文を実行することはできません。
5. 出力手続きの範囲で、SORT文と同じ手続き部内に書かれたEXIT PROGRAM文を実行することはできません。

GIVING指定の規則

1. GIVING指定を書いた場合、整列操作が終了した後、ファイル名-1のファイル中のすべてのレコードが、ファイル名-3のファイルに移されます。
2. ファイル名-3の各ファイルに対して、以下の処理が順に行われます。

   a. 初期処理が行われます。これは、ファイル名-3のファイルに対して、OUTPUT指定のOPEN文を実行したことと同じです。

   b. ファイル-1のレコードを整列操作から引き取ります。これは、ファイル名-1のファイルに対して、レコード名だけを指定したWRITE文を、ファイル名-1のレコードがなくなるまで実行したことと同じです。
   ファイル名-3に相対ファイルを指定した場合、引き取られる相対レコード番号は、最初のレコードから順に、1、2、3、…というように設定されます。ファイル名-3のファイルに相対キー項目を指定した場合、レコードが引き取られるごとに、相対レコード番号が相対キー項目に設定されます。SORT文の実行後の相対キー項目の内容は、ファイル名-3のファイルに引き取られた最後のレコードを表します。

   c. 終了処理が行われます。これは、ファイル名-3だけを指定したCLOSE文を実行したことと同じです。

   上記のa.～c.の処理の中で、ファイル名-3に関連するUSE AFTER STANDARD EXCEPTION手続きが実行されることがあります。そのUSE手続きの中では、ファイル名-3のファイルを操作する文またはファイル名-3のレコードを参照する文を実行することはできません。
3. ファイル名-3のファイルの外部で定義された区域を超えて、最初にレコードを書き出そうとすると、以下の処理が行われます。

   ー ファイル名-3に関連するUSE AFTER STANDARD EXCEPTION手続きを書いた場合

   そのUSE手続きが実行されます。そして、USE手続きから制御が戻った後、ファイル名-3だけを指定したCLOSE文を実行したかのように、終了処理が行われます。

   ー ファイル名-3に関連するUSE AFTER STANDARD EXCEPTION手続きを書かなかった場合

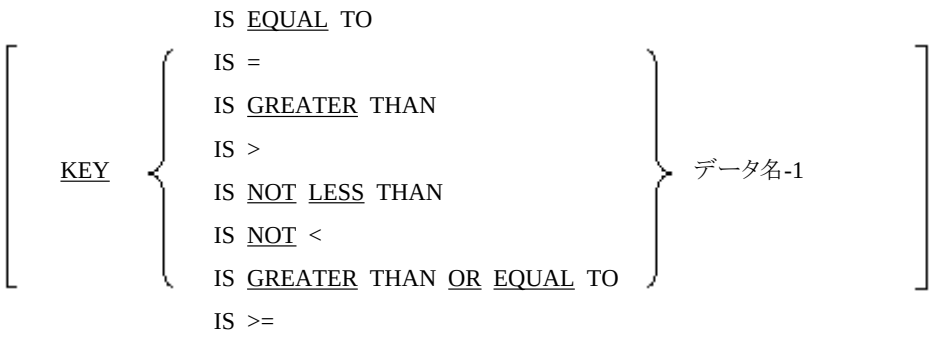
   ファイル名-3だけを指定したCLOSE文を実行したかのように、終了処理が行われます。
4. ファイル名-3のレコード形式が固定長で、ファイル名-1のレコードの長さがファイル名-3のレコード長より短い場合、そのレコードがファイル名-3のファイルに引き取られるとき、レコードの右側の余りの部分には空白が詰められます。

6.4.42 START文（相対ファイル）

レコードを順呼出しするためにファイルを論理的に位置付けます。

【書き方】

START ファイル名-1



[INVALID KEY 無条件文-1]

[NOT INVALID KEY 無条件文-2]

[END-START]

注意

“＝”、“＞”、“＜”および“＞＝”は必要語です。ここでは、他の記号との混同を避けるために下線を付けていません。

構文規則

1. ファイル名-1は、順呼出し法または動的呼出し法のファイルでなければなりません。
2. データ名-1は、ファイル名-1に関連するファイル管理記述項のACCESS MODE句のRELATIVE KEY指定に書いたデータ項目(相対キー項目)でなければなりません。データ名-1は、修飾することができます。
3. ファイル名-1に関連するUSE AFTER STANDARD EXCEPTION手続きを書かない場合、INVALID KEY指定を書かなければなりません。

   ただし、このコンパイラでは、関連するUSE AFTER STANDARD EXCEPTION手続きとINVALID KEY指定の両方を省略することができます。

一般規則

1. ファイル名-1のファイルは、START文を実行する前に入力モードまたは入出力両用モードで開いておかなければなりません。
2. KEY指定を省略した場合、比較演算子“IS EQUAL TO”を指定したものとみなされます。
3. START文を実行しても、ファイル名-1のレコード領域の内容は変更されません。
4. KEY指定の記述に従って、ファイル名-1のファイルのレコードに関連するキーと、ファイル名-1の相対キー項目が比較されます。比較では数字比較の規則が適用されます。比較の結果、以下の処理が行われます。

   ー ファイル名-1のファイル中に比較条件を満足するレコードが存在した場合

   比較条件を満足する最初のレコードに、ファイル位置指示子が設定されます。

   ー ファイル名-1のファイル中に比較条件を満足するレコードが存在しなかった場合

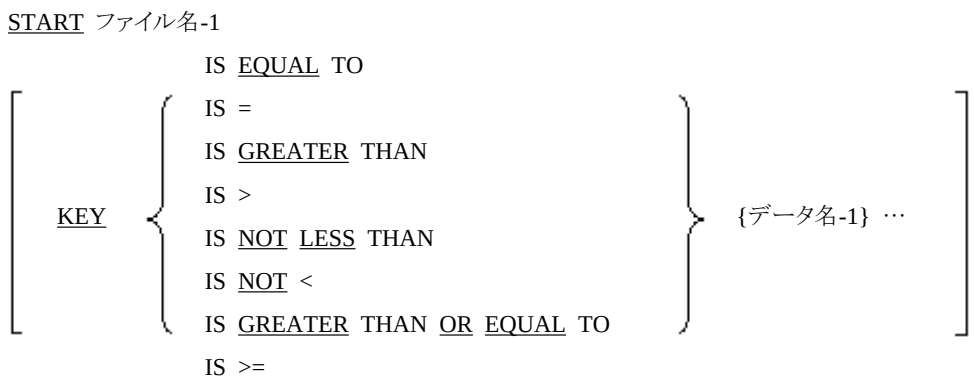
   無効キー条件が発生し、START文の実行は不成功になります。
5. START文を実行すると、ファイル名-1の入出力状態の値が更新されます。
6. START文を実行する前に、ファイル位置指示子が入力の不定ファイルが存在しないことを示している場合、無効キー条件が発生し、START文の実行は不成功になります。
7. START文の実行が不成功になった場合、ファイル位置指示子には有効な次のレコードがないことを示す値が設定されます。

8. START文の実行中に無効キー条件が発生すると、ファイル名-1の入出力状態に無効キー条件を示す値が設定された後、“6.3.12 INVALID KEY指定”の規則に従って制御が移ります。
9. START文の実行中に例外条件が発生しなかった場合、ファイル名-1の入出力状態が設定された後、“6.3.12 INVALID KEY指定”の規則に従って制御が移ります。
10. END-START指定は、START文の範囲を区切ります。
11. ファイル名-1に関連するファイルのLOCK MODE句にAUTOMATICを指定した場合、START文を実行すると、存在するレコードのロックが解除されます。

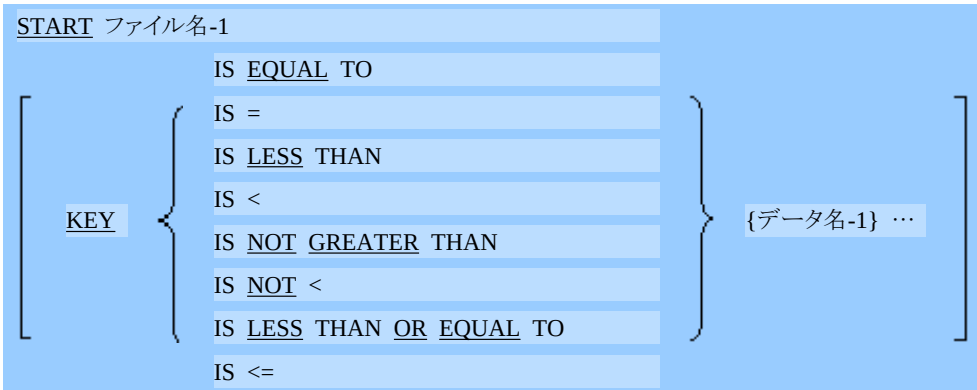
6.4.43 START文（索引ファイル）

レコードを順呼出しするためにファイルを論理的に位置付けます。

【書き方1】正順に順呼出しするために、ファイルを論理的に位置付ける



【書き方2】逆順に順呼出しするために、ファイルを論理的に位置付ける



【書き方3】参照キーの先頭または末尾のレコードにファイルを論理的に位置付ける

```
START ファイル名-1
    FIRST
    [KEY IS {データ名-1} … ]
    [WITH REVERSED ORDER]
    [INVALID KEY 無条件文-1]
    [NOT INVALID KEY 無条件文-2]
    [END-START]
```

注意

“＝”、“＞”、“＜”、“＞＝”および“＜＝”は必要語です。ここでは、他の記号との混同を避けるために下線を付けていません。

構文規則

1. ファイル名-1は、順呼出し法または動的呼出し法のファイルでなければなりません。
2. データ名-1は、修飾することができます。
3. ファイル名-1に関連するUSE AFTER STANDARD EXCEPTION手続きを書かない場合、INVALID KEY指定を書かなければなりません。

   ただし、このコンパイラでは、関連するUSE AFTER STANDARD EXCEPTION手続きとINVALID KEY指定の両方を省略することができます。
4. KEY指定を書く場合、データ名-1は以下のいずれかでなければなりません。

   ただし、書き方3では以下のa.でなければなりません。

   a. ファイル名-1に関連するファイル管理記述項のRECORD KEY句に書いたデータ項目(主レコードキー項目)またはALTERNATE RECORD KEY句に書いたデータ項目(副レコードキー項目)

      ただし、RECORD KEY句に2つ以上のデータ名を書いた場合、データ名-1の並びに、RECORD KEY句に書いた順に同じ個数のデータ名を書かなければなりません。同様に、ALTERNATE RECORD KEY句に2つ以上のデータ名を書いた場合、データ名-1の並びに、ALTERNATE RECORD KEY句に書いた順に同じ個数のデータ名を書かなければなりません。

   b. ファイル名-1に関連するRECORD KEY句に書いたデータ名の並びの一部。

      すなわち、RECORD KEY句に書いたn個のデータ名のうちの1番目からn-1番目以下までのデータ名を、RECORD KEY句に書いたときと同じ順に同じ名前で指定します。

   c. ファイル名-1に関連するALTERNATE RECORD KEY句に書いたデータ名の並びの一部

      すなわち、ALTERNATE RECORD KEY句に書いたn個のデータ名のうちの1番目からn-1番目以下までのデータ名をALTERNATE RECORD KEY句に書いたときと同じ順に同じ名前で指定します。

   d. ファイル名-1に関連するRECORD KEY句にデータ名を1つだけ書いた場合で、以下の条件をすべて満足するデータ項目

      - ファイル名-1に関連するレコードの中に存在し、左端の文字位置が主レコードキー項目の左端の文字位置と等しい。
      - 大きさが主レコードキー項目の大きさより小さいか等しい。
      - 英数字項目、日本語項目または集団項目である。
        このとき、データ名-1の並びには、1つのデータ名だけ書くことができます。

e. ファイル名-1に関連するALTERNATE RECORD KEY句にデータ名を1つだけ書いた場合で、以下の条件をすべて満足するデータ項目

- ファイル名-1に関連するレコードの中に存在し、左端の文字位置が副レコードキー項目の左端の文字位置と等しい。
- 大きさが副レコードキー項目の大きさより小さい、または等しい。
- 英数字項目、日本語項目または集団項目である。
このとき、データ名-1の並びには、1つのデータ名だけ書くことができます。

一般規則

書き方1～書き方3に共通する規則

1. ファイル名-1のファイルは、START文を実行する前に入力モードまたは入出力両用モードで開いておかなければなりません。
2. START文を実行しても、レコード領域の内容は変更されません。また、ファイル名-1に関連するRECORD句のDEPENDING ON指定に指定したデータ項目の内容も変更されません。
3. START文を実行すると、ファイル名-1の入出力状態の値が更新されます。
4. START文を実行する前に、ファイル位置指示子が不定ファイルが存在しないことを示している場合、無効キー条件が発生し、START文の実行は不成功になります。
5. START文の実行が不成功になった場合、ファイル位置指示子には、有効な次のレコードがないことを示す値が設定されます。参照キーは規定されません。
6. 参照キーは、以下の規則に従って決定されます。以下の規則は、この順序で適用されます。
   a. KEY指定を省略した場合、ファイル名-1の主レコードキーが参照キーになります。
   b. KEY指定のデータ名-1の並びに、RECORD KEY句に書いた順に同じ個数で同じ名前のデータ名の並びを書いた場合、主レコードキーが参照キーになります。
   c. KEY指定のデータ名-1の並びに、ALTERNATE RECORD KEY句に書いた順に同じ個数で同じ名前のデータ名の並びを書いた場合、副レコードキーが参照キーになります。
   d. KEY指定のデータ名-1の並びに、RECORD KEY句に書いたデータ名の並びの一部を書いた場合、すなわち、RECORD KEY句に書いたn個のデータ名のうちの1番目からn-1番目以下までのデータ名を同じ順に書いた場合、データ名-1の並びに書いた主レコードキーが参照キーになります。
   e. KEY指定のデータ名-1の並びに、ALTERNATE RECORD KEY句に書いたデータ名の並びの一部を書いた場合、すなわち、ALTERNATE RECORD KEY句に書いたn個のデータ名のうちの1番目からn-1番目以下までのデータ名を同じ順に書いた場合、以下の条件を満足する副レコードキーが参照キーになります。
      - データ名-1が1つのALTERNATE RECORD KEY句で指定した副レコードキーの一部に一致する場合、その副レコードキーが参照キーになります。
      - データ名-1が2つ以上のALTERNATE RECORD KEY句で指定した副レコードキーの一部に一致する場合、最小の個数のデータ名から構成される副レコードキーが参照キーになります。最小の個数のデータ名から構成される副レコードキーが2つ以上存在する場合は、ファイル名-1のファイル管理記述項でALTERNATE RECORD KEY句を先に書いた方の副レコードキーが参照キーになります。
   f. KEY指定のデータ名-1に、RECORD KEY句に書いたデータ名以外かつALTERNATE RECORD KEY句に書いたデータ名以外のデータ名を指定した場合、以下の条件をすべて満足するレコードキーが参照キーになります。
      - 最左端の文字位置がデータ名-1のデータ項目の最左端の文字位置と対応する。
      - 大きさがデータ名-1のデータ項目の大きさより大きい、または等しい。
      - ただ1つのデータ項目から構成されている。
      - 英数字項目、日本語項目または集団項目である。

g. 上記f.の条件をすべて満足するレコードキーが2つ以上存在する場合は、以下に示すレコードキーが参照キーになります。

- 主レコードキーが、f.の条件を満足する場合、主レコードキーが参照キーになります。
- 主レコードキーが、f.の条件を満足しない場合、条件を満足する副レコードキーのうち、大きさが最も小さい副レコードキーが参照キーになります。大きさが最も小さい副レコードキーが2つ以上存在する場合は、ALTERNATE RECORD KEY句で先に指定した副レコードキーが参照キーになります。

7. 6.の規則に従って決定された参照キーが、START文でレコードキーの順序を定めるために使われます。START文の実行が成功した場合、START文で設定された参照キーは、引き続く順呼出しのREAD文のためにも使われます。
8. START文の実行が成功すると、ファイル名-1のファイルに対する順呼出し法のREAD文の検索方法は、以下のように決定されます。
   - REVERSED ORDER指定なしのSTART文の場合、検索方法は正順になります。
   - REVERSED ORDER指定付きのSTART文の場合、検索方法は逆順になります。
9. START文の実行中に無効キー条件が発生すると、ファイル名-1の入出力状態に無効キー条件を示す値が設定された後、“6.3.12 INVALID KEY指定”の規則に従って制御が移ります。
10. START文の実行中に例外条件が発生しなかった場合、ファイル名-1の入出力状態が設定された後、“6.3.12 INVALID KEY指定”の規則に従って制御が移ります。
11. END-START指定は、START文の範囲を区切ります。
12. ファイル名-1に関連するファイルのLOCK MODE句にAUTOMATICを指定した場合、START文を実行すると、存在するレコードのロックが解除されます。

書き方1および書き方2に共通する規則

1. KEY指定の記述に従って、ファイル名-1のファイルに存在するレコードのキーと、以下のデータ項目が比較されます。
   - KEY指定を書いた場合、データ名-1のデータ項目が比較されます。
   - KEY指定を省略した場合、ファイル名-1のRECORD KEY句に書いたデータ項目が比較されます。KEY指定の比較演算子には“IS EQUAL TO”を書いたものとみなされます。
2. 1.の比較は、文字の大小順序に従って昇順に並んでいる、ファイル内のレコードの参照キーに対して行われます。作用対象の長さが異なる場合、長い方の右側が短い方と同じ長さになるように切り捨てられ、文字比較が行われます。比較の結果、以下の処理が行われます。
   - 書き方1の場合、ファイル位置指示子は、比較条件を満足する最初のレコードの示す値に設定されます。
   - 書き方2の場合、ファイル位置指示子は、比較条件を満足する最後のレコードの示す値に設定されます。
   - ファイル中のどのレコードも上記の条件を満足しない場合、無効キー条件が発生し、START文の実行は不成功になります。

書き方3の規則

1. 書き方3のSTART文を実行すると、ファイル名-1のファイル位置指示子が以下のように位置付けられます。
   - REVERSED ORDER指定を省略した場合、ファイル名-1のファイルは、その参照キーにおける先頭のレコードに位置付けられます。ファイル位置指示子は、位置付けられたレコードの参照キーの値に設定されます。
   - REVERSED ORDER指定を書いた場合、ファイル名-1のファイルは、その参照キーにおける末尾のレコードに位置付けられます。ファイル位置指示子は、位置付けられたレコードの参照キーの値に設定されます。
   - ファイル中に有効なレコードが1つも存在しない場合、無効キー条件が発生し、START文の実行は不成功になります。
2. データ名-1は、参照キーを決定するためだけに指定します。データ名-1の値は無視されます。

例

START文における参照キー選択の例

```
*〔環境部〕
   INPUT-OUTPUT SECTION.
     FILE-CONTROL.
       SELECT IFILE-1 ASSIGN TO SYS006
         ORGANIZATION IS INDEXED
         RECORD KEY IS MKEY1 MKEY2                    *> [A]
         ALTERNATE RECORD KEY IS IKEY1 IKEY2 IKEY3    *> [B]
         ALTERNATE RECORD KEY IS IKEY2 IKEY3 IKEY4    *> [C]
         ALTERNATE RECORD KEY IS IKEY1 IKEY3          *> [D]
         ALTERNATE RECORD KEY IS IKEY3 IKEY4 IKEY5    *> [E]
         ALTERNATE RECORD KEY IS IKEY3 IKEY4          *> [F]
         ALTERNATE RECORD KEY IS IKEY3 IKEY5.         *> [G]
*〔データ部〕
  FILE SECTION.
   FD  IX-FS1.
   01  IX-FS1R1-F-G-240.
     02 IX-FS1-GRP-120.
       03 MKEY1    PIC X(5).
       03 MKEY2    PIC X(5).
       03 IKEY1    PIC X(29).
       03 IKEY2    PIC S9(9).
       03 IKEY3    PIC X(10).
       03 IKEY4    PIC X(100).
       03 IKEY5    PIC 9(7).
          :
*〔手続き部〕
     START IX-FS1.                              *> 規則 6.a)により参照キーは[A]
     START IX-FS1 KEY IS = MKEY1 MKEY2.         *> 規則 6.b)により参照キーは[A]
     START IX-FS1 KEY IS = IKEY1 IKEY2 IKEY3.   *> 規則 6.c)により参照キーは[B]
     START IX-FS1 KEY IS = MKEY1.               *> 規則 6.d)により参照キーは[A]
     START IX-FS1 KEY IS = IKEY1 IKEY2.         *> 規則 6.e)により参照キーは[B]
     START IX-FS1 KEY IS = IKEY3.               *> 規則 6.e)により参照キーは[F] (*1)
```

*1 :

- [B][C][D]は、IKEY3が1番目に書かれていないので選択から除外されます。
- [E][F][G]では、構成されるデータ名の個数が小さい[F][G]が選択されます。
- [F][G]では、先に書かれている[F]が選択されます。

6.4.44 STOP文（中核）

実行単位の実行を終了します。定数-1を指定したSTOP文は、廃要素です。

【書き方】

STOP { RUN / 定数-1 }

構文規則

1. 定数-1は、以下であってはいけません。
   - ALLで始まる表意定数
   - ALLで始まる日本語定数
2. 定数-1に数字定数を指定する場合、符号なしの整数でなければなりません。

   ただし、このコンパイラでは、定数-1に符号付きまたは小数部を持つ定数を指定することもできます。
3. STOP RUN文を1つの完結文の中の一連の無条件文の中に書く場合、STOP RUN文は、一連の無条件文の中の最後の文でなければなりません。

一般規則

1. STOP RUN文を実行すると、実行単位の実行が終わり、制御がオペレーティングシステムに移されます。
2. STOP RUN文を実行したとき、開いた状態のファイルがある場合は、それらのファイルに対して選択指定のないCLOSE文が実行されます。これらのファイルに関連するUSE手続きは、実行されません。
3. 定数-1を書いたSTOP文を実行すると、実行単位の実行が中断し、定数-1の値が操作員に示されます。操作員が実行の継続を指示した場合、STOP文の次の実行文から、実行が再開されます。

6.4.45 STRING文（中核）

文字列を連結します。

使い方の例については、“付録H サンプル集”の“H.41 STRING文”を参照してください。

【書き方1】

```
STRING { { 一意名-1 } … DELIMITED BY { 一意名-2 } } …
       { { 定数-1   }                { 定数-2   }
                                     { SIZE     }
  INTO 一意名-3 [WITH POINTER 一意名-4]
  [ON OVERFLOW 無条件文-1]
  [NOT ON OVERFLOW 無条件文-2]
  [END-STRING]
```

【書き方2】

```
STRING { 一意名-5 } … INTO 一意名-6
       { 定数-3   }
  BY { CSV-FORMAT } [TYPE IS { MODE-1 } ]
     { TSV-FORMAT }          { MODE-2 }
                             { MODE-3 }
                             { MODE-4 }
  [WITH POINTER 一意名-7]
  [ON OVERFLOW 無条件文-3]
  [NOT ON OVERFLOW 無条件文-4]
  [END-STRING]
```

注意

【書き方2】は、[Win32][WinIPF][Winx64][Solaris][LinuxIPF][Linux64]固有機能です。

構文規則

書き方1の規則

1. 定数-1および定数-2は、以下でなければなりません。
   - 文字定数
   - 日本語定数
   - ALLで始まらない表意定数
2. 一意名-1～一意名-3の用途は、表示用でなければなりません。
3. 一意名-1および一意名-2に数字項目を指定する場合、それはPICTURE句の文字列に文字“P”を含まない整数項目でなければなりません。
4. 一意名-3は、以下であってはいけません。
   - 英数字編集項目
   - 日本語編集項目
   - 数字編集項目

   また、一意名-3は、JUSTIFIED句を指定したデータ項目であってはいけません。
5. 一意名-3は、部分参照することはできません。
6. 一意名-4は、PICTURE句の文字列に文字“P”を含まない整数項目でなければなりません。また、一意名-4は、一意名-3のデータ項目の文字数に1を加えた値を持つことができる大きさでなければなりません。
7. 一意名-1～一意名-3、定数-1または定数-2のいずれかの項類が日本語または日本語編集の場合、これらの項類はすべて日本語または日本語編集でなければなりません。

書き方2の規則

1. DECIMAL POINT IS COMMA句が指定されてはいけません。
2. 定数-3は、文字定数、日本語定数または表意定数SPACEでなければなりません。
3. 一意名-5は、以下に挙げる句を含んではいけません。また、用途が表示用のデータ項目、または用途が表示用のデータ項目のみで構成される集団項目でなければなりません。
   - ANY LENGTH句
   - REDEFINES句
   - BLANK WHEN ZERO句
   - CHARACTER TYPE句
   - JUSTIFIED句
   - DEPENDING ON指定のOCCURS句
   - PRINTING POSITION句
   - SYNCHRONIZED句
   - TYPE句

4. 一意名-6は、編集データ項目でない、項類が英数字のデータ項目でなければなりません。また、以下の句を記述してはいけません。

   ー JUSTIFIED句

   ー ANY LENGTH句

5. 一意名-7は、PICTURE句の文字列に文字“P”を含まない整数項目でなければなりません。また、一意名-7は、一意名-6のデータ項目の文字数に1を加えた値をもつことができる大きさでなければなりません。

6. 一意名-5または一意名-5に従属する基本項目が数字項目の場合、PICTURE句の文字列に文字“P”を含まない整数項目でなければなりません。

一般規則

書き方1の規則

1. STRING文は、一意名-1または定数-1のいくつかの文字列を連結して、一意名-3に転記します。一意名-1および定数-1を「送出し側項目」といいます。一意名-3を「受取り側項目」といいます。STRING文を実行すると、各送出し側項目の文字列全体または文字列の一部が、送出し側項目を書いた順に、受取り側項目の最左端文字位置または一意名-4で指定した文字位置以降に転記されます。

   ー 送出し側項目のすべての文字列を転記する場合、DELIMITED BY指定にSIZEを書きます。また、送出し側項目の最左端文字位置からある文字までの文字列を転記する場合、DELIMITED BY指定に一意名-2または定数-2を指定します。一意名-2または定数-2を「区切り文字」といいます。

   ー 受取り側項目の特定の文字位置以降に送出し側項目を転記する場合、POINTER指定を書き、POINTER指定の一意名-4にその文字位置を設定します。一意名-4の値は、転記が行われるごとに転記された文字列の長さだけ増やされます。受取り側項目の最左端文字位置以降に送出し側項目を転記する場合は、POINTER指定を省略します。

2. DELIMITED BY指定の記述に従って、各送出し側項目の中の以下の文字列が転記されます。

   a. DELIMITED BY指定に一意名-2または定数-2を書いた場合、送出し側項目の最左端から、区切り文字と同じ文字が現れるまでの文字列。なお、区切り文字自身は、転記されません。

   b. DELIMITED BY指定にSIZEを書いた場合、送出し側項目の文字列全体。

3. 一意名-4の初期値は、STRING文を実行する前に設定しなければなりません。初期値は、1以上でなければなりません。「現在の転記位置」の初期値として、一意名-4の初期値が使われます。「現在の転記位置」は転記が行われるごとに1ずつ変化し、STRING文の実行が終了したときの「現在の転記位置」が一意名-4の値になります。

4. POINTER指定を省略した場合、「現在の転記位置」の初期値は1です。

5. 各送出し側項目から受取り側項目への転記は、転記の規則に従って、1文字ずつ行われます。転記の際、空白詰めは行われません。各送出し側項目から受取り側項目への転記では、以下の処理が繰り返されます。

   a. 「現在の転記位置」が受取り側項目の文字数を超えていない場合、送出し側項目の中の1文字が、現在の転記位置に転記されます。そして、「現在の転記位置」の値が1だけ増やされます。

   b. 「現在の転記位置」が受取り側項目の文字数を超えている場合、オーバフロー条件が発生し、転記が終了します。そして、ON OVERFLOW指定の記述に従って、制御が移ります。

   c. DELIMITED BY指定に書いた区切り文字を検出するか(DELIMITED BY指定に一意名-2または定数-2を書いた場合)、送出し側項目のデータを全部転記するか、または受取り側項目の終わりに達するまでa.またはb.の処理が繰り返されます。

6. 受取り側項目の中で、STRING文の実行によって値が変化するのは、転記が行われた部分だけです。受取り側項目の他の部分は、STRING文を実行しても変化しません。

7. 区切り文字に表意定数を指定した場合、表意定数は1文字の文字定数または日本語定数とみなされます。

8. STRING文の実行で転記処理を行った後の制御の移行は、ON OVERFLOW指定およびNOT ON OVERFLOW指定の有無によって異なります。転記処理の後の制御の移行を、下表に示します。

ON OVERFLOW指定	NOT ON OVERFLOW指定	制御の移行	
		転記処理でオーバフロー条件が発生した場合(*3)	転記処理でオーバフロー条件が発生しなかった場合
あり	あり	[1]転記が終了し、無条件文-1に制御が移ります。 [2]無条件文-1の実行後、STRING文の終わりに制御が移ります。(*1)	[1]すべての転記が終了した後、無条件文-2に制御が移ります。 [2]無条件文-2の実行後、STRING文の終わりに制御が移ります。(*2)
あり	なし	[1]転記が終了し、無条件文-1に制御が移ります。 [2]無条件文-1の実行後、STRING文の終わりに制御が移ります。(*1)	すべての転記が終了した後、STRING文の終わりに制御が移ります。
なし	あり	転記が終了し、STRING文の終わりに制御が移ります。	[1]すべての転記が終了した後、無条件文-2に制御が移ります。 [2]無条件文-2の実行後、STRING文の終わりに制御が移ります。(*2)
なし	なし	転記が終了し、STRING文の終わりに制御が移ります。	すべての転記が終了した後、STRING文の終わりに制御が移ります。

*1 ： 無条件文-1に制御の明示移行を起こす手続き分岐または条件文を書いた場合、その文の規則に従って制御が移ります。

*2 ： 無条件文-2に制御の明示移行を起こす手続き分岐または条件文を書いた場合、その文の規則に従って制御が移ります。

*3 ： オーバフロー条件は、以下の場合に発生します。

- 一意名-4の初期値が、1より小さいかまたは一意名-3の文字列の文字数より大きいとき
- 「現在の転記位置」が、一意名-3の文字列の文字数より大きくなったとき。すなわち、一意名-3の最右端まで転記されたのに、送出し側項目の中に、転記されない文字が残っているとき

9. END-STRING指定は、STRING文の範囲を区切ります。
10. 以下の場合、STRING文の実行結果は規定されません。
    a. 一意名-1または一意名-2が、一意名-3または一意名-4と同じ記憶領域を占有する場合
    b. 一意名-3と一意名-4が同じ記憶領域を占有する場合

書き方2の規則

1. 一意名-5および定数-3を「送出し側項目」といいます。一意名-6を「受取り側項目」といいます。
2. 定数-3に表意定数を使うと、用途が表示用である1けたのデータ項目とみなされます。
3. STRING文を実行すると、一意名-5または定数-3のデータ項目に格納されている文字列を、以下のように加工してから、基本項目単位にBY指定の形式で連結し、一意名-6に転記します。一意名-5が集団項目の場合、従属するデータ項目の先頭から順に転記します。
    - 項類が英数字および日本語の場合、後置空白を削除する
    - 項類が数字の場合、先行ゼロ列（有効な数字より上位桁のゼロ）および小数部をもつ場合は後置ゼロ列を削除する
4. 3.の連結処理において、送出し側と受取り側の字類が異なる場合、受取り側項目の字類に合わせてデータを変換します。データの変換処理で異常を検出した場合、オーバフロー条件が生じます。
5. BY指定は、加工するデータ形式を指定します。
    a. CSV-FORMATが指定された場合、一意名-6はTYPE指定によって指定されたCSV形式に編集されます。CSV形式とは、複数の文字列データをカンマで区切った文字データのことです。

b. TSV-FORMATが指定された場合、一意名-6はTYPE指定によって指定されたTSV形式に加工されます。TSV形式とは、複数の文字列データをタブで区切った文字データのことです。

6. TYPE指定は、編集形式を指定します。MODE-1～4のそれぞれの形式については、NetCOBOL使用手引書を参照してください。

7. TYPE指定を書かない場合は、MODE-1が指定されたものとみなします。

8. POINTER（文字位置）指定を書く場合、一意名-7のデータ項目の初期値は、STRING文の実行前に利用者が設定しなければなりません。初期値は1以上でなければなりません。

9. POINTER指定を書かない場合は、一意名-7に初期値として1を入れたものとみなします。

10. 一意名-6のデータ項目に文字が送られるときの動作は、次のようになります。一意名-7のデータ項目の値が、一意名-6のデータ項目の文字数を超えていなければ、文字は、送出し側から一意名-6のデータ項目中の、一意名-7のデータ項目の値が示す文字位置へ、1文字ずつ送られます。その後、次の文字が送られる前またはSTRING文の実行が終わる前に、一意名-7のデータ項目が1だけ増やされます。STRING文の実行中に、一意名-7のデータ項目の値が変わるのは、この場合だけとします。

11. STRING文の実行が終わったとき、一意名-6のデータ項目の中でその値が変わるのは、このSTRING文で文字が送られた部分だけです。一意名-6のデータ項目の他の部分には、このSTRING文の実行前の内容が残ります。

12. STRING文の実行で転記処理を行った後の制御の移行は、ON OVERFLOW指定およびNOT ON OVERFLOW指定の有無によって異なります。転記処理後の制御の移行を、下表に示します。

ON OVERFLOW指定	NOT ON OVERFLOW指定	制御の移行	
		転記処理でオーバフロー条件が発生した場合(*3)	転記処理でオーバフロー条件が発生しなかった場合
あり	あり	[1]転記が終了し、無条件文-3に制御が移ります。 [2]無条件文-3の実行後、STRING文の終わりに制御が移ります。(*1)	[1]すべての転記が終了した後、無条件文-4に制御が移ります。 [2]無条件文-4の実行後、STRING文の終わりに制御が移ります。(*2)
あり	なし	[1]転記が終了し、無条件文-3に制御が移ります。 [2]無条件文-3の実行後、STRING文の終わりに制御が移ります。(*1)	すべての転記が終了した後、STRING文の終わりに制御が移ります。
なし	あり	転記が終了し、STRING文の終わりに制御が移ります。	[1]すべての転記が終了した後、無条件文-4に制御が移ります。 [2]無条件文-4の実行後、STRING文の終わりに制御が移ります。(*2)
なし	なし	転記が終了し、STRING文の終わりに制御が移ります。	すべての転記が終了した後、STRING文の終わりに制御が移ります。

*1 ： 無条件文-3に制御の明示移行を起こす手続き分岐または条件文を書いた場合、その文の規則に従って制御が移ります。

*2 ： 無条件文-4に制御の明示移行を起こす手続き分岐または条件文を書いた場合、その文の規則に従って制御が移ります。

*3 ： オーバフローの条件は、以下の場合に発生します。

－ 一意名-7の初期値が1より小さいか、一意名-6のデータ項目の文字数より大きい

－ 転記時に受取り側の字類に合わせてデータ変換する際、異常を検出した

－ 一意名-7の値が、一意名-6のデータ項目の文字数を超えた

13. END-STRING指定は、STRING文の範囲を区切ります。

14. 一意名-5が一意名-6または一意名-7と同じ記憶領域を占有する場合、または一意名-6と一意名-7が同じ記憶領域を占有する場合、たとえそれらが同じデータ記述項によって定義されていても、この文の実行結果は規定されません。

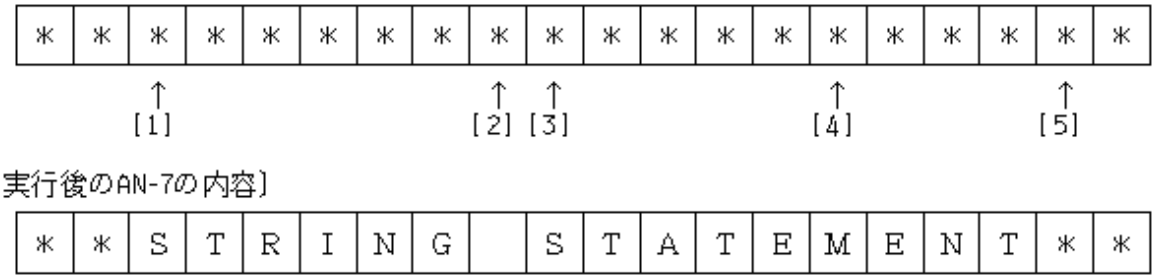
STRING文の処理の流れ

以下のSTRING文を例に、処理の流れを説明します。

```
*〔データ部〕
  77  AN-1  PIC X(6)   VALUE "STRING".
  77  AN-2  PIC X(5)   VALUE "STATE".
  77  AN-4  PIC X(6)   VALUE "MENT#0".
  77  AN-5  PIC X(4)   VALUE "#123".
  77  AN-7  PIC X(20)  VALUE "**********".
  77  ED-8  PIC 99     VALUE 3.
*〔手続き部〕
     STRING AN-1 SPACE AN-2 DELIMITED BY SIZE
            AN-4 AN-5 DELIMITED BY "#"
                 INTO AN-7 WITH POINTER ED-8.
```

このSTRING文を実行すると、送出し側項目(AN-1、SPACE、AN-2、AN-4およびAN-5)が順に連結され、受取り側項目(AN-7)に転記されます。

図6.5 実行前のAN-7の内容



1. ED-8の初期値が3なので、AN-7の3文字目([1]の位置)以降が、転記の対象になります。
2. 最初に、AN-1のすべての文字列が転記されます。転記の結果、ED-8の値は9([2]の位置)に更新されます。
3. 次に、表意定数SPACEが文字定数1文字として転記されます。転記の結果、ED-8の値は10([3]の位置)に更新されます。
4. 次に、AN-2のすべての文字列が転記されます。転記の結果、ED-8の値は15([4]の位置)に更新されます。
5. 次に、AN-4の文字列のうち“#”が現れるまでの文字列が転記されます。転記の結果、ED-8の値は19([5]の位置)に更新されます。
6. 次に、AN-5の文字列のうち“#”が現れるまでの文字列がないので、何も転記されません。ED-8の値は更新されません。
7. STRING文の実行が終了したときのED-8の値は、19です。

6.4.46 SUBTRACT文（中核）

減算の結果を求めます。

【書き方1】減算の結果を被減数と置き換える

SUBTRACT { 一意名-1 / 定数-1 } …

FROM {一意名-2 [ROUNDED]} …

[ON SIZE ERROR 無条件文-1]

[NOT ON SIZE ERROR 無条件文-2]

[END-SUBTRACT]

【書き方2】減算の結果を被減数とは異なるデータ項目に格納する

SUBTRACT { 一意名-1 / 定数-1 } …

FROM { 一意名-2 / 定数-2 } GIVING {一意名-3 [ROUNDED]} …

[ON SIZE ERROR 無条件文-1]

[NOT ON SIZE ERROR 無条件文-2]

[END-SUBTRACT]

【書き方3】2つの集団項目に従属するデータ項目の対応をとって減算する

SUBTRACT { CORRESPONDING / CORR }

一意名-1 FROM 一意名-2 [ROUNDED]

[ON SIZE ERROR 無条件文-1]

[NOT ON SIZE ERROR 無条件文-2]

[END-SUBTRACT]

構文規則

1. 書き方1と書き方2の場合、一意名-1および一意名-2は、数字項目でなければなりません。書き方3の場合、一意名-1および一意名-2は、集団項目でなければなりません。
2. 一意名-3は、数字項目または数字編集項目でなければなりません。
3. 定数-1および定数-2は、数字定数でなければなりません。
4. CORRESPONDINGとCORRは同義語です。

一般規則

書き方1の規則

書き方1のSUBTRACT文は、FROMの前の各作用対象の和を一意名-2から引き、一意名-2に格納します。一意名-2の並びを書いた順に、この減算を行います。

書き方2の規則

書き方2のSUBTRACT文は、FROMの前の各作用対象の和を、FORMの後の作用対象から引き、一意名-3に格納します。一意名-3の並びを書いた順に、この減算の結果を格納します。

書き方3の規則

書き方3のSUBTRACT文は、一意名-1に従属するデータ項目と一意名-2に従属するデータ項目のうち、名前の修飾語の系列が同じものどうしの差を求めます。一意名-1に従属するデータ項目を減数、一意名-2に従属するデータ項目を被減数として差を求め、一意名-2に従属するデータ項目に格納します。この結果は、対応する一意名ごとに、別々のSUBTRACT文を書いた結果と同じです。

書き方1～書き方3に共通する規則

1. END-SUBTRACT指定は、SUBTRACT文の範囲を区切ります。
2. ROUNDED指定、ON SIZE ERROR指定、CORRESPONDING指定、演算および転記の規則については、“6.3 文に関する共通の規則”を参照してください。

6.4.47 SUPPRESS文（報告書作成）

報告集団の表示を抑制します。

【書き方】

SUPPRESS PRINTING

注意

[WinIPF][Winx64][LinuxIPF][Linux64][.NET]では報告書作成機能は使用できません。

構文規則

SUPPRESS文は、USE BEFORE REPORTING手続きの中にだけ書くことができます。

一般規則

1. SUPPRESS文が表示を抑制する報告集団は、そのSUPPRESS文が書かれているUSE手続きに指定された報告集団だけです。
2. 報告集団の表示を抑制するときは、そのつどSUPPRESS文を実行しなければなりません。
3. SUPPRESS文を実行すると、報告書作成管理システムは、以下に示す処理を行いません。
   - 報告集団の印字行の表示
   - 報告集団中のすべてのLINE NUMBER句の処理
   - 報告集団中のNEXT GROUP句の処理
   - 行カウンタの更新

6.4.48 TERMINATE文（報告書作成）

報告書の処理を終了させます。

【書き方】

TERMINATE {報告書名-1} …

注意

[WinIPF][Winx64][LinuxIPF][Linux64][.NET]では報告書作成機能は使用できません。

構文規則

報告書名-1は、データ部の報告書節の報告書記述項で指定しなければなりません。

一般規則

1. TERMINATE文を実行すると、報告書作成管理システムは、すべての制御脚書き報告集団を低いレベルから順に作成表示し、続いて報告書脚書き報告集団を作成表示します。報告書作成管理システムは、最高レベルの制御データ名の制御切れを検出したのと同じように、前に保存した制御データ項目の古い値を、制御脚書き報告集団および報告書脚書き報告集団のSOURCE句やUSE手続きで使えるようにします。
2. ある報告書に対して、INITIATE文の実行後、1回もGENERATE文を実行しないでTERMINATE文を実行すると、報告書作成管理システムは、報告集団の作成表示やそれに関連する処理を一切行いません。
3. 報告書作成管理システムは、報告書の表示中に、本体集団を表示するために改ページしなければならないとき、ページ頭書き報告集団およびページ脚書き報告集団が定義されていれば、自動的にそれらの処理を行います。
4. ある報告書に対して、INITIATE文がまだ実行されていないときおよびTERMINATE文がすでに実行されているときは、TERMINATE文を実行してはいけません。
5. 2つ以上の報告書名-1を書いた場合のTERMINATE文の実行結果は、各報告書名を書いた順に別々のTERMINATE文を実行した場合と同じです。
6. TERMINATE文は、この報告書に関するファイルを閉じる機能は持ちません。TERMINATE文の実行後に、そのファイルを閉じるためには、CLOSE文を別に実行しなければなりません。そのファイルに属する報告書で開始されたものは、CLOSE文を実行する前にすべて終了しておかなければなりません。

6.4.49 UNLOCK文（順ファイル・相対ファイル・索引ファイル）

ファイル結合子に対して取得されたすべてのロックを解除します。

【書き方】

```
UNLOCK ファイル名-1  { RECORD  }
                     { RECORDS }
```

一般規則

1. ファイル名-1は、ロックの有無に関係なく、OPEN文の実行が成功し生成されたファイルでなければなりません。
2. UNLOCK文を実行すると、ファイル名-1のファイル結合子に対して取得されたロックが解除されます。
3. ファイル名-1のファイルにロックが存在していなくても、UNLOCK文の実行は成功します。

6.4.50 UNSTRING文（中核）

文字列を分解します。

使い方の例については、“付録H サンプル集”の“H.42 UNSTRING文”を参照してください。

【書き方1】

```
UNSTRING 一意名-1
[ DELIMITED BY [ALL] { 一意名-2 } [ OR [ALL] { 一意名-3 } ] ] …
                     { 定数-1  }             { 定数-2  }
    INTO {一意名-4 [DELIMITER IN 一意名-5]
```

[COUNT IN 一意名-6]} …
[WITH POINTER 一意名-7]
[TALLYING IN 一意名-8]
[ON OVERFLOW 無条件文-1]
[NOT ON OVERFLOW 無条件文-2]
[END-UNSTRING]

【書き方2】

UNSTRING 一意名-9 INTO 一意名-10 …
BY { CSV-FORMAT / TSV-FORMAT }
[WITH POINTER 一意名-11]
[TALLYING IN 一意名-12]
[ON OVERFLOW 無条件文-3]
[NOT ON OVERFLOW 無条件文-4]
[END-UNSTRING]

注意

【書き方2】は、[Win32][WinIPF][Winx64][Solaris][LinuxIPF][Linux64]固有機能です。

構文規則

書き方1の規則

1. 定数-1および定数-2は、以下でなければなりません。
   - 文字定数
   - 日本語定数
   - ALLで始まらない表意定数
2. 一意名-1、一意名-2、一意名-3および一意名-5は、以下でなければなりません。
   - 英数字項目
   - 日本語項目
   - 日本語編集項目
3. 一意名-4は、以下でなければなりません。
   - 英字項目
   - 英数字項目
   - 日本語項目
   - 数字項目

   数字項目の場合、用途は表示用で、PICTURE句は“P”を含まない文字列でなければなりません。
4. 一意名-6および一意名-8は、PICTURE句の文字列に“P”を含まない整数項目でなければなりません。

5. 一意名-7は、PICTURE句の文字列に“P”を含まない整数項目でなければなりません。また、一意名-7は、一意名-1のデータ項目の文字数に1を加えた値を持つことができる大きさでなければなりません。
6. 一意名-1に部分参照子を付けてはいけません。
7. 一意名-1～一意名-5、定数-1または定数-2のいずれかの項類が日本語または日本語編集の場合、これらの項類は、すべて日本語または日本語編集でなければなりません。

書き方2の規則

1. DECIMAL POINT IS COMMA句が指定されてはいけません。
2. 一意名-9は、編集データ項目でない、項類が英数字のデータ項目でなければなりません。また、以下の句を記述してはいけません。
   - JUSTIFIED句
   - ANY LENGTH句
3. 一意名-10は、PICTURE句の文字列中に文字“P”を含まない、用途が表示用のデータ項目または用途が表示用のデータ項目のみで構成される集団項目でなければいけません。また、以下の句を含んではいけません。
   - ANY LENGTH句
   - REDEFINES句
   - BLANK WHEN ZERO句
   - CHARACTER TYPE句
   - JUSTIFIED句
   - DEPENDING ON指定のOCCURS句
   - PRINTING POSITION句
   - SYNCHRONIZED句
   - TYPE句
4. 一意名-11は整数項目であり、一意名-9のデータ項目の文字数に1を加えた値を持つことができなければなりません。また、一意名-11のPICTURE句の文字列中に文字“P”を含んではいけません。
5. 一意名-12は、PICTURE句の文字列中に文字“P”を含まない整数項目でなければいけません。

一般規則

書き方1の規則

1. UNSTRING文は、一意名-1の文字列をDELIMITED BY指定の条件に従って分解し、一意名-4に転記します。一意名-1を、「送出し側項目」といいます。一意名-4を、「受取り側項目」といいます。一意名-2、一意名-3、定数-1および定数-2を、「区切り文字」といいます。

   DELIMITED BY指定付きのUNSTRING文は、送出し側項目の最左端文字位置または特定の文字位置以降の文字列を区切り文字で区切り、区切った順に各受取り側項目に転記します。DELIMITED BY指定なしのUNSTRING文は、送出し側項目の最左端文字位置または特定の文字位置以降の文字列を、各受取り側項目の長さで区切り、区切った順に各受取り側項目に転記します。

   送出し側項目の最左端文字位置以降の部分を転記の対象にする場合、POINTER指定を省略します。送出し側項目の特定の文字位置以降の部分を転記の対象にする場合、POINTER指定を書き、POINTER指定の一意名-7にその文字位置を設定します。一意名-7の値は、区切り文字との比較が行われるごとに増やされます。

   各受取り側項目への転記において、区切り文字または区切り文字までの文字の個数を得ることができます。区切り文字を得る場合、DELIMITER IN指定を書きます。区切り文字までの文字の個数を得る場合、COUNT IN指定を書きます。

   また、受取り側項目のうち、実際に文字列が転記された受取り側項目の個数を数えることができます。転記された受取り側項目の個数を数える場合、TALLYING IN指定を書きます。

2. 一意名-7の初期値は、UNSTRING文を実行する前に設定しなければなりません。初期値は、1以上でなければなりません。「現在の検査位置」の初期値として、一意名-7の初期値が使われます。「現在の検査位置」の値は、送出し側項目の文字が検査されるごとに、1ずつ増やされます。UNSTRING文の実行が終了したときの「現在の検査位置」の値が、一意名-7の値になります。

3. POINTER指定を省略した場合、「現在の検査位置」の初期値は1です。

4. DELIMITED BY指定を書いた場合、送出し側項目が以下の手順で分割され、各受取り側項目に転記されます。以下で、最初の「現在の受取り側項目」は、INTO指定に書いた最初の一意名-4です。

   a. 「現在の検査位置」が送出し側項目の文字数を超えていない場合、「現在の検査位置」以降の送出し側項目が、区切り文字と1文字ずつ比較されます。区切り文字を2つ以上書いた場合、同じ位置から始まる文字列が、区切り文字を書いた順に区切り文字と一致するまで繰り返し比較されます。区切り文字が2文字以上の場合、区切り文字と同じ文字数の文字列が、区切り文字と比較されます。
   「現在の検査位置」が送出し側項目の文字数を超えている場合、区切り文字の検査を終了します。

   b. a.の比較操作で「現在の検査位置」より後に区切り文字と一致する文字が見つかった場合、「現在の検査位置」から区切り文字と一致した文字の前までの文字列が、「現在の受取り側項目」に転記されます。このとき、区切り文字と一致した文字は、転記されません。DELIMITER IN指定を書いた場合、区切り文字が一意名-5に転記されます。

   c. a.の比較操作で区切り文字と一致する文字が見つからなかった場合、送出し側項目の文字列が「現在の受取り側項目」に転記されます。DELIMITER IN指定に書いた一意名-5には、空白が転記されます。

   d. a.の比較操作で「現在の検査位置」の文字が区切り文字と一致した場合、以下の値が「現在の受取り側項目」に転記されます。

      - 「現在の受取り側項目」が英字項目または英数字項目の場合、空白が転記されます。
      - 「現在の受取り側項目」が日本語項目の場合、空白が転記されます。
      - 「現在の受取り側項目」が数字項目の場合、ゼロが転記されます。

   e. COUNT IN指定を書いた場合、「現在の受取り側項目」に転記された文字の個数が、一意名-6に設定されます。d.の場合は、一意名-6にゼロが設定されます。

   f. 「現在の検査位置」と「現在の受取り側項目」が以下のように変更され、送出し側項目のすべての文字がなくなるかまたは受取り側項目がなくなるまで、a.～f.の処理が繰り返されます。

      - 区切り文字と一致した文字のすぐ右の文字位置が「現在の検査位置」になります。
      - 次の受取り側項目が「現在の受取り側項目」になります。

5. DELIMITED BY指定を省略した場合、送出し側項目が以下の手順で分割され、各受取り側項目に転記されます。以下で、「現在の受取り側項目」の初期値は、INTO指定の中の最初の一意名-4です。

   a. 「現在の検査位置」が送出し側項目の文字数を超えていない場合、「現在の検査位置」以降の送出し側項目の文字数が、「現在の受取り側項目」の文字数と比較されます。「現在の受取り側項目」が数字項目の場合、その文字数として以下の値が使われます。

      - SIGN句にSEPARATEを指定した場合、演算符号の文字数を除いた文字数が使われます。
      - PICTURE句で小数点を指定した場合、小数点の文字数を除いた文字数が使われます。

   b. 「現在の検査位置」が送出し側項目の文字数を超えている場合、転記を終了します。

   c. 「現在の検査位置」から始まる「現在の受取り側項目」の文字数だけの文字列が、「現在の受取り側項目」に転記されます。そして、「現在の検査位置」と「現在の受取り側項目」が以下のように変更され、送出し側項目のすべての文字がなくなるか、または受取り側項目がなくなるまで、a.～c.の処理が繰り返されます。

      - 転記された最後の文字のすぐ右の文字位置が「現在の検査位置」になります。
      - 次の受取り側項目が「現在の受取り側項目」になります。

6. TALLYING指定を書いた場合、受取り側項目のうち、実際に文字列が転記された受取り側項目の数が、一意名-8に加算されます。一意名-8には、初期値を設定しておかなければなりません。一意名-8の値は、1つの一意名-4ごとに1ずつ増加します。

7. 区切り文字に表意定数を指定した場合、表意定数は1文字の文字定数とみなされます。

8. 区切り文字の前にALLを書いた場合、送出し側項目の文字列のうち、区切り文字と同じ文字が現れてから区切り文字以外の文字が現れるまでの文字は、読み飛ばされます。DELIMITER IN指定を書いた場合、読み飛ばされた文字数とは関係なく、1回の繰返しの区切り文字が一意名-5に設定されます。
9. 区切り文字には、計算機文字集合中の任意の文字を設定することができます。
10. UNSTRING文の実行で転記処理が行われた後の制御の移行は、ON OVERFLOW指定およびNOT ON OVERFLOW指定の有無によって異なります。転記処理の後の制御の移行を、下表に示します。

ON OVERFLOW指定	NOT ON OVERFLOW指定	制御の移行	
		転記処理でオーバフロー条件が発生した場合(*3)	転記処理でオーバフロー条件が発生しなかった場合
あり	あり	[1]転記が終了し、無条件文-1に制御が移ります。 [2]無条件文-1の実行後、UNSTRING文の終わりに制御が移ります。(*1)	[1]すべての転記が終了した後、無条件文-2に制御が移ります。 [2]無条件文-2の実行後、UNSTRING文の終わりに制御が移ります。(*2)
あり	なし	[1]転記が終了し、無条件文-1に制御が移ります。 [2]無条件文-1の実行後、UNSTRING文の終わりに制御が移ります。(*1)	すべての転記が終了した後、UNSTRING文の終わりに制御が移ります。
なし	あり	転記が終了し、UNSTRING文の終わりに制御が移ります。	[1]すべての転記が終了した後、無条件文-2に制御が移ります。 [2]無条件文-2の実行後、UNSTRING文の終わりに制御が移ります。(*2)
なし	なし	転記が終了し、UNSTRING文の終わりに制御が移ります。	すべての転記が終了した後、UNSTRING文の終わりに制御が移ります。

*1 ： 無条件文-1に制御の明示移行を起こす手続き分岐または条件文を書いた場合、その文の規則に従って制御が移ります。

*2 ： 無条件文-2に制御の明示移行を起こす手続き分岐または条件文を書いた場合、その文の規則に従って制御が移ります。

*3 ： オーバフロー条件は、以下の場合に発生します。

- 一意名-4の初期値が、1より小さいかまたは一意名-3の文字列の文字数より大きいとき
- UNSTRING文の実行中、すべての受取り側が使われたのに、一意名-3のデータ項目中にまだ検査されていない文字が残っているとき

11. END-UNSTRING指定は、UNSTRING文の範囲を区切ります。
12. 以下の場合、この文の実行結果は規定されません。
    a. 一意名-1～一意名-3が、一意名-4～一意名-8と同じ記憶領域を占有する場合
    b. 一意名-4～一意名-6が、一意名-7または一意名-8と同じ記憶領域を占有する場合
    c. 一意名-7が一意名-8と同じ記憶領域を占有する場合

書き方2の規則

1. 一意名-9を「送出し側項目」、一意名-10を「受取り側項目」といいます。
2. 一意名-11のデータ項目の値は、一意名-9の領域中の文字位置を表します。
3. 一意名-12のデータ項目は、UNSTRING文の実行で作用を受けた一意名-10のデータ項目の1つにつき1ずつ増加するカウンタになります。

4. UNSTRING文を実行すると、一意名-9のデータ項目に格納されている文字列をBY指定の形式で分割して、一意名-10に転記の規則に従って転記します。一意名-10が集団項目の場合、従属するデータ項目の先頭から順に転記の規則に従って転記します。

5. 4.の転記において、送出し側と受取り側の字類が異なる場合、受取り側項目の字類に合わせてデータを変換します。データの変換処理で異常を検出した場合は、オーバフロー条件が生じます。

6. BY指定は、分割する形式を指定します。

   ー CSV-FORMATが指定された場合、一意名-9はCSV形式で分割します。CSV形式とは、複数の文字列データをカンマで区切った文字データのことです。このコンパイラでは、文字の扱いを以下とします。

   - 送出し側の分割データが引用符で囲まれた文字列の場合、引用符を取り除いて受取り側へ転記します。このとき、データ中に連続する2つの引用符が現れた場合、1つの引用符に置き換えます。

   - 送出し側のデータ中に2つの連続したカンマを検出した場合、対応する受取り側のデータ記述に従い、英数字または日本語なら空白、数字ならゼロで満たします。

   ー TSV-FORMATが指定された場合、一意名-9はTSV形式で分割します。TSV形式とは、複数の文字列データをタブで区切った文字データのことです。このコンパイラでは、文字の扱いを以下とします。

   - 送出し側の分割データが引用符で囲まれた文字列の場合、引用符を取り除いて受取り側へ転記します。このとき、データ中に連続する2つの引用符が現れた場合、1つの引用符に置き換えます。

   - 送出し側のデータ中に2つの連続したタブを検出した場合、対応する受取り側のデータ記述に従い、英数字または日本語なら空白、数字ならゼロで満たします。

7. POINTER指定を書くと、一意名-9の文字列は、一意名-11のデータ項目の内容で示される文字位置から検査されます。POINTER指定を書かないと、文字列は最左端の文字位置から検査されます。

8. POINTER指定およびTALLYING（分解計数）指定のデータ項目の初期値は、利用者が設定しておかなければなりません。

9. 一意名-11のデータ項目の内容は、一意名-9のデータ項目中の文字が1文字調べられるごとに1ずつ増加します。POINTER指定付きのUNSTRING文の実行が終了すると、一意名-11のデータ項目の内容は、その初期値に一意名-9のデータ項目中の検査された文字数を加えた値になります。

10. TALLYING指定付きのUNSTRING文の実行が終了すると、一意名-12のデータ項目の内容は、UNSTRING文の実行開始時の値にUNSTRING文の実行中に作用を受けた各受取り側項目の一意名-10の個数を加えた値になります。

11. UNSTRING文の実行で転記処理を行った後の制御の移行は、ON OVERFLOW指定およびNOT ON OVERFLOW指定の有無によって異なります。転記処理後の制御の移行を、下表に示します。

ON OVERFLOW指定	NOT ON OVERFLOW指定	制御の移行	
		転記処理でオーバフロー条件が発生した場合(*3)	転記処理でオーバフロー条件が発生しなかった場合
あり	あり	[1]転記が終了し、無条件文-3に制御が移ります。 [2]無条件文-3の実行後、UNSTRING文の終わりに制御が移ります。(*1)	[1]すべての転記が終了した後、無条件文-4に制御が移ります。 [2]無条件文-4の実行後、UNSTRING文の終わりに制御が移ります。(*2)
あり	なし	[1]転記が終了し、無条件文-3に制御が移ります。 [2]無条件文-3の実行後、UNSTRING文の終わりに制御が移ります。(*1)	すべての転記が終了した後、UNSTRING文の終わりに制御が移ります。
なし	あり	転記が終了し、UNSTRING文の終わりに制御が移ります。 ただし、オーバフロー条件(*3)のd)だった場合、転記を継続し、すべての転記が終了した後、UNSTRING文の終わりに制御が移ります。	[1]すべての転記が終了した後、無条件文-4に制御が移ります。 [2]無条件文-4の実行後、UNSTRING文の終わりに制御が移ります。(*2)

ON OVERFLOW指定	NOT ON OVERFLOW指定	制御の移行	
		転記処理でオーバフロー条件が発生した場合(*3)	転記処理でオーバフロー条件が発生しなかった場合
なし	なし	転記が終了し、UNSTRING文の終わりに制御が移ります。 ただし、オーバフロー条件(*3)のd)だった場合、転記を継続し、すべての転記が終了した後、UNSTRING文の終わりに制御が移ります。	すべての転記が終了した後、UNSTRING文の終わりに制御が移ります。

*1 ： 無条件文-3に制御の明示移行を起こす手続き分岐または条件文を書いた場合、その文の規則に従って制御が移ります。

*2 ： 無条件文-4に制御の明示移行を起こす手続き分岐または条件文を書いた場合、その文の規則に従って制御が移ります。

*3 ： オーバフローの条件は、以下の場合に発生します。

a. 一意名-11の初期値が1より小さいか、一意名-9のデータ項目の文字数より大きい

b. すべての受取り側が使われたのに、一意名-9のデータ項目中にまだ検査されていない文字が残っている

c. 転記時に受取り側の字類に合わせてデータ変換する際、異常を検出した

d. 受取り側項目への転記処理において、切捨てが発生した

12. END-UNSTRING指定は、UNSTRING文の範囲を区切ります。

13. 一意名-9が一意名-10、一意名-11もしくは一意名-12と同じ記憶領域を占有する場合、または一意名-11が一意名-12と同じ記憶領域を占有する場合、たとえそれらが同じデータ記述項によって定義されていても、この文の実行結果は規定しません。

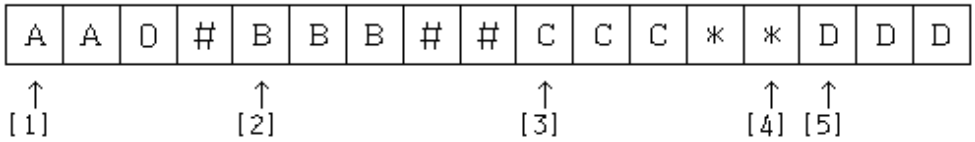
UNSTRING文の処理の流れ

以下のUNSTRING文を例に、処理の流れを説明します。

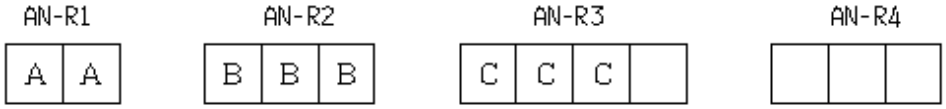
```
*〔データ部〕
  77  AN-SND PIC X(17) VALUE "AAO#BBB##CCC**DDD".
  77  AN-DEL PIC X     VALUE "*".
  77  AN-R1  PIC X(2)  VALUE "II".
  77  AN-R2  PIC X(3)  VALUE "JJJ".
  77  AN-R3  PIC X(4)  VALUE "KKKK".
  77  AN-R4  PIC X(3)  VALUE "LLL".
  77  AN-D2  PIC X(2)  VALUE "MM".
  77  AN-D4  PIC X     VALUE "N".
  77  ED-C3  PIC 99    VALUE 11.
  77  ED-C4  PIC 99    VALUE 22.
  77  ED-PTR PIC 99    VALUE  1.
  77  ED-TLY PIC 99    VALUE  0.
*〔手続き部〕
     UNSTRING AN-SND
         DELIMITED BY ALL "#" OR AN-DEL
         INTO AN-R1
              AN-R2 DELIMITER IN AN-D2
              AN-R3 COUNT IN ED-C3
              AN-R4 DELIMITER IN AN-D4
                    COUNT IN ED-C4
         WITH POINTER ED-PTR
         TALLYING IN ED-TLY
         ON OVERFLOW GO TO OVERFLOW-PROC.
```

このUNSTRING文を実行すると、送出し側項目(AN-SND)が分解され、順に受取り側項目(AN-R1、AN-R2、AN-R3およびAN-R4)に転記されます。

〔実行前のAN-SNDの内容〕



〔実行後の受取り側項目の内容〕



〔実行後の区切り文字の受取り側項目の内容〕

1. ED-PTRの初期値が1なので、AN-SNDの1文字目([1]の位置)以降が転記の対象になります。転記の対象となった文字列が、区切り文字で区切られて、送出し側項目に転記されます。区切り文字は、“#”の繰り返し、および1文字の“＊”です。
2. 最初に、[1]から始まり“#”が現れるまでの文字列がAN-R1に転記されます。転記の結果、ED-PTRの値は5([2]の位置)に変更されます。
3. 次に、[2]から始まり、“#”が現れるまでの文字列がAN-R2に転記されます。転記の結果、ED-PTRの値は10([3]の位置)に変更されます。また、区切り文字の繰返しの1回分が、AN-D2に転記されます。
4. 次に、[3]から始まり、“＊”が現れるまでの文字列がAN-R3に転記されます。転記の結果、ED-PTRの値は14([4]の位置)に変更されます。また、区切り文字までの文字の個数が3文字なので、ED-C3の値は3に変更されます。
5. 次に、[4]の文字位置が“＊”なので、空白がAN-R4に転記されます。転記の結果、ED-PTRの値は15([5]の位置)に変更されます。また、区切り文字までの文字の個数が0文字なので、ED-C4の値は0に変更されます。
6. UNSTRING文の実行が終了したときのED-PTRの値は、15です。AN-SNDの中に、まだ転記されていない文字が残っているので、OVERFLOW-PROCに制御が移ります。
7. 4つの受取り側項目(AN-R1、AN-R2、AN-R3およびAN-R4)に文字列が転記されたので、ED-TLYに4が加算され、ED-TLYの値は4に変更されます。

UNSTRING文で使用するデータを、以下のように定義したものと仮定します。

```
77  SEND-A PICTURE X(30).
77  RCV-1  PICTURE X(10).
77  RCV-2  PICTURE X(10).
77  RCV-3  PICTURE X(10).
77  DEL-1  PICTURE XX.
77  DEL-2  PICTURE XX.
77  DEL-3  PICTURE XX.
77  CTR-1  PICTURE 99.
77  CTR-2  PICTURE 99.
77  CTR-3  PICTURE 99.
77  PTR-A  PICTURE 99.
77  TALL-A PICTURE 99.
```

UNSTRING文を実行する前、各データ項目には以下の値が設定されていると仮定します。以降の説明で、“!”は1文字の空白を表します。

```
SEND-A:
          ┌──────────────────────────┐
          │MUNICH/NEW!YORK//TOKYO******** │
          └──────────────────────────┘
RCV-1、RCV-2、RCV-3、DEL-1、DEL-2およびDEL-3:  空白
CTR-1、CTR-2、CTR-3およびTALL-A:  00
PTR-A:  01
```

以下に、UNSTRING文の記述例と実行結果を示します。

例

例1

```
    UNSTRING SEND-A
        INTO RCV-1 RCV-2 RCV-3
        WITH POINTER PTR-A.
```

一意名	実行後の一意名の値
RCV-1	MUNICH/NEW
RCV-2	!YORK//TOK
RCV-3	YO********
PTR-A	31

例2

```
    UNSTRING SEND-A
        DELIMITED BY "/"
        INTO RCV-1 COUNT IN CTR-1
             RCV-2 COUNT IN CTR-2
             RCV-3 COUNT IN CTR-3
        WITH POINTER PTR-A TALLYING IN TALL-A.
```

一意名	実行後の一意名の値
RCV-1	MUNICH!!!!
RCV-2	NEW!YORK!!
RCV-3	!!!!!!!!!!
CTR-1	06
CTR-2	08
CTR-3	00
PTR-A	18
TALL-A	03

この例では、すべての受取り側項目が使われたのにもかかわらず、SEND-Aにはまだ検査されていない文字が残っています。この例のUNSTRING文にON OVERFLOW指定を書いた場合、ON OVERFLOW指定に書いた無条件文が実行されます。

例3

```
    UNSTRING SEND-A
        DELIMITED BY ALL "/" OR ALL "*"
```

```
        INTO RCV-1 DELIMITER IN DEL-1
                   COUNT IN CTR-1
             RCV-2 DELIMITER IN DEL-2
                   COUNT IN CTR-2
             RCV-3 DELIMITER IN DEL-3
                   COUNT IN CTR-3
        WITH POINTER PTR-A TALLYING IN TALL-A.
```

一意名	実行後の一意名の値
RCV-1	MUNICH!!!!
RCV-2	NEW!YORK!!
RCV-3	TOKYO!!!!!
DEL-1	/!
DEL-2	/!
DEL-3	*!
CTR-1	06
CTR-2	08
CTR-3	05
PTR-A	31
TALL-A	03

6.4.51 USE文（順ファイル・相対ファイル・索引ファイル・表示ファイル・報告書作成）

入出力誤り処理手続きの開始を指定します。

使い方の例については、“付録H サンプル集”の“H.43 USE文”を参照してください。

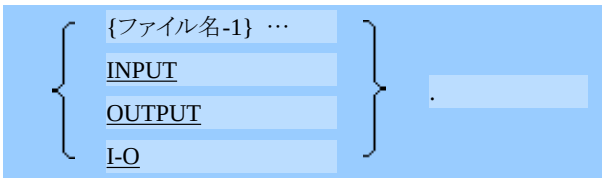
表示ファイルは、富士通拡張仕様です。

【書き方1】順ファイル・相対ファイル・索引ファイル

```
USE [GLOBAL] AFTER STANDARD
    { EXCEPTION }
    { ERROR     } PROCEDURE ON
    { {ファイル名-1} …  }
    { INPUT             }
    { OUTPUT            }
    { I-O               }
    { EXTEND            } .
```

【書き方2】表示ファイル

```
USE [GLOBAL] AFTER STANDARD
    { EXCEPTION } PROCEDURE ON
    { ERROR     }
```



【書き方3】報告書ファイル

```
USE [GLOBAL] AFTER STANDARD
    { EXCEPTION }
    { ERROR     } PROCEDURE ON
    { {ファイル名-1} …  }
    { OUTPUT           }
    { EXTEND           } .
```

[WinIPF][Winx64][LinuxIPF][Linux64][.NET]では報告書作成機能は使用できません。

構文規則

1. USE文は、手続き部の宣言部分の節の見出しに続けて書かなければなりません。USE文は、1つのUSE文だけで完結文でなければなりません。USE文の後に、USE手続きを定義するいくつかの段落を書きます。USE手続き段落は、省略することもできます。なお、ここでは、USE AFTER STANDARD EXCEPTION手続きを、「USE手続き」と略記します。
2. 2つ以上のUSE文に、ファイル名-1に同じファイル名を指定することによって、複数のUSE手続きが同時に実行されることがあってはなりません。
3. EXCEPTIONとERRORは同義語です。
4. ファイル名-1の並びに、ORGANIZATION句の指定が異なるファイルまたはACCESS MODE句の指定が異なるファイルを指定することもできます。また、ORGANIZATION句の指定が異なるファイルまたはACCESS MODE句の指定が異なるファイルを、USE文で暗に参照することもできます。
5. INPUT指定、OUTPUT指定、I-O指定またはEXTEND指定を書いたUSE文は、1つの手続き部の宣言部分では、それぞれただ1つだけ書くことができます。

一般規則

1. USE文は、USE手続きを実行する条件を指定します。USE文自身は実行されません。
2. USE手続きは、AT END指定(順ファイル、相対ファイル、索引ファイルまたは表示ファイル)またはINVALID KEY指定(相対ファイルまたは索引ファイル)が優先する場合を除いて、入出力文(暗黙に実行される入出力文も含む)の実行が不成功になったときに実行されます。USE手続きを実行する条件は、以下のように指定します。
    a. ファイル名-1指定は、ファイル名-1に対する入出力文の実行が不成功になったときにUSE手続きを実行することを指定します。
    b. INPUT指定は、入力モードで開かれているファイルに対する入出力文の実行中、または入力モードでファイルを開く処理が不成功になったときにUSE手続きを実行することを指定します。
    c. OUTPUT指定は、出力モードで開かれているファイルに対する入出力文の実行中、または出力モードでファイルを開く処理が不成功になったときにUSE手続きを実行することを指定します。
    d. I-O指定は、入出力両用モードで開かれているファイルに対する入出力文の実行中、または入出力両用モードでファイルを開く処理が不成功になったときにUSE手続きを実行することを指定します。
    e. EXTEND指定は、拡張モードで開かれているファイルに対する入出力文の実行中、または拡張モードでファイルを開く処理が不成功になったときにUSE手続きを実行することを指定します。

3. 宣言節は、一番外側のプログラムおよび内部プログラムのどのプログラムにも書くことができます。
4. 入れ子のプログラムで入出力文の実行が不成功になったときに実行される宣言節は、以下の規則をこの順に適用することによって選択されます。条件を満足するUSE文が見つかった場合、それに続くUSE手続きだけが実行されます。
    a. 不成功になった入出力文を書いたプログラムの中の、条件を満足するUSE文
    b. 不成功になった入出力文を書いたプログラムを直接または間接に含むプログラム中のGLOBAL指定を書いたUSE文で、条件を満足するUSE文
    該当するUSE文が2つ以上ある場合は、最も内側のプログラムの中のUSE文
5. USE文に指定した条件は、USE文を含む翻訳単位のプログラムが実行されているときに検査されます。別の翻訳単位に書いたUSE文の条件は検査されません。
6. USE手続きの中から手続き部分の手続きを参照してはいけません。
7. 宣言節の節名およびUSE手続きの段落名を、別の宣言節または手続き部分の手続きで参照する場合、PERFORM文によって参照しなければなりません。
8. USE文にファイル名-1を書いた場合、他のUSE文に書いた条件は、ファイル名-1に対して適用されません。
9. USE手続きの実行後、入出力状態が重大誤りでなければ、USE手続きを実行させた入出力文の次の実行文に制御が移ります。入出力状態が重大誤りの場合の制御の移行については、“NetCOBOL使用手引書”を参照してください。
10. USE手続きの実行が終了する前に、その手続きを再び実行させる可能性のある文を実行することはできません。

6.4.52 USE BEFORE REPORTING文（報告書作成）

報告集団を作成表示する直前に実行する手続きの開始を指定します。

【書き方】

USE BEFORE REPORTING 一意名-1

[WinIPF][Winx64][LinuxIPF][Linux64][.NET]では報告書作成機能は使用できません。

構文規則

1. USE BEFORE REPORTING文は、手続き部の宣言部分の節の見出しに続けて書かなければなりません。この文は、1つの文だけからなる完結文でなければなりません。この文の後にUSE BEFORE REPORTING手続きを定義するいくつかの段落を書きます。USE BEFORE REPORTING手続きは、省略することもできます。
2. 一意名-1は、データ部の報告節の報告集団でなければなりません。同じ一意名-1を、2つ以上のUSE BEFORE REPORTING文に指定することはできません。
3. GENERATE文、INITIATE文またはTERMINATE文を、USE BEFORE REPORTING手続きの中に書くことはできません。USE BEFORE REPORTING手続きの中のPERFORM文の実行範囲に、GENERATE文、INITIATE文またはTERMINATE文を書くことはできません。
4. USE BEFORE REPORTING手続きの中で、制御データ項目の値を変更することはできません。

一般規則

1. USE BEFORE REPORTING文は、USE BEFORE REPORTING手続きを実行する条件を指定します。USE BEFORE REPORTING文自身は、実行されません。

2. 宣言節は、一番外側のプログラムおよび内部プログラムのどのプログラムにも書くことができます。
3. USE BEFORE REPORTING手続きは、一意名-1に指定した報告集団を作成表示する直前に、報告書作成管理システムによって実行されます。
4. USE BEFORE REPORTING手続きの中から手続き部分の手続きを参照することはできません。
5. 宣言節の節名およびUSE BEFORE REPORTING手続きの段落名を、別の宣言節または手続き部分の手続きで参照する場合、PERFORM文によって参照しなければなりません。
6. USE BEFORE REPORTING手続きの実行が終了する前に、そのUSE BEFORE REPORTING手続きを再び実行させる可能性のある文を実行することはできません。

6.4.53 USE FOR DEAD-LOCK文

デッドロックが生じたときに実行すべき手続きを指定します。

【書き方】

USE FOR DEAD-LOCK

注意

この文は[DS][Win32][WinIPF][Winx64][Solaris][Linux][LinuxIPF][Linux64]で指定できます。

構文規則

1. USE FOR DEAD-LOCK文は、手続き部の宣言部分の節の見出しに続けて書き、それだけで完結文としなければなりません。この後に手続きを定義するいくつかの手続きを書いてください。手続き段落は、省略することができます。
2. USE FOR DEAD-LOCK文自身は、実行されるものではなく、USE FOR DEAD-LOCK手続きの実行の条件を指定するだけです。
3. USE FOR DEAD-LOCK文は、1つのプログラムに一度だけ指定できます。
4. USE FOR DEAD-LOCK文は、別のプログラムに含まれるプログラムに書くことはできません。

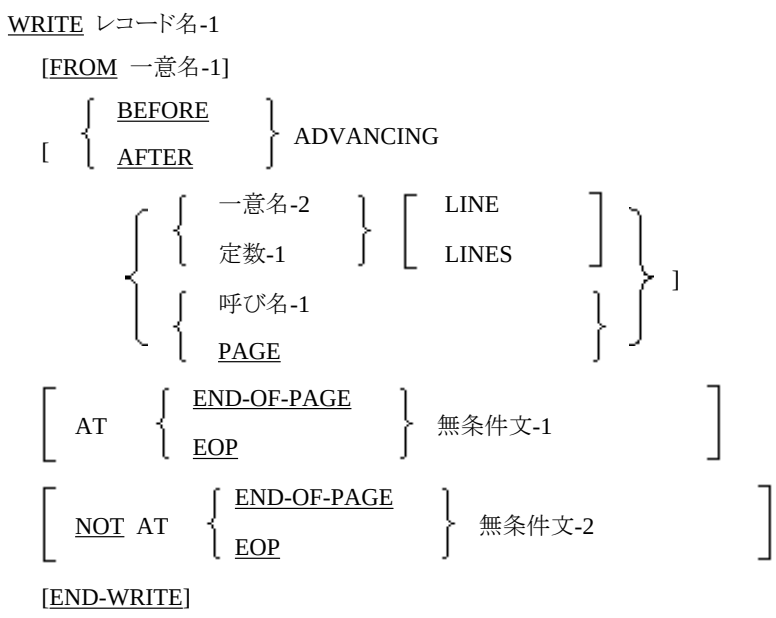
一般規則

1. 指定された手続きは、トランザクションコントロールシステムによって実行されます。
2. USE FOR DEAD-LOCK文に関連する手続き名は、PERFORM文によってだけ、他の宣言節中または手続き部分の手続き中で参照できます。
3. USE FOR DEAD-LOCK手続き実行後は、GO TO文などによって宣言部分以外の手続きを参照しなければならず、USE FOR DEAD-LOCK手続きの最後まで実行してはなりません。
4. USE FOR DEAD-LOCK手続きが最後まで実行されると、COBOLの提供する標準の処理手続きが実行されます。
5. USE FOR DEAD-LOCK手続きの中で、そのUSE FOR DEAD-LOCK手続きを再び実行させる可能性のある文を実行してはなりません。
6. ある実行単位中にUSE FOR DEAD-LOCK文が書かれたプログラムが2つ以上含まれている場合、USE FOR DEAD-LOCK文が記述されているプログラムの階層上で最近実行されたプログラムに制御が渡ります。
7. ある実行単位中にUSE FOR DEAD-LOCK文が指定されている場合の動作については、“NetCOBOL使用手引書”を参照してください。

6.4.54 WRITE文（順ファイル）

レコードのファイルへの書出しまたは論理ページの縦方向への行送りを行います。

【書き方】



構文規則

1. レコード名-1は、データ部のファイル節で定義したレコードの名前でなければなりません。レコード名-1は、ファイル名で修飾することができます。
2. 一意名-1にデータ項目を指定する場合、レコード名-1の領域と一意名-1の領域は同じであってはいけません。一意名-1に関数一意名を指定する場合、英数字関数でなければなりません。
3. 一意名-2は、整数項目でなければなりません。
4. 整数-1には、0を指定することもできます。
5. レコード名-1に関連するファイル記述項にLINAGE句を書いた場合、呼び名-1を書くことはできません。
6. 呼び名-1は、環境部の特殊名段落で機能名(CHANNEL-01～CHANNEL-12、SLC、CTL、STACKER-01、STACKER-02)に対応付けておかなければなりません。
7. 1つのWRITE文に、ADVANCING PAGE指定とEND-OF-PAGE指定の両方を書くことはできません。
8. END-OF-PAGE指定またはNOT END-OF-PAGE指定を書く場合、レコード名-1に関連するファイル記述項にLINAGE句を書かなければなりません。
9. END-OF-PAGEとEOPは同義語です。
10. 印刷ファイルに対するWRITE文にだけ、END-OF-PAGE指定を書くことができます。ただし、FORMAT句付きの印刷ファイルに対するWRITE文には、END-OF-PAGE指定を書くことはできません。
11. 印刷ファイルに対するWRITE文にだけADVANCING指定を書くことができます。

    [HP][Win32][WinIPF][Winx64][Solaris][LinuxIPF][Linux64][.NET]では、行順ファイルに対するWRITE文にもADVANCING指定を書くことができます。
12. FORMAT句付きの印刷ファイルに対するWRITE文にADVANCING指定を書く場合、機能名CTLに対応付けた呼び名を呼び名-1に指定することはできません。

13. CONTROL RECORDS句に指定したレコードをレコード名-1に指定する場合、ADVANCING指定を書くことはできません。

14. 一意名-1が、明にまたは暗にCHARACTER TYPE句またはPRINTING POSITION句を指定したデータ項目を従属する場合、一意名-1を部分参照することはできません。

一般規則

1. レコード名-1に関連するファイルは、WRITE文を実行する前に出力モードまたは拡張モードで開いておかなければなりません。

2. WRITE文の実行が成功すると、レコード名-1のレコードはレコード領域で使用不可能になります。ただし、入出力管理段落のSAME RECORD AREA句にレコード名-1に関連するファイルを指定した場合、WRITE文の実行によってレコードが使用不可能になることはありません。レコード名-1のレコードは、そのプログラムで使用可能です。

3. FROM指定付きのWRITE文の実行結果は、以下に示す文をこの順で実行した結果と同じです。なお、印刷結果については、一般規則20. にしたがいます。

    a. MOVE文の規則に従った以下の文

    ```
    MOVE  一意名-1  TO  レコード名-1
    ```

    b. FROM指定のない同じWRITE文

4. WRITE文の実行が完了した後、SAME RECORD AREA句にレコード名-1に関連するファイルを指定している場合を除いてレコード名-1の領域の情報は使用不可能になります。ただし、一意名-1の領域の情報は使用可能です。

5. WRITE文を実行しても、ファイル位置指示子は変更されません。

6. WRITE文を実行すると、レコード名-1に関連するファイルの入出力状態の値が更新されます。

7. WRITE文を実行すると、レコードが入出力管理システムに渡されます。

8. レコード名-1のレコードの文字位置の個数は、レコード名-1に関連するファイルの最大レコード長より大きくてはいけません。また、レコード名-1のレコードの文字位置の個数は、レコード名-1に関連するファイルの最小レコード長より小さくてはいけません。これらの条件を満足しない場合、WRITE文の実行は不成功になり、レコードは書き出されません。このとき、レコード名-1に関連するファイルのレコード領域は変更されません。レコード名-1に関連するファイルの入出力状態には、この状態の原因を示す値が設定されます。

    ただし、このコンパイラでは、最小レコードより小さいレコードを書き出した場合、WRITE文の実行は成功します。

9. END-WRITE指定は、WRITE文の範囲を区切ります。

10. 順ファイル内のレコードの順序は、ファイルを生成するときのWRITE文の実行順序によって定まり、ファイルの終わりにレコードを追加する場合以外に変わることはありません。

11. 拡張モードかつ共用モードで開いたファイルに対して、2つ以上のファイル結合子に対するWRITE文の実行によってレコードが追加される場合、追加されるレコードの並びは規定されません。

12. レコード名-1に関連するファイルを拡張モードで開いた場合、WRITE文を実行すると、そのファイルが出力モードで開かれているかのように、ファイルの終わりにレコードが追加されます。ファイル中にレコードが存在する場合、EXTEND指定のOPEN文の実行後の最初のWRITE文によって、ファイルの最後のレコードの直後にレコードが追加されます。

13. 外部で明または暗に定義されたファイル容量を超えてレコードを書き出そうとした場合、例外条件になり、以下の処理が行われます。レコード領域の内容は変更されません。

    a. レコード名-1に関連するファイルの入出力状態に、区域外書出しを示す値が設定されます。

    b. レコード名-1に関連するファイルに関連するUSE AFTER STANDARD EXCEPTION手続きを書いた場合、その手続きが実行されます。そして、その手続きの実行後、USE文の規則に従って制御が渡されます。USE AFTER EXCEPTION手続きを書かなかった場合、レコード名-1に関連するファイルにFILE STATUS句を指定したときは、WRITE文の最後に制御が移ります。USE AFTER EXCEPTION手続きもFILE STATUS句も書かなかったときの実行結果は、規定されません。

14. リール/ユニットファイルの終了になり、かつそのファイルの外部で定義された区域を超えていない場合、以下の処理が行われます。

    a. 標準の終わりリール/ユニットラベル手続きが実行されます。

    b. リール/ユニット交換が行われます。すなわち、ボリューム指示子が、そのファイルの次のリール/ユニットを指すように更新されます。

    c. 標準の始めリール/ユニットラベル手続きが実行されます。

15. 印刷ファイルに対するADVANCING指定とEND-OF-PAGE指定は、印字ページの表現の縦方向の行制御を行います。ADVANCING指定を省略した場合、AFTER ADVANCING 1 LINEを指定したものとみなされ、自動的に行送りが行われます。ADVANCING指定を書いた場合、行送りは以下のように行われます。

    a. 整数-1または一意名-2に正の数を設定した場合、その値に等しい行数だけ行送りされます。

    b. 一意名-2のデータ項目に負の数を設定した場合、結果は規定されません。

    c. 整数-1または一意名-2に0を設定した場合、行送りは行われません。

    d. 呼び名-1を書いた場合、呼び名-1に対応付けた機能名に従って、チャネル1～12へスキップされたり行送りが抑制されたりします。呼び名-1に対応付ける機能名の意味は、以下のとおりです。

        - 機能名SLCは、行送りの抑制を意味します。

        - 機能名CHANNEL-01～CHANNEL-12は、それぞれチャネル1からチャネル12にスキップすることを意味します。チャネル1は、改ページすることを意味します。

        - 機能名CTLは、ページの形式を定義する制御レコードの書出しを意味します。制御レコードの書出しについては、一般規則の19.を参照してください。

    e. BEFORE指定を書いた場合、a.～d.の規則に従って行送りする前に、行が書き出されます。

    f. AFTER指定を書いた場合、a.～d.の規則に従って行送りした後に、行が書き出されます。

    g. ADVANCING PAGE指定を書き、レコード名-1に関連するファイル記述項にLINAGE句を書いた場合、LINAGE句で指定した次の論理ページの本体の最初の行に印字位置を位置付ける前(BEFORE)または後(AFTER)に、レコードが論理ページに書き出されます。

    h. ADVANCING PAGE指定を書き、レコード名-1に関連するファイル記述項にLINAGE句を書かなかった場合、次の物理ページに印字位置を位置付ける前(BEFORE)または後(AFTER)に、レコードが物理ページに書き出されます。

    i. FORMAT句付きの印刷ファイルに対するWRITE文にAFTER指定を書いた場合、装置は画面帳票定義体で定める行に位置付けられます。

16. [DS][Win16][Linux]では、行順ファイルに対してADVANCING指定付きのWRITE文を実行した場合の実行結果については、規定されていません。

    [HP][Win32][WinIPF][Winx64][Solaris][LinuxIPF][Linux64][.NET]では、行順ファイルに対してADVANCING指定付きのWRITE文を実行した場合、そのADVANCING指定に対応した制御文字がファイル中に出力されます。ADVANCING指定を書いた場合、以下のように制御文字がファイル中に出力されます。

    a. 整数-1または一意名-2に正の数を設定した場合、その値に等しいだけの改行コードがファイル中に出力されます。

    b. 一意名-2のデータ項目に負の数を設定した場合、結果は規定されません。

    c. PAGEを書いた場合、改ページコードがファイル中に出力されます。

    d. 呼び名-1を書いた場合、呼び名-1に対応付けた機能名に従って、以下のように動作します。

        - 機能名SLCおよびCTLは、整数-1または一意名-2に0を設定した場合と同じ動作となります。

        - 機能名CHANNEL-01は、PAGEを書いた場合と同じ動作となります。

        - 機能名CHANNEL-02～CHANNEL-12は、整数-1または一意名-2に1を設定した場合と同じ動作となります。

e. BEFORE指定を書いた場合、a.～d.の規則に従って制御コードを出力する前に、レコード名-1のレコードがファイル中に出力されます。

f. AFTER指定を書いた場合、a.～d.の規則に従って制御コードを出力した後に、レコード名-1のレコードの後ろに復帰コードを付加したレコードがファイル中に出力されます。

g. 整数-1または一意名-2に0を設定した場合、BEFORE指定またはAFTER指定に関係なく、レコード名-1のレコードの前後に復帰コードを付加したレコードがファイル中に出力されます。

17. 制御レコードを書き出す場合、以下の規則に従うWRITE文を実行しなければなりません。

- FORMAT句付きの印刷ファイルに対するWRITE文の場合、レコード名-1は、CONTROL RECORDS句に指定したレコードでなければなりません。
- FORMAT句なしの印刷ファイルに対するWRITE文の場合、呼び名-1は、機能名CTLに対する呼び名でなければなりません。

制御レコードの形式については、“NetCOBOL使用手引書”を参照してください。

18. 制御レコードに対するWRITE文の直後のWRITE文は、AFTER ADVANCING PAGE指定付きのWRITE文でなければなりません。制御レコードに対するWRITE文を実行すると、次のページの属性が制御レコードに指定した属性になります。ページの属性は、次に制御レコードを書き出すまで変更されません。

19. レコード名-1のレコード記述項または一意名-1のデータ記述項に、CHARACTER TYPE句またはPRINTING POSITION句を書いた場合、下表の規則に従ってレコードが書き出されます。

FROM指定の有無	レコード名-1のレコード記述項でのCHARACTER TYPE句句またはPRINTING POSITION句の有無	一意名-1のデータ記述項でのCHARACTER TYPE句句またはPRINTING POSITION句の有無	適用規則
なし	あり	---	レコード名-1のレコード記述項に書いたCHARACTER TYPE句およびPRINTING POSITION句に従います。
あり	なし	あり	一意名-1のデータ記述項に書いたCHARACTER TYPE句およびPRINTING POSITION句に従います。
	あり		
	あり	なし	レコード名-1のレコード記述項に書いたCHARACTER TYPE句およびPRINTING POSITION句は無視されます。

20. LINAGE句を指定していないファイルに対するWRITE文の実行中に例外条件が発生した場合、またはLINAGE句を指定したファイルに対するWRITE文の実行中にページ終了条件以外の例外条件が発生した場合、以下の処理が順に行われます。

a. レコード名-1に関連するファイルの入出力状態が設定されます。

b. レコード名-1に関連するファイルに関連するUSE AFTER STANDARD EXCEPTION手続きを書いた場合、その手続きが実行されます。そして、その手続きの実行後、USE文の規則に従って制御が渡されます。USE AFTER EXCEPTION手続きを書かなかった場合、レコード名-1に関連するファイルにFILE STATUS句を指定したときは、WRITE文の最後に制御が移ります。USE AFTER EXCEPTION手続きもFILE STATUS句も書かなかったときの実行結果は、規定されません。

21. WRITE文の実行中に例外条件が発生しなかった場合、以下の処理が順に行われます。

a. レコード名-1に関連するファイルの入出力状態が設定されます。

b. レコードが書き出されます。

c. NOT AT END-OF-PAGE指定を書いた場合、無条件文-2に制御が移ります。そして、無条件文-2の実行後、WRITE文の最後に制御が移ります。ただし、無条件文-2で制御の明示移行を起こす手続き分岐または条件文を実行した場合、その文の規則に従って制御が移ります。NOT AT END-OF-PAGE指定を省略した場合は、WRITE文の最後に制御が移ります。

LINAGE句を指定したファイルのページ終了/ページあふれ条件の規則

1. END-OF-PAGE指定付きのWRITE文の実行中に、印字ページの表現が論理的な終わりに達すると、無条件文-1が実行されます。論理的な終わりは、レコード名-1に関連するファイル記述項のLINAGE句で指定します。
2. ファイル記述項でLINAGE句を指定したファイルに対して、WRITE文を実行すると、ページ終了条件またはページあふれ条件が発生することがあります。ページ終了条件は、WRITE文にEND-OF-PAGE指定を書いた場合にだけ発生します。
   a. ページ終了条件は、ページ本体の脚書き領域内に印字または行送りを行うときに起きます。これは、WRITE文の行送りによってLINAGE-COUNTERの値が、LINAGE句のFOOTING指定の整数-2またはデータ名-2の値と等しいか大きくなったときです。この場合、WRITE文の実行後、END-OF-PAGE指定の無条件文-1が実行されます。
   b. ページあふれ条件は、WRITE文の行送りが、現在のページ本体中に完全に収まらないときに起きます。これは、WRITE文の行送りによってLINAGE-COUNTERの値が、LINAGE句の整数-1またはデータ名-1の値を超えるときです。この場合、ADVANCING指定に従って、LINAGE句に指定した次の論理ページの本体の最初の行に装置が位置付けられる前(BEFORE)または後(AFTER)に、レコードが論理ページに書き出されます。END-OF-PAGE指定を書いた場合、レコードが書き出された後、無条件文-1が実行されます。
   c. WRITE文の行送りによって、LINAGE-COUNTERの値が、LINAGE句のFOOTING指定の整数-2またはデータ名-2の値と、LINAGE句の整数-1またはデータ名-1の値を同時に超える場合は、ページあふれ条件が起こります。

FORMAT句付きの印刷ファイルに対するWRITE文の規則

1. 制御レコードの帳票定義体名フィールドに帳票定義体名を設定し、制御レコードに対するWRITE文を実行すると、次のページは固定形式ページになります。
2. 制御レコードの帳票定義体名フィールドに空白を設定し、制御レコードに対するWRITE文を実行すると、次のページは不定形式ページになります。FORMAT句付きの印刷ファイルを開いた直後も、次のページは不定形式ページであるとみなされます。
3. 図表レコードを書き出す場合、FORMAT句に指定したデータ項目に帳票定義体名を設定しなければなりません。このとき、図表レコードは、帳票定義体名で定義される固定形式ページへの出力となり、帳票定義体に従って編集されて出力されます。
4. 行レコードを書き出す場合、FORMAT句に指定したデータ項目に空白を設定しなければなりません。
5. 固定形式ページにレコードを出力する場合、GROUP句に指定したデータ項目に項目群名または項目名を設定しなければなりません。
6. 図表レコードを書き出す場合、FORMAT句のデータ項目に設定されている帳票定義体名と、その時点で有効となっている帳票定義体名が異なるとき、AFTER ADVANCING PAGE指定付きのWRITE文を実行しなければなりません。
7. ページ上の先頭の位置に図表レコードを書き出す場合、ADVANCING指定を書くならば、AFTER ADVANCING PAGE指定を書かなければなりません。
8. 図表レコードを書き出した後は、帳票定義体で定義した最後の行に行送りされているものとみなされます。
9. 不定形式ページに図表レコードを書き出す場合、ADVANCING指定を評価した後の出力位置に図表レコード全体が出力できないときは、例外条件になります。

6.4.55 WRITE文（相対ファイル・索引ファイル）

レコードのファイルへの書出しを行います。

【書き方】

WRITE レコード名-1

[FROM 一意名-1]

[INVALID KEY 無条件文-1]

[NOT INVALID KEY 無条件文-2]

[END-WRITE]

構文規則

1. レコード名-1は、データ部のファイル節で定義したレコードの名前でなければなりません。レコード名-1は、ファイル名で修飾することができます。
2. 一意名-1にデータ項目を指定する場合、レコード名-1の領域と一意名-1の領域は同じであってはいけません。一意名-1に関数一意名を指定する場合、英数字関数でなければなりません。
3. レコード名-1に関連するファイルに関連するUSE AFTER STANDARD EXCEPTION手続きを書かない場合、INVALID KEY指定を書かなければなりません。

   ただし、このコンパイラでは、USE AFTER STANDARD EXCEPTION手続きとINVALID KEY指定の両方を省略することができます。

一般規則

相対ファイルおよび索引ファイルに共通する規則

1. レコード名-1に関連するファイルは、WRITE文を実行する前に出力モード、入出力両用モードまたは拡張モードで開いておかなければなりません。
2. WRITE文の実行が成功すると、レコード名-1のレコードはレコード領域で使用不可能になります。ただし、入出力管理段落のSAME RECORD AREA句にレコード名-1に関連するファイルを指定した場合、WRITE文によってレコードが使用不可能になることはありません。レコード名-1のレコードは、そのプログラムで使用可能です。
3. FROM指定付きのWRITE文の実行結果は、以下に示す文をこの順で実行した結果と同じです。

   ― MOVE文の規則に従った以下の文

   ```
   MOVE  一意名-1  TO  レコード名-1
   ```

   ― FROM指定のない同じWRITE文
4. WRITE文の実行が完了した後、SAME RECORD AREA句にレコード名-1に関連するファイルを指定している場合を除いてレコード名-1の領域の情報は使用不可能になります。ただし、一意名-1の領域の情報は使用可能です。
5. WRITE文を実行しても、ファイル位置指示子は変更されません。
6. WRITE文を実行すると、レコード名-1に関連するファイルの入出力状態の値が更新されます。
7. WRITE文を実行するとレコードが入出力管理システムに渡されます。
8. レコード名-1のレコード中の文字位置の個数は、レコード名-1に関連するファイルの最大レコード長より大きくてはいけません。また、レコード名-1のレコード中の文字位置の個数は、レコード名-1に関連するファイルの最小レコード長より小さくてはいけません。これらの条件を満足しない場合、WRITE文の実行は不成功になり、レコードは書き出されません。このとき、レコード名-1に関連するファイルのレコード領域は変更されません。レコード名-1に関連するファイルの入出力状態には、この状態の原因を示す値が設定されます。

   ただし、このコンパイラでは、最小レコードより小さいレコードを書き出した場合、WRITE文の実行は成功します。
9. WRITE文の実行中に無効キー条件が発生すると、WRITE文の実行は不成功になります。レコード名-1に関連するファイルのレコード領域は変更されません。レコード名-1に関連するファイルの入出力状態に無効キー条件を示す値が設定された後、“6.3.12 INVALID KEY指定”の規則に従って制御が移ります。

10. WRITE文の実行中に無効キー条件以外の例外条件が発生すると、レコード名-1に関連するファイルの入出力状態が設定された後、“6.3.12 INVALID KEY指定”の規則に従って制御が移ります。
11. WRITE文の実行中に無効キー条件もその他の例外条件も発生しなかった場合、以下の処理が順に行われます。
    a. レコード名-1に関連するファイルの入出力状態が設定されます。
    b. レコードが書き出されます。
    c. “6.3.12 INVALID KEY指定”の規則に従って制御が移ります。
12. 拡張モードかつ共用モードで開いたファイルに対し、2つ以上のファイル結合子に対するWRITE文の実行によってレコードが追加される場合、追加されるレコードの並びは、規定されません。
13. レコード名-1に関連するファイル管理記述項のLOCK MODE句にAUTOMATICを指定した場合、WRITE文の実行が成功すると、存在するレコードのロックが解除されます。

相対ファイルの規則

1. WRITE文を実行しても、ファイル位置指示子は変更されません。
2. 出力モードで開いた順呼出し法のファイルに対してWRITE文を実行すると、相対レコード番号が自動的に設定され、レコード名-1のレコードが書き出されます。相対レコード番号は、最初のレコードから順に、1、2、3、…というように設定されます。レコード名-1に関連するファイルのACCESS MODE句にRELATIVE KEY指定を書いた場合、書き出されたレコードの相対レコード番号が、入出力管理システムによって、相対キー項目に設定されます。
3. 出力モードで開いた乱呼出し法のファイル、または出力モードで開いた動的呼出し法のファイルに対してWRITE文を実行する場合、WRITE文を実行する前に、書き出すレコードの相対レコード番号を、相対キー項目に設定しなければなりません。
4. 拡張モードで開いたファイルに対してWRITE文を実行する場合、そのファイルは順呼出し法のファイルでなければなりません。WRITE文を実行すると、レコード名-1のレコードがファイルに追加されます。最初のWRITE文のレコードの相対レコード番号は、そのファイルに存在する最大の相対レコード番号より1つ大きい値です。それに続いて書き出されるレコードは、連続的に大きくなる相対レコード番号を持ちます。レコード名-1に関連するファイルのACCESS MODE句にRELATIVE KEY指定を書いた場合、書き出されたレコードの相対レコード番号が、相対キー項目に設定されます。
5. 入出力両用モードで開いたファイルに対してWRITE文を実行する場合、そのファイルは乱呼出し法または動的呼出し法のファイルでなければなりません。また、WRITE文を実行する前に、書き出すレコードの相対レコード番号を、相対キー項目に設定しなければなりません。WRITE文を実行すると、レコード名-1のレコードがファイルに挿入されます。
6. 以下の場合、無効キー条件が発生します。
    ― 乱呼出し法または動的呼出し法のファイルに対するWRITE文の場合、相対キー項目の値と一致するレコードが、ファイルの中に存在する場合
    ― 外部で明または暗に定義されたファイルの区域を超えてレコードを書き出そうとした場合
    ― 相対レコード番号の有効桁数が、相対キー項目の大きさより大きい場合

索引ファイルの規則

1. WRITE文を実行しても、ファイル位置指示子は変更されません。

   ただし、以下の場合、ファイル位置指示子は不定になります。

   ― レコード名-1に関連するファイルのRECORD KEY句にDUPLICATES指定を書いた場合
   ― REVERSED ORDER指定付きのSTART文によりファイル位置指示子が確定した場合
2. WRITE文を実行する前に、必要な値を主レコードキー項目に設定しなければなりません。
3. レコード名-1に関連するファイルのRECORD KEY句にDUPLICATES指定を書かなかった場合、主レコードキー項目に設定する値は、ファイル中のすべてのレコードをとおして一意でなければなりません。

   RECORD KEY句にDUPLICATES指定を書いた場合は、主レコードキー項目に設定する値は一意である必要はありません。

4. WRITE文を実行すると、レコード名-1のレコード領域の内容が書き出されます。入出力管理システムは、任意のレコードキーに基づいて後でレコードを呼び出せるようにするために、レコードキー項目の内容を使います。
5. 順呼出し法のファイルにレコードを書き出す場合、以下の規則に従って主レコードキー項目に値を設定しなければなりません。

   ― ファイルを出力モードで開いた場合

   RECORD KEY句にDUPLICATES指定を書かなかった場合、主レコードのキー項目の大小順序に従って主レコードキーの値が昇順になるように、主レコードキー項目の値を設定しなければなりません。

   RECORD KEY句にDUPLICATES指定を書いた場合は、直前に書き出したレコードの主レコードキーの値と同じ値を設定することもできます。

   ― ファイルを拡張モードで開いた場合

   RECORD KEY句にDUPLICATES指定を書かなかった場合、主レコードキー項目に、ファイル中にある最大の主レコードキーの値より大きい値を設定しなければなりません。

   RECORD KEY句にDUPLICATES指定を書いた場合は、ファイル中にある最大の主レコードキーの値と同じ値を設定することもできます。
6. 乱呼出し法または動的呼出し法のファイルにレコードを書き出す場合、任意の順序でレコードを書き出すことができます。
7. レコード名-1に関連するファイルのALTERNATE RECORD KEY句にDUPLICATES指定を書いた場合、副レコードキー項目の値は、そのファイル中のすべてのレコードをとおして一意である必要はありません。この場合、NEXT指定付きのREAD文で、レコードを書き出した順にレコードを読むことができるように、レコードが格納されます。
8. 以下の場合、無効キー条件が発生します。

   ― DUPLICATES指定なしのRECORD KEY句を指定した順呼出し法のファイルに対するWRITE文で、主レコードキーの値が直前のレコードの主レコードキーの値と同じか小さい場合

   ― DUPLICATES指定付きのRECORD KEY句を指定した順呼出し法のファイルに対するWRITE文で、主レコードキーの値が直前のレコードの主レコードキーの値より小さい場合

   ― DUPLICATES指定なしのRECORD KEY句を指定した乱呼出し法または動的呼出し法のファイルに対するWRITE文で、主レコードキーの値がファイル中に既に存在するレコードの主レコードキーの値と同じ場合

   ― DUPLICATES指定なしのALTERNATE RECORD KEY句を指定した乱呼出し法または動的呼出し法のファイルに対するWRITE文で、副レコードキーの値がファイル中に既に存在するレコードの副レコードキーの値と同じ場合

   ― 外部で明または暗に定義されたファイルの区域を超えてレコードを書き出そうとした場合

6.4.56 WRITE文（表示ファイル）

レコードのファイルへの書出しを行います。

【書き方】

WRITE レコード名-1 [FROM 一意名-1]
[END-WRITE]

構文規則

1. レコード名-1は、データ部のファイル節で定義したレコードの名前でなければなりません。レコード名-1は、ファイル名で修飾することができます。

2. 一意名-1にデータ項目を指定する場合、レコード名-1の領域と一意名-1の領域は同じであってはいけません。一意名-1に関数一意名を指定する場合、英数字関数でなければなりません。

一般規則

1. レコード名-1に関連するファイルは、WRITE文を実行する前に、出力モードまたは入出力モードで開いておかなければなりません。
2. WRITE文の実行が成功すると、レコード名-1のレコードはレコード領域で使用不能になり ます。ただし、入出力管理段落のSAME RECORD AREA句にレコード名-1に関連するファイルを指定した場合、WRITE文によってレコードが使用不可能になることはありません。レコード名-1のレコードは、そのプログラムで使用可能です。
3. FROM指定付きのWRITE文の実行結果は、以下に示す文をこの順に実行した結果と同じです。

   ー MOVE文の規則に従った文

   ```
   MOVE  一意名-1  TO  レコード名-1
   ```

   ー FROM指定のない同じWRITE
4. WRITE文の実行が完了した後、SAME RECORD AREA句にレコード名-1に関連するファイルを指定している場合を除いてレコード名-1の領域の情報は使用不可能になります。ただし、一意名-1の領域の情報は使用可能です。
5. WRITE文を実行すると、レコード名-1に関連するファイルの入出力状態の値が更新されます。
6. WRITE文を実行すると、レコードが入出力管理システムに渡されます。

6.5 関数の全般規則

関数は、いくつかのデータを引数として特定の処理を呼び出すことにより、その処理の結果の値(関数値)を一時的なデータ項目に格納してプログラムに返却する機能です。

6.5.1 関数の呼出し形式

関数名は、以下の形式で書きます。この形式の記述を、「関数一意名」といいます。

関数名は、関数の名前です。関数名は、COBOLの語の1つです。

引数は、関数値を求めるための値です。引数には、関数によって範囲などの制限を設けることがあります。引数の個数、引数の指定順序および引数の値は、関数によって異なります。

関数一意名は、手続き部にだけ書くことができます。

【書き方】

FUNCTION 関数名-1 [({引数-1} …)] [部分参照子]

6.5.2 引数の型

引数には、一意名、算術式または定数を指定することができます。引数に指定できるデータは、引数の型によって決められています。

引数の型は、5種類があります。

- 英字
- 英数字
- 数字

- 整数
- 日本語

引数は、以下の規則に従って指定しなければなりません。

1. 引数の型が英字の場合、引数として、字類が英字項目または英字だけを含む文字定数を指定しなければなりません。関数値を求めるために、引数の大きさが用いられることがあります。
2. 引数の型が英数字の場合、引数として、英字項目、英数字項目、英数字編集項目、数字編集項目または文字定数を指定しなければなりません。関数値を求めるために、引数の大きさが用いられることがあります。
3. 引数の型が数字の場合、引数として算術式を指定しなければなりません。関数値を求めるために、算術式の値(符号を含む)が用いられます。
4. 引数の型が整数の場合、引数として、その値が常に整数値となる算術式を指定しなければなりません。関数値を求めるために、算術式の値(符号を含む)が用いられます。
5. 引数の型が日本語の場合、字類が日本語の基本データ項目を指定しなければなりません。関数値を求めるために、引数の大きさが用いられることがあります。

関数によっては、上記の5種類以外のデータ項目または定数を指定できるところがあります。

引数の値の範囲は、関数ごとに決まっています。定められた範囲を超える値を引数に設定した場合、その関数値は規定されません。

6.5.3 引数に表を指定する場合の規則

引数の繰り返しに、ある次元のすべての表要素を書く場合、添字付きのデータ名を1つずつ書くかわりに、添字にALLを指定したデータ名を書くことができます。添字にALLを書くと、その次元のすべての表要素を、出現順に従って左から右へ指定したかのように扱われます。

例えば、表を以下のように定義した場合、(a)と(b)、(c)と(d)はそれぞれ等価です。

```
01  TBL.
    02  A OCCURS 3.
        03  B OCCURS 4.
            04  C PIC X.
```

(a) FUNCTION MAX (A(1) A(2) A(3))
(b) FUNCTION MAX (A(ALL))
(c) FUNCTION MAX (C(1 1) C(1 2) C(1 3) C(1 4))
(d) FUNCTION MAX (C(1 ALL))

添字のALLがDEPENDING ON指定付きのOCCURS句に関連する場合、添字のALLの上限はDEPENDING ON指定のデータ名の値に従います。

添字のALLの評価の結果、1つ以上の表要素を参照することにならなければ、関数値は規定されません。

6.5.4 関数の型

関数値が格納される一時的なデータ項目の型を、関数の型といいます。関数一意名が書ける場所は、関数の型によって決められています。

関数には5種類の型があります。

- 英数字
- 数字
- 整数
- ポインタデータ
- 日本語

関数の型が英数字、数字、整数である関数を、それぞれ「英数字関数」、「数字関数」、「整数関数」といいます。

関数の型がポインタデータ、日本語である関数を、それぞれ「ポインタデータ関数」、「日本語関数」といいます。

英数字関数

英数字関数の一時的なデータ項目の項類と字類は英数字です。そのデータ項目の用途は表示用です。

英数字関数の関数一意名は、手続き部で英数字項目が書けるところにだけ書くことができます。ただし、受取り側項目に書くことはできません。

英数字関数の関数値の文字位置の個数は、関数の種類によって決められています。

数字関数

数字関数の一時的なデータ項目の項類と字類は数字です。そのデータ項目は符号付きです。

数字関数の関数一意名は、算術式の中にだけ書くことができます。ただし、整数を指定しなければならないところに書くことはできません。

数字関数の関数値の属性と精度は、関数の種類によって決められています。

整数関数

整数関数の一時的なデータ項目の項類と字類は数字です。そのデータ項目は符号付きです。

整数関数の関数一意名は、算術式の中で整数を指定しなければならないところにだけ書くことができます。

整数関数の関数値の属性と精度は、関数の種類によって決められています。

ポインタデータ関数

ポインタデータ関数の一時的なデータ項目は、用途がポインタデータです。

ポインタデータ関数の関数一意名は、手続き部でポインタデータ項目が書けるところにだけ書くことができます。ただし、受取り側項目に書くことはできません。

日本語関数

日本語関数の字類および項類は、日本語です。

日本語関数の関数一意名は、手続き部で日本語項目が書けるところにだけ書くことができます。ただし、受取り側項目に書くことはできません。

日本語関数の関数値の文字位置の個数は、関数の種類によって決められています。

6.6 関数

この節では、各関数について説明します。

参照

[.NET]固有の関数については、“11.10.2 組込み関数”を参照してください。

6.6.1 ACOS関数

引数-1の逆余弦を返します。

【書き方】

FUNCTION ACOS (引数-1)

引数

1. 引数-1の型は、数字でなければなりません。
2. 引数-1の値は、以下の範囲でなければなりません。

```
-1 ≦ "引数-1" ≦ +1
```

関数値

1. 関数値は、引数-1の逆余弦の近似値です。関数値の単位はラジアンです。
2. 関数値の範囲は、以下のとおりです。

```
0 ≦ 関数値 ≦ π
```

関数の型

関数の型は、数字です。

6.6.2 ADDR関数

引数-1の先頭アドレスを返します。

使い方の例については、“付録H サンプル集”の“H.47 ADDR関数”を参照してください。

【書き方】

FUNCTION ADDR (引数-1)

引数

引数-1は、内部ブール項目以外のデータ項目でなければなりません。

関数値

関数値は、引数-1の先頭アドレスです。

関数の型

関数の型は、ポインタデータです。

ADDR関数の固有の規則

ADDR関数は、以下の場所にだけ書くことができます。

- ポインタ修飾子
- DISPLAY文
- MOVE文の送出し側
- IF文およびEVALUATE文の比較条件

DISPLAY文にADDR関数を書いた場合、引数-1の先頭アドレスが、16進表現で表示されます。

6.6.3 ANNUITY関数

引数-1を利率、引数-2を均等払いの期間、元金を1として、毎期の均等払い額を返します。

【書き方】

FUNCTION ANNUITY (引数-1 引数-2)

引数

1. 引数-1の型は、数字でなければなりません。
2. 引数-1の値は、ゼロ以上でなければなりません。
3. 引数-2の型は、整数でなければなりません。
4. 引数-2の値は、正の整数でなければなりません。

関数値

関数値は、以下のとおりです。

・ 引数-1が0の場合、関数値は以下の式の近似値です。

```
1 / "引数-2"
```

・ 引数-1が0でない場合、関数値は以下の式の近似値です。

```
"引数-1" / (1 - (1 + "引数-1") ** (- "引数-2"))
```

関数の型

関数の型は、数字です。

6.6.4 ASIN関数

引数-1の逆正弦を返します。

【書き方】

FUNCTION ASIN (引数-1)

引数

1. 引数-1の型は、数字でなければなりません。
2. 引数-1の値は、以下の範囲でなければなりません。

```
-1 ≦ "引数-1" ≦ +1
```

関数値

1. 関数値は、引数-1の逆正弦の近似値です。関数値の単位はラジアンです。
2. 関数値の範囲は、以下のとおりです。

```
-π / 2 ≦ 関数値 ≦ +π / 2
```

関数の型

関数の型は、数字です。

6.6.5 ATAN関数

引数-1の逆正接を返します。

【書き方】

FUNCTION ATAN (引数-1)

引数

引数-1の型は、数字でなければなりません。

関数値

1. 関数値は、引数-1の逆正接の近似値です。関数値の単位はラジアンです。
2. 関数値の範囲は、以下のとおりです。

```
-π / 2 ＜ 関数値 ＜ +π / 2
```

関数の型

関数の型は、数字です。

6.6.6 CAST-ALPHANUMERIC関数

引数-1の日本語または日本語編集の文字列を英数字の文字列として返します。

【書き方】

FUNCTION CAST-ALPHANUMERIC (引数-1)

引数-1は、長さが少なくとも1文字であって、かつその字類は日本語でなければなりません。

関数値

1. 関数値は、それぞれの日本語文字が英数文字とみなされ、引数-1と同じ文字列を返します。項類および字類が英数字に変わるだけで、文字の変換は行われません。
2. 関数値の文字数は引数-1の文字数の倍になります。

関数の型

関数の型は英数字です。

CAST-ALPHANUMERIC関数固有の規則

この関数は、動作モードがUnicode(ISO/IEC 10646-1)の場合、使用できません。

6.6.7 CHAR関数

文字の大小順序における、引数-1の順序位置にある1文字を返します。

【書き方】

FUNCTION CHAR (引数-1)

引数

1. 引数-1の型は、整数でなければなりません。
2. 文字の大小順序における最大の順序位置をnとすると、引数-1の値は、1～nでなければなりません。

関数値

1. 関数値は、文字の大小順序における、引数-1の順序位置にある1文字です。文字の大小順序とは、特殊名段落のALPHABET句で定義したものです。ALPHABET句を省略した場合は、計算機文字集合における文字の大小順序に従って、関数値が求められます。
2. 文字の大小順序において、引数-1の順序位置に2つ以上の文字がある場合は、ALPHABET句でその順序位置に指定した先頭の定数の文字が、関数値として返されます。

関数の型

関数の型は、英数字です。

6.6.8 COS関数

引数-1の余弦を返します。

使い方の例については、“付録H サンプル集”の“H.45 COS関数”を参照してください。

【書き方】

FUNCTION COS (引数-1)

引数

1. 引数-1の型は、数字でなければなりません。
2. 引数-1の値は、単位をラジアンとして設定します。

関数値

1. 関数値は、引数-1の正弦の近似値です。
2. 関数値の範囲は、以下のとおりです。

```
-1 ≦ 関数値 ≦ +1
```

関数の型

関数の型は、数字です。

6.6.9 CURRENT-DATE関数

現在の日付と時刻を返します。

使い方の例については、“付録H サンプル集”の“H.50 CURRENT-DATE関数”を参照してください。

【書き方】

FUNCTION CURRENT-DATA

関数値

関数値は、この関数を実行したときの日付、時刻、およびグリニッジ標準時との時差を表す、21文字の英数字です。関数値の各文字位置には、下表に示す値が設定されます。

文字位置	値
1～ 4	西暦の年を表す4桁の数字。
5～ 6	月を表す2桁の数字。値の範囲は01～12。
7～ 8	日を表す2桁の数字。値の範囲は01～31。
9～10	時を表す2桁の数字。値の範囲は00～23。
11～12	分を表す2桁の数字。値の範囲は00～59。
13～14	秒を表す2桁の数字。値の範囲は00～59。
15～16	1/100 秒を表す2桁の数字。値の範囲は00～99。 (*1)
17	地方時間（文字位置1～16で表される時間）が、グリニッジ標準時から遅れているか進んでいるかを表す符号。地方時間がグリニッジ標準時よりも遅れている場合は“－”、地方時間がグリニッジ標準時よりも進んでいる場合は“＋”が設定されます。 (*2)
18～19	グリニッジ標準時からの時間単位の遅れまたは進みを表す2桁の数字。文字位置17が“－”の場合、値の範囲は00～12。文字位置17が“＋”の場合、値の範囲は00～13。 (*2)
20～21	グリニッジ標準時からの分単位の遅れまたは進みを表す2桁の数字。値の範囲は00～59。 (*2)

*1 ： 秒の小数部を得る機能がシステムにない場合、文字位置15～16には、00が設定されます。

*2 ： 地方時間のグリニッジ標準時との時差を得る機能がシステムにない場合、文字位置17～21には、00000 が設定されます。

関数の型

関数の型は、英数字です。

6.6.10 DATE-OF-INTEGER関数

引数-1(通日)を標準形式の日付に変換して返します。

使い方の例については、“付録H サンプル集”の“H.54 DATE-OF-INTEGER関数”を参照してください。

【書き方】

FUNCTION DATE-OF-INTEGER (引数-1)

引数

1. 引数-1の型は、整数でなければなりません。
2. 引数-1の値は、西暦1601年1月1日を第1日とする、通日を表す正の整数でなければなりません。

関数値

1. 関数値は、引数-1に指定した通日に等価な、標準形式の日付です。
2. 関数値は、yyyymmddの形式で設定されます。yyyyは西暦の年、mmは月、ddは日を表します。

関数の型

関数の型は、整数です。

6.6.11 DAY-OF-INTEGER関数

引数-1(通日)を年日形式の日付に変換して返します。

【書き方】

FUNCTION DAY-OF-INTEGER (引数-1)

引数

1. 引数-1の型は、整数でなければなりません。
2. 引数-1の値は、西暦1601年1月1日を第1日とする、通日を表す正の整数でなければなりません。

関数値

1. 関数値は、引数-1に指定した通日に等価な、年日形式の日付です。
2. 関数値は、yyyydddの形式で設定されます。yyyyは西暦の年、dddはその年の通日を表します。

関数の型

関数の型は、整数です。

6.6.12 DISPLAY-OF関数

引数-1に含まれる日本語文字を、対応する英数字文字に置き換えます。

DISPLAY-OF関数は、[Win32][WinIPF][Winx64][Solaris][LinuxIPF][Linux64][.NET]固有の機能です。

【書き方】

FUNCTION DISPLAY-OF (引数-1 [引数-2])

引数

1. 引数-1の字類は、日本語でなければなりません。
2. 引数-2の字類は、英字または英数字で、長さは1文字でなければなりません。
3. 引数-2は、対応する英数字がなかった場合の、英数字文字の代用文字を指定します。

関数値

1. 関数値は、引数-1に含まれる日本語文字を対応する英数字文字に置き換えた文字列です。

2. 引数-2を指定した場合、対応する英数字文字を持たない引数-1中の日本語文字を、代用文字に置き換えた文字列が返却されます。
3. 引数-2を省略し、かつ、引数-1中の日本語文字に対応する英数字文字がない場合は、実行時例外が発生します。
4. 関数値の大きさは、変換後の文字列を表示用の用途で保持するために必要な大きさです。
5. 引数-1が後置空白を含む文字列の場合、後置空白を除いた文字列が変換の対象になります。

関数値の型

関数値の型は英数字です。

DISPLAY-OF関数固有の規則

この関数は、動作モードがUnicode(ISO/IEC 10646-1)の場合だけ使用できます。

6.6.13 FACTORIAL関数

引数-1の階乗を返します。

【書き方】

FUNCTION FACTORIAL (引数-1)

引数

1. 引数-1の型は、整数でなければなりません。
2. 引数-1の値は、0以上の整数でなければなりません。

関数値

1. 引数-1の値が0の場合、関数値は1です。
2. 引数-1の値が正の数の場合、関数値は引数-1の階乗です。

関数の型

関数の型は、整数です。

6.6.14 INTEGER関数

引数-1以下の最大の整数を返します。

【書き方】

FUNCTION INTEGER (引数-1)

引数

1. 引数-1の型は、数字でなければなりません。
2. 引数に浮動小数点は指定できません。

関数値

関数値は、引数-1以下の最大の整数です。例えば、引数-1の値が＋1.5の場合、＋1です。引数-1の値が－1.5の場合、－2です。

関数の型

関数の型は、整数です。

6.6.15 INTEGER-OF-DATE関数

引数-1(標準形式の日付)を通日に変換して返します。

使い方の例については、“付録H サンプル集”の“H.53 INTEGER-OF-DATE関数”を参照してください。

【書き方】

FUNCTION INTEGER-OF-DATE (引数-1)

引数

1. 引数-1の型は、整数でなければなりません。
2. 引数-1の値は、yyyymmddの形式の8桁の整数でなければなりません。
3. 引数-1には、以下の式に従って、標準形式の日付を設定します。

```
(yyyy * 10000) + (mm * 100) + dd
```

yyyy、mmおよびddの規則は、以下のとおりです。

a. yyyyは、西暦の年を表す数です。yyyyの値は、1600より大きくなければなりません。

b. mmは、月を表す数です。mmの値の範囲は、1～12でなければなりません。

c. ddは、日を表す数です。ddの値の範囲は、1～31でなければなりません。

d. yyyy、mmおよびddを組み合わせた値は、正しい年月日を表す値でなければなりません。

関数値

関数値は、引数-1(標準形式の日付)に等価な、通日を表す整数です。通日は、西暦1601年1月1日を第1日とします。

関数の型

関数の型は、整数です。

6.6.16 INTEGER-OF-DAY関数

引数-1(年日形式の日付)を通日に変換して返します。

【書き方】

FUNCTION INTEGER-OF-DAY (引数-1)

引数

1. 引数-1の型は、整数でなければなりません。
2. 引数-1の値は、yyyydddの形式の7桁の整数でなければなりません。
3. 引数-1には、以下の式に従って、年日形式の日付を設定します。

```
(yyyy * 1000) + ddd
```

yyyyおよびdddの規則は、以下のとおりです。

a. yyyyは、西暦の年を表す数です。yyyyの値は、1600より大きくなければなりません。

b. dddは、その年の通日です。dddの値の範囲は、1～366でなければなりません。

c. yyyyとdddを組み合わせた値は、正しい年日を表す値でなければなりません。

関数値

関数値は、引数-1(年日形式の日付)に等価な、通日を表す整数です。通日は、西暦1601年1月1日を第1日とします。

関数の型

関数の型は、整数です。

6.6.17 INTEGER-PART関数

引数-1の整数部分を返します。

【書き方】

FUNCTION INTEGER-PART (引数-1)

引数

1. 引数-1の型は、数字でなければなりません。
2. 引数に浮動小数点は指定できません。

関数値

1. 引数-1の値が0の場合、関数値は0です。
2. 引数-1の値が正の数の場合、関数値は引数-1の値以下の最大の整数です。例えば、引数-1が＋1.5の場合、＋1です。
3. 引数-1の値が負の数の場合、関数値は引数-1の値以上の最小の整数です。例えば、引数-1が－1.5の場合、－1です。

関数の型

関数の型は、整数です。

6.6.18 LENG関数

引数-1の大きさ(バイト数またはビット数)を返します。

使い方の例については、“付録H サンプル集”の“H.48 LENG関数”を参照してください。

【書き方】

FUNCTION LENG (引数-1)

引数

引数-1は、データ項目、文字定数、16進文字定数または日本語定数でなければなりません。

関数値

関数値は、引数-1の大きさ(バイト数)です。ただし、内部ブールの場合はビット数です。

関数の型

関数の型は、整数です。

LENG関数に固有の規則

LENG関数は、以下の場所にだけ書くことができます。

- MOVE文の送出し側データ項目
- 算術式
- 手続き部で、整数の数字定数が書けるところ

6.6.19 LENGTH関数

引数-1の長さ(以下のどちらか)を返します。

- 文字位置の個数
- 日本語文字位置の個数

【書き方】

FUNCTION LENGTH (引数-1)

引数

1. 引数-1は、以下でなければなりません。
   - 文字定数
   - 16進文字定数
   - 日本語定数
   - 任意のデータ項目
2. 引数-1に従属するデータ項目にDEPENDING ON指定付きのOCCURS句を指定した場合、LENGTH関数が評価されるときのDEPENDING ON指定のデータ名の値を使って、関数値が決定されます。

関数値

1. 関数値は、下表のとおりです。

引数-1	関数値
文字定数	引数-1の文字位置の個数（バイト数）
16進文字定数	引数-1の文字位置の個数（バイト数）
日本語定数	引数-1の日本語文字位置の個数（日本語文字の個数）
英字項目、英数字項目、英数字編集項目または数字編集項目	引数-1の文字位置の個数（バイト数）
数字項目、集団項目、指標データ項目	引数-1が占める記憶領域のバイト数
ポインタデータ項目	引数-1が占める記憶領域のバイト数
日本語項目または日本語編集項目	引数-1の日本語文字位置の個数（日本語文字の個数）
外部ブール項目	引数-1のブール文字位置の個数（バイト数）
内部ブール項目	引数-1のビット数

2. 引数-1が可変反復データ項目を含む集団項目の場合、関数値は、可変反復データ項目のOCCURS句のDEPENDING ON指定のデータ名の値を評価して決定されます。この評価は、送出し側のデータ項目にOCCURS句を指定したときの規則に従います。
3. 暗黙のFILLER項目が存在する場合、関数値はその文字数を含みます。

関数の型

関数の型は、整数です。

6.6.20 LOG関数

底をeとする引数-1の対数を返します。

【書き方】

FUNCTION LOG (引数-1)

引数

1. 引数-1の型は、数字でなければなりません。
2. 引数-1の値は、正でなければなりません。

関数値

関数値は、底をeとする引数-1の対数(自然対数)の近似値です。

関数の型

関数の型は、数字です。

6.6.21 LOG10関数

底を10とする引数-1の対数を返します。

【書き方】

FUNCTION LOG10 (引数-1)

引数

1. 引数-1の型は、数字でなければなりません。
2. 引数-1の値は、正でなければなりません。

関数値

関数値は、底を10とする引数-1の対数(常用対数)の近似値です。

関数の型と精度

関数の型は、数字です。

6.6.22 LOWER-CASE関数

引数-1の中の英大文字と等価な英小文字を返します。

使い方の例については、“付録H サンプル集”の“H.55 LOWER-CASE関数”を参照してください。

【書き方】

FUNCTION LOWER-CASE (引数-1)

引数

1. 引数-1の型は、英字または英数字でなければなりません。
2. 引数-1の長さは、1文字以上でなければなりません。

関数値

1. 関数値は、引数-1の中の英大文字を英小文字に置き換えた文字列です。すなわち、引数-1の中の英大文字が置き換えられることを除いて、引数-1と同じ文字列です。
2. 関数値の長さは、引数-1の長さと同じです。

関数の型

関数の型は、英数字です。

6.6.23 MAX関数

引数の最大値を返します。

使い方の例については、“付録H サンプル集”の“H.57 MAX関数”を参照してください。

【書き方】

FUNCTION MAX ({引数-1} …)

引数

1. 引数-1の型は、任意です。
2. 引数-1の組み合わせは、“関数の型”で示す表のいずれかでなければなりません。

関数値

1. 関数値は、引数-1の並びの中で、最大値を持つ引数-1の内容です。
2. 2つ以上の引数-1が同じ最大値を持つ場合、それらの中で最も左側に書いたものが関数値になります。
3. 最大値を求めるための比較の方法は、比較条件の規則に従います。
4. 関数の型が英数字の場合、関数値の長さは、最大値を持つ引数-1の長さと同じです。

関数の型

関数の型を下表に示します。

引数-1の型	関数の型
すべて英字 または すべて英数字	英数字
すべて整数	整数
すべて数字 または 数字と整数の混在	数字

6.6.24 MEAN関数

引数の算術平均値を返します。

【書き方】

FUNCTION MEAN ({引数-1} …)

引数

引数-1の型は、数字でなければなりません。

関数値

関数値は、引数-1の並びの算術平均値です。すなわち、引数-1の並びの合計を、引数-1の個数で割ったものです。

関数の型

関数の型は、数字です。

6.6.25 MEDIAN関数

引数の中央値を返します。

【書き方】

FUNCTION MEDIAN ({引数-1} …)

引数

引数-1の型は、数字でなければなりません。

関数値

1. 関数値は、引数-1の並びを数値の大小順序によって並べ替えたときの中央値です。中央値とは、以下の値です。
   - 引数-1の個数が奇数の場合、並べ替えの結果、中央に位置する値です。
   - 引数-1の個数が偶数の場合、並べ替えの結果、中央に位置する2つの値の算術平均値です。
2. 引数を並べ替えるための比較の方法は、比較条件の規則に従います。

関数の型

関数の型は、数字です。

6.6.26 MIDRANGE関数

引数の最小値と最大値の算術平均値を返します。

【書き方】

FUNCTION MIDRANGE ({引数-1} …)

引数

引数-1の型は、数字でなければなりません。

関数値

関数値は、引数-1の並びの中の最大値と最小値の算術平均値です。すなわち、関数値は、引数-1の並びの中の最大値と最小値の合計を、2で割ったものです。最大値と最小値を求めるための比較の方法は、比較条件の規則に従います。

関数の型

関数の型は、数字です。

6.6.27 MIN関数

引数の最小値を返します。

使い方の例については、“付録H サンプル集”の“H.58 MIN関数”を参照してください。

【書き方】

FUNCTION MIN ({引数-1} …)

引数

1. 引数-1の型は、任意です。
2. 引数-1の組み合わせは、“関数の型”で示す表のいずれかでなければなりません。

関数値

1. 関数値は、引数-1の並びの中で、最小値を持つ引数-1の内容です。
2. 2つ以上の引数-1が同じ最小値を持つ場合、それらの中で最も左側に書いたものが関数値になります。
3. 最小値を求めるための比較の方法は、比較条件の規則に従います。
4. 関数の型が英数字の場合、関数値の長さは、最小値を持つ引数-1の長さと同じです。

関数の型

関数の型を下表に示します。

引数-1の型	関数の型
すべて英字 または すべて英数字	英数字
すべて整数	整数
すべて数字 または 数字と整数の混在	数字

6.6.28 MOD関数

引数-2を法とする引数-1の整数値を返します。

【書き方】

FUNCTION MOD (引数-1 引数-2)

引数

1. 引数-1および引数-2の型は、整数でなければなりません。
2. 引数-2の値は、ゼロ以外の整数です。

関数値

1. 関数値は、引数-2を法とする引数-1の整数値です。関数値は、以下の式に従って求められます。

```
"引数-1" - ("引数-2" * FUNCTION INTEGER ("引数-1" / "引数-2"))
```

2. 関数値の例を、下表に示します。

引数-1	引数-2	関数値
11	5	1
-11	5	4
11	-5	-4
-11	-5	-1

関数の型

関数の型は、整数です。

6.6.29 NATIONAL関数

引数-1の日本語文字を除くCOBOL文字集合を、日本語文字に置き換えます。

使い方の例については、“付録H サンプル集”の“H.49 NATIONAL関数”を参照してください。

【書き方】

FUNCTION NATIONAL (引数-1)

引数

引数-1は、長さが少なくとも1文字であって、かつその字類は、英字、英数字、数字でなければなりません。ただし数字は、外部10進データ項目だけを許します。

関数値

1. 関数値は、日本語文字を除くCOBOL文字集合が、対応する日本語文字に置き換えられることを除いて、引数-1と同じ文字列です。
2. 関数値の文字数は引数-1の文字数と同じになります。
3. 引数-1に符号付き外部10進データ項目が指定された場合、符号は無視されます。
4. 引数-1に文字として不当なコードが設定されていた場合、結果は保証されません。

関数の型

関数の型は日本語です。

6.6.30 NATIONAL-OF関数

引数-1に含まれる英数字文字を、対応する日本語文字に置き換えます。

NATIONAL-OF関数は、[Win32][WinIPF][Winx64][Solaris][LinuxIPF][Linux64][.NET]固有の機能です。

【書き方】

FUNCTION NATIONAL-OF (引数-1 引数-2)

引数

1. 引数-1の字類は、英字または英数字でなければなりません。
2. 引数-2の字類は、日本語で、長さは1文字でなければなりません。
3. 引数-2は、対応する日本語文字がなかった場合の、日本語文字の代用文字を指定します。

関数値

1. 関数値は、引数-1に含まれる英数字文字を対応する日本語文字に置き換えた文字列です。
2. 引数-2を指定した場合、対応する日本語文字を持たない引数-1中の英数字文字を、代用文字に置き換えた文字列が返却されます。
3. 引数-2を省略し、かつ、引数-1中の英数字文字に対応する日本語文字がない場合は、実行時例外が発生します。
4. 関数値の大きさは、変換後の日本語文字列を保持するために必要な大きさです。
5. 引数-1が後置空白を含む文字列の場合、後置空白を除いた文字列が変換の対象になります。

関数値の型

関数値の型は日本語です。

NATIONAL-OF関数固有の規則

この関数は、動作モードがUnicode(ISO/IEC 10646-1)の場合だけ使用できます。

6.6.31 NUMVAL関数

引数-1(数字定数の形式の文字列)を数値に変換して返します。

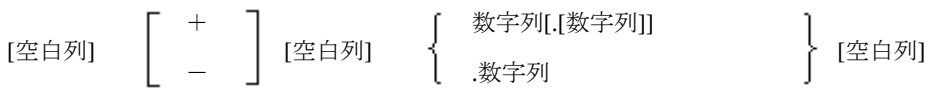
【書き方】

FUNCTION NUMVAL (引数-1)

引数

1. 引数-1は、文字定数または英数字項目でなければなりません。引数-1の内容は、以下のいずれかの形式で表現しなければなりません。以下で、空白列は1つ以上の空白の並びを表し、数字列は1つ以上の数字の並びを表します。

   (a)

   (b)

2. 引数-1の数字の桁数の合計、すなわち数字列の数字の個数の合計は、18以下でなければなりません。
3. 特殊名段落にDECIMAL-POINT IS COMMA句を書いた場合、引数-1の中の小数点の機能とコンマの機能が入れ替わります。このとき、小数点を表す終止符“.”の代わりにコンマを使わなければなりません。

関数値

関数値は、引数-1が表す数値です。

関数の型

関数の型は、数字です。

6.6.32 NUMVAL-C関数

引数-2(通貨記号)またはコンマを含む引数-1の文字列を数値に変換して返します。

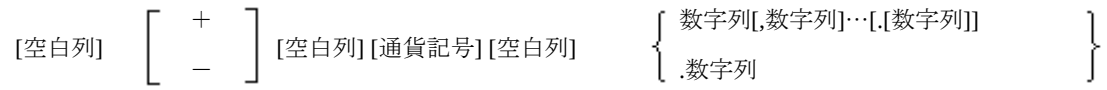
【書き方】

FUNCTION NUMVAL-C (引数-1 [引数-2])

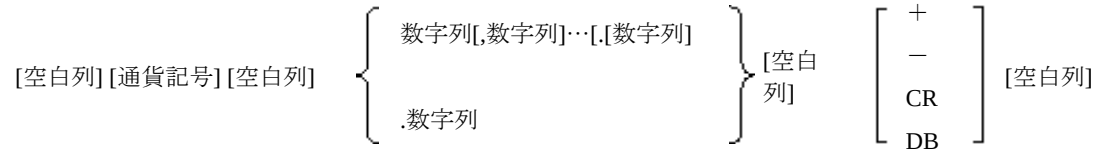
引数

1. 引数-1は、文字定数または英数字項目でなければなりません。引数-1の内容は、以下のいずれかの形式で表現しなければなりません。以下で、空白列は1つ以上の空白の並びを表し、数字列は1つ以上の数字の並びを表し、通貨記号は引数-2の文字列と同じ文字列(1文字以上)を表します。

   (a)

   (b)

2. 引数-1の数字の桁数の合計、すなわち数字列の数字の個数の合計は、18以下でなければなりません。
3. 特殊名段落にDECIMAL-POINT IS COMMA句を書いた場合、引数-1の中の小数点の機能とコンマの機能が入れ替わります。このとき、小数点を表す終止符“.”の代わりにコンマを使用し、コンマ“,”の代わりに終止符を使わなければなりません。
4. 引数-2は、文字定数または英数字項目でなければなりません。
5. 引数-2には、通貨記号を設定します。引数-2を省略した場合、引数-2に、特殊名段落のCURRENCY SIGN句で指定した通貨記号を書いたものとみなされます。CURRENCY SIGN句を省略した場合は、COBOL文字集合の通貨記号が使われます。

関数値

関数値は、引数-1が表す数値です。

関数の型

関数の型は、数字です。

6.6.33 ORD関数

文字の大小順序における、引数-1の順序位置を返します。

【書き方】

FUNCTION ORD (引数-1)

引数

1. 引数-1の型は、英字または英数字でなければなりません。
2. 引数-1の長さは、1文字でなければなりません。

関数値

関数値は、文字の大小順序における、引数-1の順序位置です。文字の大小順序とは、特殊名段落のALPHABET句で定義したものです。ALPHABET句を省略した場合は、計算機文字集合における文字の大小順序に従って、関数値が求められます。

関数値の型

関数の型は、整数です。

6.6.34 ORD-MAX関数

最大値を持つ引数の位置を返します。

【書き方】

FUNCTION ORD-MAX ({引数-1} …)

引数

1. 引数-1の型は、任意です。
2. 引数-1の型の組み合わせは、以下のいずれかでなければなりません。
   - すべて英字
   - すべて英数字
   - すべて整数
   - すべて数字
   - 数字と整数の混在

関数値

1. 関数値は、引数-1の並びの中で、最大値を持つ引数-1の位置です。例えば、左から2番目に指定した引数-1の値が、引数-1の並びの中の最大値である場合、関数値には2が設定されます。
2. 2つ以上の引数-1が同じ最大値を持つ場合、それらの中で最も左側に書いたものが最大値になります。したがって、関数値は、その位置を示す値になります。
3. 最大値を求めるための比較の方法は、比較条件の規則に従います。

関数の型

関数の型は、整数です。

6.6.35 ORD-MIN関数

最小値を持つ引数の位置を返します。

【書き方】

FUNCTION ORD-MIN ({引数-1} …)

引数

1. 引数-1の型は、任意です。
2. 引数-1の型の組み合わせは、以下のいずれかでなければなりません。
   - すべて英字
   - すべて英数字
   - すべて整数
   - すべて数字
   - 数字と整数の混在

関数値

1. 関数値は、引数-1の並びの中で、最小値を持つ引数-1の位置です。例えば、左から2番目に指定した引数-1の値が、引数-1の並びの中の最小値である場合、関数値には2が設定されます。
2. 2つ以上の引数-1が同じ最小値を持つ場合、それらの中で最も左側に書いたものが最小値になります。したがって、関数値は、その位置を示す値になります。
3. 最小値を求めるための比較の方法は、比較条件の規則に従います。

関数の型

関数の型は、整数です。

6.6.36 PRESENT-VALUE関数

引数-1を減価率として、各期末の現在価値を引数-2の並びに設定して返します。

【書き方】

FUNCTION PRESENT-VALUE (引数-1 {引数-2} …)

引数

1. 引数-1および引数-2の型は、数字でなければなりません。
2. 引数-1は、－1より大きい値でなければなりません。

関数値

関数値は、次の項からなる数列の和の近似値です。引数-2の各値に1つの項が対応します。指数nは、各項ごとに1から始めて1ずつ加算されます。

```
"引数-2" / ( 1 + "引数-1" ) ** n
```

関数の型

関数の型は、数字です。

6.6.37 RANDOM関数

一様分布からの擬似乱数の数値を返します。

【書き方】

FUNCTION RANDOM [(引数-1)]

引数

1. 引数-1の型は、整数でなければなりません。
2. 引数-1の値は、ゼロまたは正の整数でなければなりません。この値は、擬似乱数列を生成するための種子となる値として用いられます。
3. 引数-1を指定したRANDOM関数を実行すると、引数-1を種子とする擬似乱数列が生成されます。
4. 実行単位で最初に実行するRANDOM関数で引数-1を省略した場合、0を種子とする擬似乱数列が生成されます。
5. RANDOM関数を実行した後、同じ擬似乱数列から乱数を求める場合、引数-1を省略することができます。引数-1の値は、次に引数-1を書いたRANDOM関数を実行するまで有効です。

関数値

1. 関数値は、現在の擬似乱数列の中の任意の値です。
2. 関数値の範囲は、以下のとおりです。

```
0 ≦ 関数値 ＜ 1
```

3. 引数-1の値(種子となる値)が同じ場合、同じ擬似乱数列が使われます。

関数の型

関数の型は、数字です。

6.6.38 RANGE関数

引数の最大値から最小値を引いた値を返します。

【書き方】

FUNCTION RANGE ({引数-1} …)

引数

引数-1の型は、数字または整数でなければなりません。

関数値

1. 関数値は、引数-1の並びの中の最大値から最小値を引いた値です。
2. 最大値と最小値を求めるための比較の方法は、比較条件の規則に従います。

関数の型

関数の型を下表に示します。

引数-1の型	関数の型
すべて整数	整数

引数-1の型	関数の型
すべての数字 または 数字と整数の混在	数字

6.6.39 REM関数

引数-1を引数-2で割ったときの余りを返します。

使い方の例については、“付録H サンプル集”の“H.52 REM関数”を参照してください。

【書き方】

FUNCTION REM (引数-1 引数-2)

引数

1. 引数-1および引数-2の型は、数字でなければなりません。
2. 引数-2の値は、ゼロ以外の整数です。
3. 引数に浮動小数点は指定できません。

関数値

関数値は、引数-1を引数-2で割ったときの余りです。関数値は、以下の式で求められます。

```
"引数-1" - ("引数-2" * FUNCTION INTEGER-PART("引数-1" / "引数-2"))
```

関数の型

関数の型は、数字です。

6.6.40 REVERSE関数

引数-1の文字列を逆順にした文字列を返します。

使い方の例については、“付録H サンプル集”の“H.59 REVERSE関数”を参照してください。

【書き方】

FUNCTION REVERSE (引数-1)

引数

1. 引数-1の型は、英字または英数字でなければなりません。
2. 引数-1の長さは、1文字以上でなければなりません。

関数値

1. 関数値は、引数-1の文字列を逆順にした文字列です。関数値の長さは、引数-1の長さと同じです。
2. 引数-1の文字列の長さをnとすると、引数-1のj(1≦j≦n)番目の文字が、関数値のn－j＋1番目の文字として設定されます。

関数の型

関数の型は、英数字です。

6.6.41 SIN関数

引数-1の正弦を返します。

使い方の例については、“付録H サンプル集”の“H.44 SIN関数”を参照してください。

【書き方】

FUNCTION SIN (引数-1)

引数

1. 引数-1の型は、数字でなければなりません。
2. 引数-1の値は、単位をラジアンとして指定します。

関数値

1. 関数値は、引数-1の正弦の近似値です。
2. 関数値の範囲は、以下のとおりです。

```
-1 ≦ 関数値 ≦ +1
```

関数の型

関数の型は、数字です。

6.6.42 SQRT関数

引数-1の平方根を返します。

【書き方】

FUNCTION SQRT (引数-1)

引数

1. 引数-1の型は、数字でなければなりません。
2. 引数-1の値は、ゼロ以上でなければなりません。

関数値

関数値は、引数-1の平方根の近似値の絶対値です。

関数の型

関数の型は、数字です。

6.6.43 STANDARD-DEVIATION関数

引数の標準偏差の近似値を返します。

【書き方】

FUNCTION STANDARD-DEVIATION ({引数-1} …)

引数

引数-1の型は、数字でなければなりません。

関数値

1. 関数値は、引数-1の並びの標準偏差の近似値です。
2. 関数値は、以下の手順で求められます。
    a. 引数-1の平均とそれぞれの引数-1の差を求め、2乗します。
    b. a.で求めた値を、すべて加算します。この値を引数-1の個数で割ります。
    c. b.で求めた商の平方根を求めます。関数値は、この値の絶対値です。
3. 引数-1の並びがすべて同じ値の場合、関数値はゼロです。

関数の型

関数の型は、数字です。

6.6.44 STORED-CHAR-LENGTH関数

引数-1に含まれる有効文字(末尾の空白を除いた部分)の長さを返します。

使い方の例については、“付録H サンプル集”の“H.60 STORED-CHAR-LENGTH関数”を参照してください。

STORED-CHAR-LENGTH関数は、[Win32][WinIPF][Winx64][Solaris][Linux][LinuxIPF][Linux64][.NET]固有の機能です。

【書き方】

FUNCTION STORED-CHAR-LENGTH (引数-1)

引数

1. 引数-1は、英数字または日本語でなければなりません。ただし、集団項目であってはなりません。
2. 引数-1の長さは、1文字以上でなければなりません。

関数値

1. 関数値は、引数-1に含まれるデータの有効文字の文字位置の個数を返します。
2. 引数-1の字類が英数字の場合、関数値は有効文字の英数字文字位置の個数を表します。
3. 引数-1の字類が日本語の場合、関数値は有効文字の日本語文字位置の個数を表します。

4. 引数-1に格納されている文字は、引数-1の字類によって決定される表現形式に従うものでなければなりません。それ以外の値が含まれる場合、結果は保証されません。

関数の型

関数の型は整数です。

6.6.45 SUM関数

引数の合計を返します。

使い方の例については、“付録H サンプル集”の“H.51 SUM関数”を参照してください。

【書き方】

FUNCTION SUM ({引数-1} …)

引数

引数-1の型は、数字または整数でなければなりません。

関数値

関数値は、引数-1の並びの合計です。

関数の型

関数の型を下表に示します。

引数-1の型	関数の型
すべて整数	整数
すべての数字 または 数字と整数の混在	数字

6.6.46 TAN関数

引数-1の正接を返します。

使い方の例については、“付録H サンプル集”の“H.46 TAN関数”を参照してください。

【書き方】

FUNCTION TAN (引数-1)

引数

1. 引数-1の型は、数字でなければなりません。
2. 引数-1の値は、単位をラジアンとして設定します。

関数値

関数値は、引数-1の正接の近似値です。

関数の型

関数の型は、数字です。

6.6.47 UCS2-OF関数

引数-1に含まれる文字の表現形式をUCS-2に変換します。

使い方の例については、“付録H サンプル集”の“H.61 UCS2-OF関数”を参照してください。

UCS2-OF関数は、[Win32][WinIPF][Winx64][Solaris][Linux][LinuxIPF][Linux64][.NET]固有の機能です。

【書き方】

FUNCTION UCS2-OF (引数-1)

引数

1. 引数-1は、英字または英数字でなければなりません。ただし、集団項目であってはなりません。
2. 引数-1の長さは、1文字以上でなければなりません。

関数値

1. 関数値は、引数-1に含まれる文字を対応するUCS-2で表現される文字列に置き換えたものを返します。
2. 関数値の大きさ(バイト数)は、変換後の文字列の大きさに従います。
3. 引数-1に含まれる文字は、英数字項目に格納可能な文字でなければなりません。それ以外の値が含まれる場合、結果は保証されません。

関数の型

関数の型は日本語です。

UCS2-OF関数固有の規則

この関数は、動作モードがUnicode(ISO/IEC 10646-1)の場合だけ使用できます。

6.6.48 UPPER-CASE関数

引数-1の中の英小文字と等価な英大文字を返します。

使い方の例については、“付録H サンプル集”の“H.56 UPPER-CASE関数”を参照してください。

【書き方】

FUNCTION UPPER-CASE (引数-1)

引数

1. 引数-1の型は、英字または英数字でなければなりません。
2. 引数-1の長さは、1文字以上でなければなりません。

関数値

1. 関数値は、引数-1の中の英小文字を英大文字に置き換えた文字列です。すなわち、引数-1の中の英小文字が置き換えられることを除いて、引数-1と同じ文字列です。
2. 関数値の長さは、引数-1の長さと同じです。

関数の型

関数の型は、英数字です。

6.6.49 UTF8-OF関数

引数-1に含まれる文字の表現形式をUTF-8に変換します。

使い方の例については、“付録H サンプル集”の“H.62 UTF8-OF関数”を参照してください。

UTF8-OF関数は、[Win32][WinIPF][Winx64][Solaris][Linux][LinuxIPF][Linux64][.NET]固有の機能です。

【書き方】

FUNCTION UTF8-OF (引数-1)

引数

1. 引数-1は、日本語でなければなりません。
2. 引数-1の長さは、1文字以上でなければなりません。

関数値

1. 関数値は、引数-1に含まれる文字を対応するUTF-8で表現される文字列に置き換えたものを返します。
2. 関数値の大きさ(バイト数)は、変換後の文字列の大きさに従います。
3. 引数-1に含まれる文字は、日本語項目に格納可能な文字でなければなりません。それ以外の値が含まれる場合、結果は保証されません。

関数の型

関数の型は英数字です。

UTF8-OF関数固有の規則

この関数は、動作モードがUnicode(ISO/IEC 10646-1)の場合だけ使用できます。

6.6.50 VARIANCE関数

引数の分散の近似値を返します。

【書き方】

FUNCTION VARIANCE ({引数-1} …)

引数

引数-1の型は、数字でなければなりません。

関数値

1. 関数値は、引数-1の並びの分散の近似値です。すなわち、引数-1の並びの標準偏差の2乗です。標準偏差については、“6.6.43 STANDARD-DEVIATION関数”を参照してください。
2. 引数-1の並びがすべて同じ値の場合、関数値はゼロです。

関数の型

関数の型は、数字です。

6.6.51 WHEN-COMPILED関数

プログラムを翻訳したときの日付と時刻を返します。

使い方の例については、“付録H サンプル集”の“H.63 WHEN-COMPILED関数”を参照してください。

【書き方】

FUNCTION WHEN-COMPILED

関数値

1. 関数値は、プログラムを翻訳したときの日付、時刻、およびグリニッジ標準時との時差を表す、21文字の英数字です。関数値の各文字位置には、下表に示す値が設定されます。

文字位置	値
1～ 4	西暦の年を表す4桁の数字。
5～ 6	月を表す2桁の数字。値の範囲は01～12。
7～ 8	日を表す2桁の数字。値の範囲は01～31。
9～10	時を表す2桁の数字。値の範囲は00～23。
11～12	分を表す2桁の数字。値の範囲は00～59。
13～14	秒を表す2桁の数字。値の範囲は00～59。
15～16	1/100 秒を表す2桁の数字。値の範囲は00～99。(*1)
17	地方時間（文字位置1～16で表される時間）が、グリニッジ標準時から遅れているか進んでいるかを表す符号。地方時間がグリニッジ標準時よりも遅れている場合は“－”、地方時間がグリニッジ標準時よりも進んでいる場合は“＋”が設定されます。(*2)
18～19	グリニッジ標準時からの時間単位の遅れまたは進みを表す2桁の数字。文字位置17が“－”の場合、値の範囲は00～12。文字位置17が“＋”の場合、値の範囲は00～13。(*2)
20～21	グリニッジ標準時からの分単位の遅れまたは進みを表す2桁の数字。値の範囲は00～59。(*2)

*1 ： 秒の小数部を得る機能が、プログラムを翻訳したシステムにない場合、文字位置15 ～16には00が設定されます。

*2 ： 地方時間のグリニッジ標準時との時差を得る機能が、プログラムを翻訳したシステムにない場合、文字位置17～21には、00000 が設定されます。

2. 関数値は、この関数を書いた原始プログラムを翻訳した日時です。この関数を内部プログラムに書いた場合、関数値は、一番外側のプログラムが翻訳された日時になります。

関数の型

関数の型は、英数字です。

第7章 原始文操作

原始文操作機能は、原始プログラムの一部をCOBOL登録集から複写したり、原始プログラムの一部を置き換えたりする機能です。原始文操作機能には、COPY文およびREPLACE文の2つの文があります。

COPY文は、登録集原文を、COPY文を書いたプログラムに複写します。複写するときに、登録集原文の一部を置き換えることもできます。REPLACE文は、原始プログラム原文の一部を置き換えます。

COPY文およびREPLACE文は、原始プログラムの任意の場所に書くことができます。また、COPY文だけで原始プログラムを構成することができます。これらの文は、翻訳時、他の文が翻訳される前に処理されます。実行時は意味を持ちません。

原文

原始プログラムおよびCOBOL登録集の中の、行または行の集まりを「原文」といいます。原始プログラムの中の原文を「原始プログラム原文」といい、COBOL登録集の中の原文を「登録集原文」といいます。

原文語

COBOL登録集、原始プログラムおよび仮原文の中のA領域とB領域にある文字列のうち、以下の文字列を「原文語」といいます。

- 空白、仮原文区切り記号および分離符の引用符を除く分離符。分離符の左括弧と右括弧は、その記述位置に関係なく、つねに原文語とみなされます。
- 定数は、1つの原文語です。
  以下の場合、定数値の開始の分離符から終了の分離符までの一連の文字列が、1つの原文語です。
  - 文字定数
  - 日本語定数
  - ブール定数
  - 16進文字定数
- 分離符と分離符の間にある、分離符でも定数でもない文字列は、1つの原文語です。ただし、注記行にある文字列および語COPYは、原文語ではありません。

仮原文

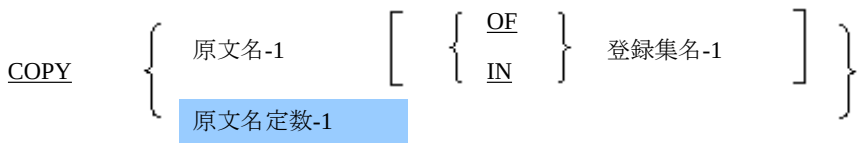
原始プログラムおよびCOBOL登録集の中の、2つの仮原文区切り記号の間にある原文語、注記行および分離符の空白を「仮原文」といいます。「仮原文区切り記号」とは、2個の連続した等号(＝＝)のことです。仮原文に、仮原文区切り記号は含まれません。

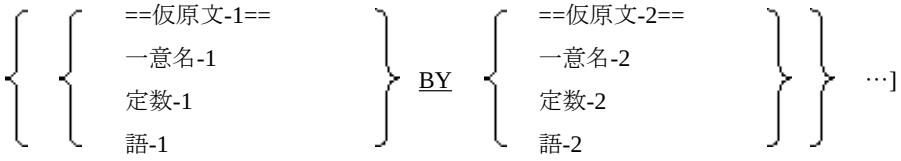
7.1 COPY文

登録集原文を原始プログラムに複写します。

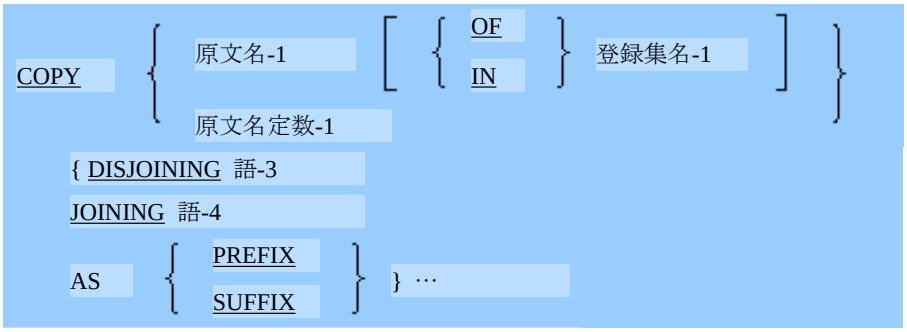
使い方の例については、“付録H サンプル集”の“H.64 COPY文”を参照してください。

【書き方1】登録集原文をそのまま、または登録集原文の一部を置き換えて複写する

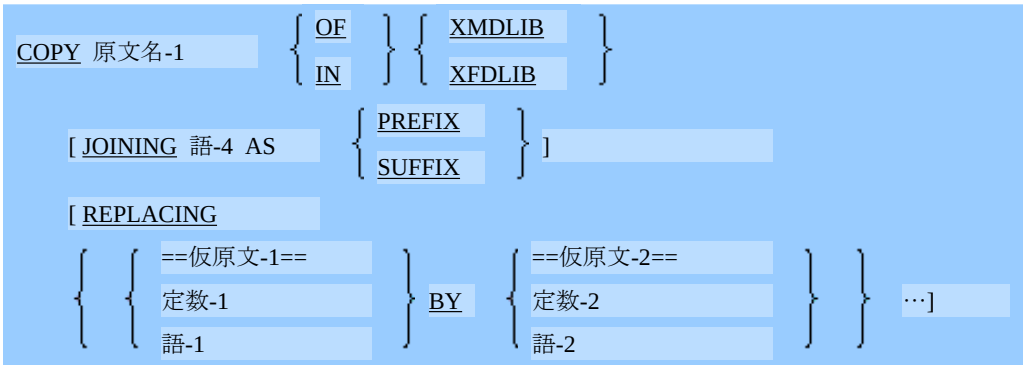




【書き方2】登録集原文中のプレフィックスまたはサフィックスを置き換えて複写する



【書き方3】登録集原文にプレフィックスまたはサフィックスを付けて複写する



構文規則

1. 1つの翻訳単位の中で2つ以上のCOBOL登録集を使う場合、原文名-1は、その原文が存在するCOBOL登録集の登録集名で修飾しなければなりません。COBOL登録集の指定の方法については、“NetCOBOL使用手引書”を参照してください。
2. 原文名-1は、1つのCOBOL登録集の中で一意でなければなりません。
3. 以下の記述規則については、“1.2 言語の基本要素”を参照してください。
   - 原文名-1
   - 原文名定数-1
4. COPY文は、空白の後から書き始め、分離符の終止符で止めなければなりません。
5. 仮原文-1および仮原文-2は、正書法に従って書かなければなりません。
6. 仮原文-1には、1つ以上の原文語を書かなければなりません。
7. 仮原文-2には、1つ以上の原文語を書きます。原文語は、省略することもできます。
8. 仮原文-1および仮原文-2は、正書法に従って、2行以上にわたって記述することができますが、一意名-1および一意名-2は2行以上にわたって記述することはできません。

9. 以下は、COBOLの語の規則に従った文字列でなければなりません。

   ― 語-1

   ― 語-2

   ― 語-3

   ― 語-4

   ただし、語COPYを指定してはいけません。

   語-3および語-4を日本語文字で構成する場合、日本語文字のJIS非漢字の英大文字、英小文字および数字からなる文字列を書くこともできます。

10. COPY文は、右側の引用符以外の分離符または文字列が書けるところならば、どこにでも書くことができます。ただし、COPY文の中にCOPY文を書くことはできません。

11. 仮原文-1、仮原文-2および登録集原文の中の1つの原文語は、1～324文字でなければなりません。

12. 仮原文-1に、分離符のコンマまたは分離符のセミコロンだけを書くことはできません。

13. 語COPYを注記項、注記行または文字定数の中に書いた場合、語COPYは、それらを構成する文字列の一部とみなされます。

14. 登録集原文中の利用者語が英数字の場合、語-4は英数字でなければなりません。登録集原文中の利用者語が日本語文字の場合、語-4は日本語文字でなければなりません。

15. 語-4の長さの制限については、“付録B システムの定量制限”を参照してください。

16. 書き方3で、JOINING指定とREPLACING指定を同時に指定してはなりません。

一般規則

書き方1～書き方3に共通する規則

1. COPY文は、翻訳時、プログラムの中に登録集原文を複写します。登録集原文は、以下で指定します。

   ― 原文名-1

   ― 原文名定数-1

   以降の説明で、「登録集原文」は、原文名-1または原文名定数-1に指定した登録集原文を表します。

   登録集原文を複写する規則は、以下のとおりです。

   ― 書き方1でREPLACING指定を省略した場合

   登録集原文をそのまま複写します。

   ― 書き方1でREPLACING指定を書いた場合

   登録集原文中の原文語からBYの左辺と一致する文字列を検索し、一致した文字列をBYの右辺に置き換えて複写します。一致しなかった文字列は、そのまま複写します。

   ― 書き方2の場合

   PREFIX指定を書いた場合、登録集原文中の原文語のプレフィックス(原文語の左端からハイフンが現れるまでの文字列)が、語-3と一致するものを検索し、一致した文字列を語-4と置き換えて複写します。
   SUFFIX指定を書いた場合、登録集原文中の原文語のサフィックス(原文語の右端からハイフンが現れるまでの文字列)が語-3と一致するものを検索し、一致した文字列を語-4に置き換えて複写します。

   ― 書き方3の場合

   登録集原文中の、レベル番号の直後またはREDEFINES句に書いたすべてのデータ名を、置き換えの対象にします。PREFIX指定を書いた場合、置き換えの対象となったデータ名の左端文字の前にプレフィックス(語-4とハイフンをこの順に結合した文字列)を挿入して複写します。SUFFIX指定を書いた場合、置き換えの対象となったデータ名の右端文字の後にサフィックス(ハイフンと語-4をこの順に結合した文字列)を挿入して複写します。置き換えの対象にならなかった文字列は、そのまま複写します。

2. COPY文を書いた原始プログラムの翻訳は、すべてのCOPY文を処理した後に、原始プログラムの翻訳を行うことと同じです。翻訳時、COPYで始まり終止符で終わるCOPY文全体が、登録集原文または登録集原文の一部を編集したものに置き換えられます。COPY文は、実行時には意味を持ちません。
3. 登録集原文は、COBOLの正書法の規則に従わなければなりません。
4. 登録集原文中に、COPY文を書くことができます。ただし、以下の記述を処理した結果、新しいCOPY文が生成されるようなCOPY文を書くことはできません。

   ― REPLACING指定

   ― JOINING指定

書き方1の規則

1. REPLACING指定による原文の置き換えは、以下の順序で行われます。
   a. 置き換えのための比較は、登録集原文中の、分離符のコンマおよび分離符のセミコロン以外の最初の原文語から開始します。この原文語を「現在の原文語」とします。この原文語の前にある空白、分離符のコンマおよび分離符のセミコロンは、そのまま複写します。
   b. BYの左辺を、「現在の原文語」から始まる登録集原文と1文字ずつ比較します。BY指定を2つ以上書いた場合、それらを書いた順に、BY指定のすべての作用対象を比較します。比較の方法は、以下のとおりです。
      - 一意名-1、語-1または定数-1を書いた場合、その内容を「現在の原文語」と比較します。
      - 仮原文-1を書いた場合、仮原文-1に書いた文字列全体を「現在の原文語」から始まる登録集原文と比較します。このとき、仮原文-1および登録集原文の中の、1つ以上の分離符のコンマ、1つ以上の分離符のセミコロン、および1つ以上の空白は、1つの空白とみなします。
   c. b.の比較を、比較の結果が一致するまで、またはBYの左辺がなくなるまで繰り返します。
   d. b.の比較で、どの比較の結果も一致しなかった場合、「現在の原文語」をそのまま複写します。そして、「現在の原文語」の次の原文語を「現在の原文語」にして、b.の処理に戻ります。
   e. b.の比較で、比較の結果が一致した場合、以下の処理を行います。
      - 一意名-1、語-1または定数-1を書いた場合、「現在の原文語」をBYの右辺に置き換えて複写します。そして、「現在の原文語」の次の登録集原文語を「現在の原文語」にして、b.の処理に戻ります。
      - 仮原文-1を書いた場合、仮原文-1と一致した部分を、BYの右辺に置き換えて複写します。そして、仮原文-1と一致した部分の最後の原文語の次の原文語を「現在の原文語」として、b.の処理に戻ります。
   f. b.～e.の処理を、登録集原文中の最後の原文語を比較するまで繰り返します。
2. 登録集原文および仮原文-1の中の注記行および空白行は、置き換えのための比較で無視されます。
3. 登録集原文、仮原文-1および仮原文-2の中に、デバッグ行を書くことができます。比較を行うとき、登録集原文および仮原文-1の中の標識領域の“D”は無視されます。
4. 登録集原文中の置き換えの対象にならない原文語は、登録集原文上の領域と同じ領域に位置するように複写されます。すなわち、登録集原文のA領域から始まる原文語はA領域から始まるように複写され、B領域から始まる原文語はB領域から始まるように複写されます。ただし、A領域に2つ以上の原文語が存在し、そのうちのいずれかの原文語がその長さよりも長い原文語に置き換えられる場合、A領域から始まるように複写できない原文語は、B領域から始まるように複写されます。
5. 登録集原文中の置き換えの対象となった原文語は、新しい原文語(一意名-2、定数-2または語-2または仮原文-2の中のそれぞれの原文語)に置き換えられて、正書法に従って複写されます。
6. 登録集原文中の原文語が、一意名-2、定数-2または語-2で置き換えられるとき、新しい原文語(一意名-2、定数-2または語-2)は、置き換えの対象となった原文語の登録集原文上の領域と同じ領域から始まるように複写されます。
7. 登録集中の原文語が、仮原文-2の中のそれぞれの原文語で置き換えられるとき、仮原文-2のそれぞれの原文語は、仮原文-2上の領域と同じ領域から始まるように複写されます。仮原文-2の中の原文語間の空白は、そのまま複写されます。
8. 以下の原文語は、デバッグ行上に存在するように複写されます。
   a. COPY文をデバッグ行に書いた場合、そのCOPY文によって複写される登録集原文中の原文語

b. 登録集原文中のデバッグ行上の原文語

c. 登録集原文中の置き換えの対象となった最初の原文語がデバッグ行に位置する場合、BYの右辺に書いた原文語

d. 仮原文-2の中のデバッグ行に書いた原文語

上記の原文語は、標識領域にデバッグ行を示す文字“D”を設定した行上に置かれます。これは、原文語の置き換えなどによって、登録集原文の1行が複数行になったとしても同じです。

9. COPY文の処理の結果、原文語の置き換えによって、定数-2、仮原文-2または登録集原文の中に書いた定数が複数行にまたがって複写される場合、前の行から継続して複写される行の標識領域には、行のつなぎを示す“－”が設定されます。ただし、デバッグ行上に複写される定数が継続されるような置き換えを行ってはいけません。

10. COPY文の処理の結果、原文語の置き換えによって新しい行が追加される場合、8.および9.の場合を除いて、追加される行の標識領域には、置き換えの対象となる原文語を含む行の標識領域の文字と同じ文字が設定されます。ただし、その行の標識領域がハイフンである場合、追加される行の標識領域には、空白が設定されます。

11. 登録集原文中の注記行および空白行は、そのまま複写されます。ただし、登録集原文中の注記行または空白行が、仮原文-1と一致する一連の原文語の間に存在する場合、注記行または空白行は複写されません。

12. 仮原文-2の中の注記行および空白行は、仮原文-2が複写されるときはいつでも、そのまま複写されます。

書き方2の規則

1. 書き方2のCOPY文による原文の置き換えは、以下の規則に従って行われます。

   a. 置き換えのための比較は、登録集原文中の、分離符のコンマおよび分離符のセミコロン以外の最初の原文語から開始します。この原文語を「現在の原文語」とします。この原文語の前にある空白、分離符のコンマおよび分離符のセミコロンは、そのまま複写します。

   b. 語-3を、「現在の原文語」と比較します。語-3を2つ以上書いた場合、それらを書いた順に、すべての語-3を比較します。比較の方法は、以下のとおりです。

      - 語-3に英数字の文字列を指定した場合、「現在の原文語」が、プレフィックスまたはサフィックスを持つ、英数字だけから構成される文字列かどうかを検査します。PREFIX指定を書いた場合、「現在の原文語」が、語-3、ハイフンおよび1文字以上の文字をこの順につないだ文字列である場合、一致したものとみなします。SUFFIX指定を書いた場合、「現在の原文語」が、1文字以上の文字、ハイフンおよび語-3をこの順につないだ文字列である場合、一致したものとみなします。

      - 語-3に日本語の文字列を指定した場合、「現在の原文語」が、プレフィックスまたはサフィックスを持つ、日本語文字だけから構成される文字列かどうかを検査します。PREFIX指定を書いた場合、「現在の原文語」が、語-3、JIS非漢字の負号、および1文字以上の日本語文字をこの順につないだ文字列である場合、一致したものとみなします。SUFFIX指定を書いた場合、「現在の原文語」が、1文字以上の日本語文字、JIS非漢字の負号および語-3をこの順につないだ文字列である場合、一致したものとみなします。

   c. b.の比較を、比較の結果が一致するまで、または語-3がなくなるまで繰り返します。

   d. b.の比較で、比較の結果が一致するものがなかった場合、「現在の原文語」をそのまま複写します。そして、「現在の原文語」の次の原文語を「現在の原文語」にして、b.の処理に戻ります。

   e. b.の比較で、比較の結果が一致した場合、「現在の原文語」の語-3と一致した部分を語-4に置き換えて複写します。そして、「現在の原文語」の次の原文語を「現在の原文語」にして、b.の処理に戻ります。

   f. b.～e.の処理を、登録集原文中の最後の原文語を比較するまで繰り返します。

2. 登録集原文中の置き換えの対象となった原文語は、新しい原文語(プレフィックスまたはサフィックスを語-4に置き換えたもの)に置き換えられて、正書法に従って複写されます。

3. 登録集原文中の原文語が、置き換えられるとき、新しい原文語(プレフィックスまたはサフィックスを語-4に置き換えたもの)は、置き換えの対象となった原文語の登録集原文中での領域と同じ領域から始まるように複写されます。

書き方3の規則

1. 登録集原文中の、レベル番号の直後またはREDEFINES句に書いたすべてのデータ名が、以下の規則に従って置き換えられます。その他の原文語は、そのまま複写されます。以下で、「元のデータ名」は、レベル番号の直後またはREDEFINES句に書いたすべてのデータ名を表します。

    a. 語-4に英数字の文字列を指定した場合、元のデータ名は英数字の文字列でなければなりません。PREFIXを書いた場合、元のデータ名を、語-4、ハイフンおよび元のデータ名をこの順につないだ文字列に置き換えて複写します。SUFFIXを書いた場合、元のデータ名を、元のデータ名、ハイフンおよび語-4をこの順につないだ文字列に置き換えて複写します。

    b. 語-4に日本語文字の文字列を指定した場合、元のデータ名は日本語文字の文字列でなければなりません。PREFIXを書いた場合、元のデータ名を、語-4、JIS非漢字の負号および元のデータ名をこの順につないだ文字列に置き換えて複写します。SUFFIXを書いた場合、元のデータ名を、元のデータ名、JIS非漢字の負号および語-4をこの順につないだ文字列に置き換えて複写します。

2. 登録集原文中の置き換えの対象となった原文語は、新しい原文語(語-4をプレフィックスまたはサフィックスとして付加したもの)に置き換えられて、正書法に従って複写されます。

3. 登録集原文中の原文語が置き換えられるとき、新しい原文語(語-4をプレフィックスまたはサフィックスとして付加したもの)は、置き換えの対象となった原文語の登録集原文中での領域と同じ領域から始まるように複写されます。

4. 原文名-1は、XMDLIBが指定された場合には画面帳票定義体名として、XFDLIBが指定された場合にはファイル定義体名として扱われます。
   XFDLIBは、[DS][Win16][Win32][Solaris][.NET]固有の機能です。

7.2 REPLACE文

原始プログラム原文を置き換えます。

【書き方1】置き換える文字列を指定し、置き換えの開始を宣言する

REPLACE {== 仮原文-1 == BY == 仮原文-2 ==} …

【書き方2】原文の置き換えの終了を宣言する

REPLACE OFF

構文規則

1. REPLACE文は、原始プログラム中で文字列が書けるところなら、どこにでも書くことができます。翻訳単位中の最初の文である場合を除いて、REPLACE文の前に分離符の終止符が先行しなければなりません。
2. REPLACE文は、分離符の終止符で止めなければなりません。
3. 仮原文-1および仮原文-2は、正書法に従って書かなければなりません。
4. 仮原文-1には、1つ以上の原文語を書かなければなりません。
5. 仮原文-2には、1つ以上の原文語を書きます。原文語は、省略することもできます。
6. 仮原文-1および仮原文-2の中の文字列は、正書法に従って、2行以上にわたって継続することができます。
7. 仮原文-1および仮原文-2の中の1つの原文語は、1～324文字でなければなりません。
8. 仮原文-1に、分離符のコンマまたは分離符のセミコロンだけを書くことはできません。
9. 語REPLACEを注記項、注記行または文字定数の中に書いた場合、語REPLACEは、それらを構成する文字列の一部とみなされます。

一般規則

1. REPLACE文は、翻訳時、原始プログラム原文を置き換えます。書き方1のREPLACE文は、原始プログラム中の仮原文-1と一致する文字列を、仮原文-2に置き換えることを指定し、原文の置き換えの開始を宣言します。書き方2のREPLACE文は、原文の置き換えの終了を宣言します。書き方1のREPLACE文の置き換えの規則は、次のREPLACE文が現れるまで、または翻訳単位の終わりまで、効果があります。
2. REPLACE文は、翻訳時、COPY文の処理の後に処理されます。REPLACE文は、実行時は意味を持ちません。
3. REPLACE文を処理した結果、新しいREPLACE文が生成されるようなREPLACE文を書くことはできません。
4. 原文の置き換えは、以下の順序で行われます。
    a. 置き換えのための比較は、原始プログラム中の最初の原文語から開始します。この原文語を「現在の原文語」とします。
    b. 仮原文-1に書いた文字列全体を、「現在の原文語」から始まる原始プログラム原文と1文字ずつ比較します。仮原文-1を2つ以上書いた場合、それらを書いた順に、すべての仮原文-1を比較します。このとき、仮原文-1および原始プログラム原文の中の、1つ以上の分離符のコンマ、1つ以上の分離符のセミコロン、および1つ以上の分離符の空白は、1つの空白とみなします。
    c. b.の比較を、比較の結果が一致するまで、または仮原文-1がなくなるまで繰り返します。
    d. b.の比較で、どの比較の結果も一致しなかった場合、「現在の原文語」の次の原文語を「現在の原文語」にして、b.の処理に戻ります。
    e. b.の比較で、比較の結果が一致した場合、仮原文-1と一致した部分を、正書法に従って、仮原文-2に置き換えます。そして、仮原文-1と一致した部分の最後の原文語の次の原文語を「現在の原文語」として、b.の処理に戻ります。
    f. b.～e.の処理を、原始プログラム中の最後の原文語を比較するまで繰り返します。
5. 原始プログラムおよび仮原文-1の中の注記行および空白行は、置き換えのための比較において無視されます。
6. 仮原文-1および仮原文-2の中に、デバッグ行を書くことができます。比較を行うとき、仮原文-1および仮原文-2の中の標識領域の“D”は無視されます。
7. 原始プログラム原文中の仮原文-1と一致した部分が仮原文-2に置き換えられるとき、仮原文-2のそれぞれの原文語は、正書法に従って挿入されます。仮原文-2の中の原文語間の空白は、そのまま挿入されます。
8. REPLACE文の処理の結果、原文の置き換えによって、仮原文-2の中に書いた定数が複数行にまたがって挿入される場合、前の行から継続して挿入される行の標識領域には、行のつなぎを示す“－”が設定されます。ただし、デバッグ行として挿入される定数が継続されるような置き換えを行ってはいけません。
9. REPLACE文の処理の結果、原文の置き換えによって新しい行が追加される場合、8.の場合を除いて、追加される行の標識領域には、置き換えの対象となる原文を含む行の標識領域の文字と同じ文字が設定されます。ただし、その行の標識領域がハイフンである場合、追加される行の標識領域には、空白が設定されます。
10. 仮原文-2の中の注記行および空白行は、仮原文-2が挿入されるときはいつでも、そのまま挿入されます。
11. REPLACE文により、プログラム名段落およびプログラム終わり見出しを変更してはいけません。

第8章 データベース（SQL）

データベース(SQL)機能は、このコンパイラの規定する埋込みSQL文を使用して、各種データベースにアクセスし、データ操作をすることができます。

データベース(SQL)機能は、[Win16][Win32][WinIPF][Winx64][Linux][.NET]固有の機能です。

データベース

データベースは、二次元の表の集合であり、表は、行および列の集合で構成されます。

データベース中のデータは、利用者が示す値によって動的に関係付けることができます。利用者は、データの物理的な配置や順序を意識せずに表中のデータを呼び出すことができます。処理はすべて、論理的な利用者インタフェースである表への操作で行います。

表

実表とビュー表を総称して表と呼びます。データは実表に格納されます。ビュー表は、データ操作を行う際に使用する仮想的な表で、データの実体は存在しません。

埋込みSQL

埋込みSQLは、応用プログラムからデータベースを操作するための記述です。埋込みSQLは、以下の2つから構成されます。

埋込みSQL宣言節

埋込みSQL宣言節は、埋込みSQL開始宣言と、埋込みSQL終了宣言によって区切られる節です。埋込みSQL宣言節では、ホスト変数を定義します。

埋込みSQL文

埋込みSQL文は、データベースを操作したり、例外処理を記述したりするための文です。

本章で規定する埋込みSQL文は、このコンパイラの規定する埋込みSQL文です。埋込みSQL文の動作および記述の詳細については、各データベースの仕様に従います。

ホスト変数

ホスト変数は、データベースと応用プログラムの間で、データを受け渡すために使用するデータ項目です。応用プログラムからデータをデータベースに格納したり、データベースのデータを読み込んだりするために使用します。

ホスト変数の定義には、データベースの1つの列を、基本項目として定義する方法(単一列指定ホスト変数)と、集団項目として定義する方法(複数列指定ホスト変数、複数行指定ホスト変数、表指定ホスト変数)があります。

8.1 埋込みSQLの正書法

ここでは、埋込みSQLの正書法について説明します。

8.1.1 全般規定

埋込みSQL開始宣言、埋込みSQL終了宣言および埋込みSQL文は、SQL先頭子(“EXEC SQL”)とSQL終了子(“END-EXEC”)で囲んで記述しなければなりません。

埋込みSQL開始宣言、埋込みSQL終了宣言および埋込みSQL文は、すべてB領域に記述しなければなりません。

8.1.2 行のつなぎ

埋込みSQL開始宣言、埋込みSQL終了宣言および埋込みSQL文の継続(行のつなぎ)の規則は、COBOLの正書法の規則に準じます。

8.1.3 COBOLの注記行および行内注記

COBOLの注記行、デバッグ行および空白行は、埋込みSQLの記述中に書くことができます。しかし、行内注記は記述できません。

8.2 データ部

データ部では、埋込みSQL宣言節を記述し、ホスト変数を定義します。

8.2.1 埋込みSQL宣言節

埋込みSQL宣言節では、ホスト変数を宣言します。

【書き方】

EXEC SQL

　　BEGIN DECLARE SECTION END-EXEC.

[{ ホスト変数定義 } …]

EXEC SQL

　　END DECLARE SECTION END-EXEC.

一般規則

1. 埋込みSQL宣言節は、データ部の以下の節に記述することができます。
   - 作業場所節
   - 連絡節
2. 埋込みSQL開始宣言(EXEC SQL BEGIN DECLARE SECTION END-EXEC)と埋込みSQL終了宣言(EXEC SQL END DECLARE SECTION END-EXEC)は省略できません。
3. 埋込みSQL開始宣言と埋込みSQL終了宣言は、対で指定しなければなりません。また、入れ子であってはなりません。
4. 埋込みSQL開始宣言または埋込みSQL終了宣言をはさんで，データを再定義することはできません。
5. 埋込みSQL宣言節は、埋込みDCSQL宣言節内に含まれてはなりません。また、埋込みSQL宣言節内に、埋込みDCSQL宣言節を含んではなりません。なお、この規則は、[Win16]固有の規則です。

8.2.2 ホスト変数定義

ホスト変数定義は、ホスト変数の性質を指定します。

【書き方】

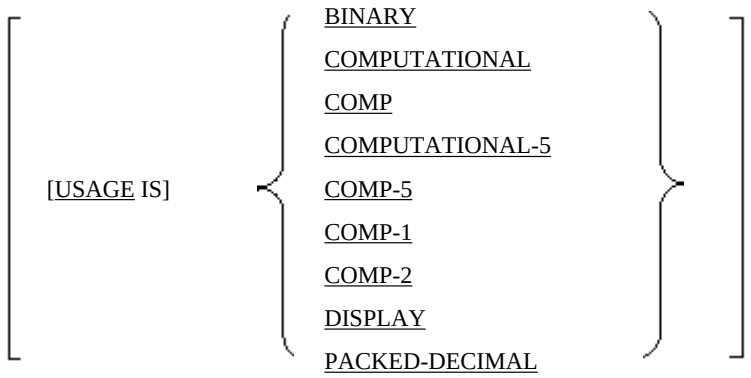
レベル番号 ホスト変数名

　[REDEFINES句]

　[IS EXTERNAL]

　[IS GLOBAL]

　[{ PICTURE / PIC } IS 文字列]



[[SIGN IS]

{ LEADING SEPARATE CHARACTER / TRAILING }]

[OCCURS 整数-1 [TIMES]].

REDEFINES句の記述は、[Win32][WinIPF][Winx64][.NET]固有機能です。

構文規則

1. ホスト変数は、レベル番号01～48または77の項目でなければなりません。
2. データベースの可変長文字列のデータに対応するホスト変数は、レベル番号49の基本項目を従属する集団項目でなければなりません。
3. OCCURS句を基本項目に指定する場合、その項目はレベル番号49の基本項目であってはなりません。また、OCCURS句を集団項目に指定する場合、その項目はデータベースの可変長文字列のデータに対応する集団項目でなければなりません。
4. REDEFINES句を指定する場合、その項目はレベル番号49の項目であってはなりません。
5. ホスト変数は、レコード中の可変位置にある項目であってはなりません。
6. 各句の構文規則は、COBOLのデータ記述項における各句の構文規則と同じです。

一般規則

1. 各句の一般規則は、COBOLのデータ記述項における各句の一般規則と同じです。
   ただし、集団項目および集団に従属するホスト変数名が大域名である場合、それは参照できる範囲内で修飾することなく一意になる名前でなければなりません。
2. 単一列指定ホスト変数は、以下に示すいずれかのデータ型をもたなければなりません。
   a. 4桁または9桁の符号付き2進整数項目
   b. 18桁以下の符号付き内部10進項目
   c. 18桁以下の符号付き外部10進項目
   d. 英数字項目(項目長については、“NetCOBOL使用手引書”を参照してください。)
   e. 日本語項目(項目長については、“NetCOBOL使用手引書”を参照してください。)
   f. 単精度内部浮動小数点項目
   g. 倍精度内部浮動小数点項目

h. 以下のデータ型およびデータ構造を持つ集団項目(これは、可変長文字列型です。mまたはnの値については、“NetCOBOL使用手引書”を参照してください。)

```
      レベル番号  データ名-1.
          49  データ名-2  PIC  S9(m)  BINARY.
          49  データ名-3  PIC  X(n).
または
      レベル番号  データ名-1.
          49  データ名-2  PIC  S9(m)  COMP-5.
          49  データ名-3  PIC  X(n).
または
      レベル番号  データ名-1.
          49  データ名-2  PIC  S9(m)  BINARY.
          49  データ名-3  PIC  N(n).
または
      レベル番号  データ名-1.
          49  データ名-2  PIC  S9(m)  COMP-5.
          49  データ名-3  PIC  N(n).
```

3. 日本語項目の使用の可否は、データベースおよびその関連製品に依存します。

4. 複数列指定ホスト変数は、データベースの複数の列に対応するホスト変数です。

```
レベル番号-1  データ名-1.    ←  複数列指定ホスト変数
    単一列指定ホスト変数-1
    単一列指定ホスト変数-2
            :
    単一列指定ホスト変数-x
```

5. 複数行指定ホスト変数は、OCCURS句を持つ基本項目であり、データベースの複数の行に対応するホスト変数です。

```
レベル番号-1  データ名-1.
    単一列指定ホスト変数  OCCURS  整数-1  ←  複数行指定ホスト変数
```

6. 表指定ホスト変数は、複数行指定ホスト変数だけを従属する集団項目であり、行と列からなるデータベースの表に対応するホスト変数です。

```
レベル番号-1  データ名-1.    ←  表指定ホスト変数
    複数行指定ホスト変数-1
    複数行指定ホスト変数-2
            :
    複数行指定ホスト変数-x
```

7. 複数行指定ホスト変数または表指定ホスト変数を相手指定で使用する場合、整数-1には取り出すデータの行数を指定します。複数行指定ホスト変数または表指定ホスト変数を値指定で使用する場合、整数-1にはそのSQL文を実行する回数を指定します。

8. 表指定ホスト変数に従属する項目の整数-1の値を、この表指定ホスト変数の反復回数とみなします。

9. 表指定ホスト変数に従属する項目の間で、整数-1の値は同じでなければなりません。

8.2.3 ホスト変数の参照

ホスト変数は、埋込みSQLの中でも、一般のCOBOL文の中でも、参照することができます。埋込みSQL文中でホスト変数を参照する場合、コロン(:)をホスト変数の直前に付加しなければなりません。たとえば、“A”という名前のホスト変数があれば、埋込みSQL文中では“:A”と記述します。

ホスト変数を一般のCOBOL文の中で参照する場合は、コロンを付加してはなりません。

なお、SQLSTATE、SQLCODE、SQLMSGまたはSQLERRDを、埋込みSQL文中でホスト変数として参照してはなりません。

8.2.4 SQLSTATE/SQLCODE

SQL文の実行によって例外事象が発生した場合、例外事象の内容を示すコードが応用プログラムに通知されます。SQLSTATE/SQLCODEは、このコードを格納するための領域です。利用者は、SQLSTATE/SQLCODEを参照することにより、例外事象に応じた処理を行うことができます。

SQLSTATE

- [Win16][WinIPF][Linux]の場合、SQLSTATEは、レベル番号01または77の、5桁の英数字項目として、埋込みSQL宣言節で必ず定義しなければなりません。
- [Win32][Winx64][.NET]の場合、SQLSTATEは、レベル番号01、77またはレベル番号01のSQLCAに直接従属する、5桁の英数字項目として定義しなければなりません。また、レベル番号01または77のSQLSTATEと、SQLCAに従属するSQLSTATEの両方を定義することはできません。SQLCAに従属するSQLSTATEのレベル番号は02～49のいずれかでなければなりません。
- SQLSTATEに、REDEFINES句を指定することはできません。

SQLCODE

- [Win16][WinIPF][Linux]の場合、SQLCODEは、レベル番号01または77の、9桁の符号付き2進整数項目として、埋込みSQL宣言節で定義しなければなりません。
- [Win32][Winx64][.NET]の場合、SQLCODEは、レベル番号01、77またはレベル番号01のSQLCAに直接従属する、9桁の符号付き2進整数項目として定義しなければなりません。また、レベル番号01または77のSQLCODEと、SQLCAに従属するSQLCODEの両方を定義することはできません。SQLCAに従属するSQLCODEのレベル番号は02～49のいずれかでなければなりません。
- SQLCODEに、REDEFINES句を指定することはできません。

8.2.5 SQLMSG

SQL文の実行によって例外事象が発生した場合、例外事象の内容を示すメッセージが応用プログラムに通知されます。SQLMSGは、このメッセージを格納するための領域です。利用者は、出力文を利用して、SQLMSGの内容を印字または表示することができます。

- [Win16][WinIPF][Linux]の場合、SQLMSGは、レベル番号01または77の英数字項目として、埋込みSQL宣言節で定義しなければなりません。
- [Win32][Winx64][.NET]の場合、SQLMSGは、レベル番号01または77の英数字項目として定義しなければなりません。
- SQLMSGに、REDEFINES句を指定することはできません。

8.2.6 SQLERRD

SQLERRDは、以下の2つの情報を格納するための領域です。

ストアドプロシージャの戻り値

[Win32][WinIPF][Winx64][.NET]では、利用者は、配列の1番目の要素であるSQLERRD(1)を参照して、ストアドプロシージャの戻り値を知ることができます。

処理された行

複数行指定ホスト変数を使用して複数行処理が行われた場合、処理された行数が応用プログラムに通知されます。利用者は、配列の3番目の要素であるSQLERRD(3)を参照して、処理行数を知ることができます。

【書き方1】

```
01  SQLINFOA.
```

02 SQLERRD PIC S9 (9) { COMP / COMP-5 / BINARY } OCCURS 6 [TIMES].

【書き方2】

01 SQLCA.

レベル番号 SQLERRD PIC S9 (9) { COMP / COMP-5 / BINARY } OCCURS 6 [TIMES].

【書き方2】は、[Win32][Winx64][.NET]固有の機能です。

構文規則

書き方1および書き方2に共通する規則

- [Win16][WinIPF][Linux]の場合、SQLERRDは、埋込みSQL宣言節で定義しなければなりません。
- SQLINFOAに従属するSQLERRDとSQLCAに従属するSQLERRDの両方を定義することはできません。

書き方2の規則

[Win32][Winx64][.NET]SQLCAに従属するSQLERRDのレベル番号は02～49のいずれかでなければなりません。

一般規則

ストアドプロシージャの戻り値はSQLERRD(1)に、処理された行数はSQLERRD(3)に通知されます。その他の領域はシステム予約域のため、使用してはなりません。

SQLERRD(1)は[Win32][WinIPF][Winx64][.NET]固有です。

8.3 手続き部

手続き部では、埋込みSQL文を記述します。

本節の【書き方】で使う記号の意味を、以下に示します。

＜＞

文を構成する要素の名前を示します。

::＝

要素を構成するための演算子です。ここで定義する要素は、この演算子の左側に現れ、その要素を定義する公式は右側に現れます。

[ ]

省略可能な要素を示します。

{ }

波括弧内に書いたいくつかの要素の内の1つを、利用者が選択することを示します。選択できる要素は、縦棒(“|”)で区切っているか、縦に並べて記述しています。

|

縦棒の後に続く要素が、縦棒の前の要素の代わりに記述できることを示します。縦棒は波括弧の中で使用します。

…

前の要素を繰り返すことを示します。

8.3.1 文字

SQL文に記述できる文字は、以下の4種類です。

英字

英字については、“1.1 文字と文字集合”を参照してください。

数字

数字については、“1.1 文字と文字集合”を参照してください。

特殊文字

特殊文字は、以下の18個です。

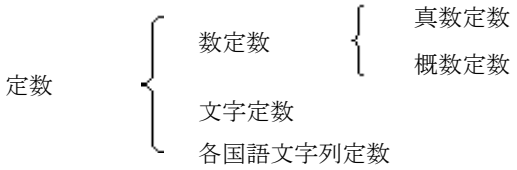
.	<	(	+	*	)	;	−	/	,	%	_	>	?	:	=	”	’

各国語文字

各国語文字の使用の可否については、データベースおよびその関連製品に依存します。

8.3.2 定数

埋込みSQLの定数は、ナル値でない値を指定します。定数には、以下の種類があります。



8.3.2.1 数定数

数定数は、数値を値として持つ定数です。

【書き方】

<数定数> ::= <真数定数> | <概数定数>

<真数定数> ::=

[+ | -]{<符号なし整数>[.<符号なし整数>] | <符号なし整数>. | .<符号なし整数>}

<概数定数> ::= <仮数>E<指数>

<仮数> ::= <真数定数>

<指数> ::= [+ | -]<符号なし整数>

<符号なし整数> ::= <数字> …

参照

“8.3.1 文字”の<数字>

構文規則

1. 真数定数は、0～9の数字および小数点からなる文字列です。
2. 真数定数は、正号または負号を指定することができます。

3. 真数定数のデータ型は真数です。真数定数の精度は、それが含む数字の個数です。真数定数の位取りは、小数点の右側の数字の個数です。
4. 真数定数の精度および位取りは、データ型が真数型であるホスト変数の精度および位取りの規則に従います。
5. 概数定数は、0～9の数字、小数点および文字“E”からなる文字列です。
6. 概数定数は、正号または負号を指定することができます。
7. 概数定数のデータ型は、概数です。概数定数の精度は、その仮数の精度です。仮数は、真数定数と同じ形式で指定します。
8. 概数定数の精度および位取りは、データ型が概数型であるホスト変数の精度および位取りの規則に従います。

一般規則

1. 真数定数は、固定小数点の数値データを表します。
2. 真数定数の数値は、符号付き位取り10進表記法の通常の数学的解釈から導出されます。
3. 概数は、浮動小数点の数値データを表します。
4. 概数定数の数値は、仮数によって表される真数値と、指数によって表される数の10のべき乗によって得られる数の積です。

8.3.2.2 文字列定数

文字列定数は、文字を値として持つ定数です。

【書き方】

'<文字> …'

参照

“8.3.1 文字”

構文規則

1. 文字列定数は、その始まりと終わりを引用符で区切った文字列であり、文字のデータを指定します。
2. 文字列定数のデータ型は文字列です。
3. 文字列定数の長さは、データ型が文字列型であるホスト変数の長さの規則に従います。

一般規則

文字列定数の値は、それが含む文字の並びです。

8.3.2.3 各国語文字列定数

各国語文字列定数は、各国語文字を値として持つ定数です。各国語文字列定数の使用の可否については、データベースおよびその関連製品に依存します。

【書き方】

N'<各国語文字> …'

参照

"8.3.1 文字"の<各国語文字>

構文規則

1. 各国語文字列定数は、その始まりを"N"と引用符の並びで、終わりを引用符で区切った各国語文字列であり、各国語文字のデータを指定します。
2. 各国語文字列定数のデータ型は各国語文字型です。各国語文字列定数の長さはそれが含む各国語文字の個数です。
3. 各国語文字列定数の長さは、データ型が各国語文字列であるホスト変数の長さの規則に従います。

一般規則

各国語文字列定数の値は、それが含む各国語文字の並びです。

8.3.3 トークン

SQL文を構成する最小単位を指定します。

【書き方】

<トークン> ::= <非区切りトークン> | <区切りトークン>

<非区切りトークン> ::= <識別子> | <キーワード> | <数定数>

<識別子> ::= <英字>[{[<下線>] {<英字> | <数字>}}…] | <各国語文字識別子>

<各国語文字識別子> ::= <各国語文字列定数>

<下線> ::= _

<キーワード> ::= "NetCOBOL使用手引書"参照

<区切りトークン> ::= <文字列定数> | <各国語文字列定数>

| , | (|) | < | > | . | : | = | * | + | - | / | <> | >= | <=

<分離符号> ::= {<注釈> | <空白文字> | <改行>} …

<注釈> ::= <注釈導入子> [{<文字> | <各国語文字>} …]

<注釈導入子> ::= - -

参照

"8.3.1 文字"

"8.3.1 文字"の<英字>

"8.3.1 文字"の<数字>

"8.3.1 文字"の<各国語文字>

"8.3.2.1 数定数"

"8.3.2.2 文字列定数"

“8.3.2.3 各国語文字列定数”

構文規則

注釈導入子は、1つの空白または改行によって分離されず、かつ定数に含まれない、2つ以上の連続する“ー”(ハイフン)の並びです。

8.3.4 名前

埋込みSQLの名前には、以下の6種類があります。

表名

表名は、実表の名前です。表名は、データ操作の対象とする表を指定する場合に使用します。表名は、修飾することもできます。表名を修飾する場合は、修飾される表名と修飾する名前を“.”で区切って表現します。

カーソル名

カーソル名は、カーソルに付ける名前です。カーソルは、表の中の1行を特定する行指示子であり、カーソル宣言で定義します。

カーソル名は、18文字以下の英数字または18文字以下の各国語文字の並びでなければなりません。

列名

列名は、表の各列の名前です。列名は、表名または相関名で修飾することができます。列名を修飾する場合は、修飾される列名と修飾する名前を“.”で区切って表現します。

相関名

相関名は、表に付ける別名です。

SQL文識別子

SQL文識別子は、動的SQL文で使用する文の識別子です。SQL文識別子は、18文字以下の英数字または18文字以下の各国語文字の並びでなければなりません。

ストアドプロシージャ名

ストアドプロシージャ名は、ストアドプロシージャ定義で定義される名前です。ストアドプロシージャ名は、修飾することもできます。ストアドプロシージャ名を修飾する場合は、修飾されるストアドプロシージャ名と修飾する名前を“.”で区切って表現します。

ストアドプロシージャ名は、修飾する名前を含めて、90文字以下の英数字または90文字以下の各国語文字の並びでなければなりません。

8.3.5 値指定と相手指定

値指定では、値をホスト変数、標識変数および定数で指定します。

相手指定は、値を代入するホスト変数および標識変数を指定します。

【書き方】

<値指定> ::= <変数指定> | <定数>

<変数指定> ::= <ホスト変数名> [<標識変数>]

<動的パラメタ指定> ::= ?

<標識変数> ::= [INDICATOR] <ホスト変数名>

<ホスト変数名> ::= [{<ホスト変数>.} …] <ホスト変数>

<ホスト変数> ::= 各規則参照

<相手指定> ::= <変数指定>

参照

“8.3.2 定数”

構文規則

1. ホスト変数名は、ホスト変数に付ける名前です。ホスト変数は、手続き部の埋込みSQL文およびCOBOLの【書き方】中で、一意名やデータ名が指定可能なところに書くことができます。
2. ホスト変数名は、30文字以下の英数字または30文字以下の各国語文字の並びでなければなりません。その構成は、COBOLの利用者語の規則に従います。
3. ホスト変数は、埋込みSQL宣言節であらかじめ定義しておかなければなりません。
4. ホスト変数をSQL文中で部分参照または添字付けすることはできません。
5. ホスト変数として複数列指定ホスト変数または表指定ホスト変数が記述された場合、それに従属する項目が定義された並びですべてホスト変数に指定されたものと同様に扱います。
6. ホスト変数を一意にするために修飾することができます。SQL文中でホスト変数を修飾する場合は、修飾する名前を先に、修飾される名前を後に書き、両者を“.”(ピリオド)で区切って表記します。
7. 修飾する名前は、修飾される項目を直接従属する集団項目の名前でなければなりません。
8. 名前が一意になるのであれば、修飾される名前に最上位階層までのすべての上位項目名を書かなくてもかまいません。
9. 標識変数で指定したホスト変数が、標識変数になります。
10. 標識変数は、4桁の符号付き2進整数項目でなければなりません。
11. 標識変数は、表の列にホスト変数の値を設定するとき、または列からホスト変数に値を読み込むときに、そのホスト変数と対にして指定することができます。
12. 動的パラメタ指定は、準備可能文の中で値指定として変数を指定する場合に使用します。動的パラメタ指定は、値指定を記述する部分に“?”(疑問符)を指定します。準備可能文の中の動的パラメタ指定は、埋込みSQL文の変数指定に相当します。

一般規則

1. 複数列指定ホスト変数または表指定ホスト変数は、以下のSQL文で使用できます。
   - SELECT文の相手指定
   - INSERT文
   - FETCH文
   - EXECUTE文のUSING句およびINTO句
2. 複数行指定ホスト変数は、以下のSQL文で使用できます。
   - SELECT文の相手指定
   - INSERT文
   - FETCH文
   - UPDATE文(探索)の設定句とWHERE句

- DELETE文(探索)
- EXECUTE文のUSING句およびINTO句

3. 複数行指定ホスト変数を記述できるところでは、複数行指定ホスト変数と、単一列指定ホスト変数または複数列指定ホスト変数を混在させてはいけません。
4. 表指定ホスト変数を記述できるところでは、表指定ホスト変数と、単一列指定ホスト変数または複数列指定ホスト変数を混在させてはなりません。
5. 1つのSQL文中で使用される複数行指定ホスト変数および表指定ホスト変数の従属項目に指定された反復回数は、すべて同じでなければなりません。同じでない場合、最小の値を反復回数とみなします。
6. 標識変数によって関連付けたホスト変数の値を、表の列に設定するとき、標識変数に以下の値を設定しておくことにより、ナル値を設定するかどうかの制御を行うことができます。
   - 0または正: 標識変数に関連付けたホスト変数の値が、表の列に設定されます。
   - 負: ナル値が表の列に設定されます。この場合、標識変数に関連付けたホスト変数の内容は無視されます。
7. 標識変数に関連付けたホスト変数に、列から値を読み込むとき、そのSQL文を実行した後、データベースシステムにより以下の値が標識変数に設定されます。
   - 0または正: 入力した値がナル値でなく、かつ入力した値が標識変数に関連付けたホスト変数に格納されています。
   - 負: 入力した値がナル値です。この場合、標識変数に関連付けたホスト変数の内容は意味を持ちません。
8. 複数列指定ホスト変数に対応する標識変数は、従属項目数が同じである複数列指定ホスト変数または従属項目数と同じ数の繰返し回数を指定したOCCURS句を持つ基本項目でなければなりません。ただし、標識変数の従属項目数または繰返し回数が、ホスト変数の従属項目数より大きくてもかまいません。
9. 複数行指定ホスト変数に対応する標識変数は、同じ繰返し回数を指定した複数行指定ホスト変数でなければなりません。ただし、標識変数の繰返し回数が、ホスト変数の繰返し回数より大きくてもかまいません。
10. 表指定ホスト変数に対応する標識変数は、同じ従属項目数と繰返し回数を持つ表指定ホスト変数でなければなりません。ただし、標識変数の従属項目数または繰返し回数が、ホスト変数の従属項目数または繰返し回数がより大きくてもかまいません。
11. 標識変数の従属項目は、ホスト変数の従属項目の定義の並び順に対応付けられます。配列は、配列要素の先頭から順に対応付けられます。
12. 標識変数の従属項目数または繰返し回数が、ホスト変数の従属項目数または繰返し回数より大きい場合、残りの標識変数は無視されます。

8.3.6 列指定

列指定は、列名で指定された列を参照します。列名については、“列名”を参照してください。

8.3.7 集合関数指定

集合関数指定は、関数を適用することによって導出される値を指定します。

【書き方1】表の全行数を返す

COUNT(*)

【書き方2】DISTINCT集合関数

{ AVG / MAX / MIN / SUN / COUNT } (DISTINCT 列指定)

【書き方3】ALL集合関数

{AVG | MAX | MIN | SUM} ([ALL] 値式)

参照

"8.3.6 列指定"

"8.3.8 値式"

一般規則

1. 書き方1の集合関数は、ナル値を持つ列を含む、すべての行の数を返します。
2. DISTINCT集合関数は、ナル値を持つ列を取り除き、重複する値を持つ列を1つにした結果を返します。
3. ALL集合関数は、ナル値を持つ列を取り除いた結果を返します。
4. 書き方2、書き方3のそれぞれの集合関数の機能を以下に示します。
   - AVG: 平均値を求めます。
   - MAX: 最大値を求めます。
   - MIN: 最小値を求めます。
   - SUM: 総和を求めます。
   - COUNT: 表の基数を求めます。

8.3.8 値式

値式は、値を指定します。

【書き方】

<値式> ::= <項> | <値式>+<項> | <値式>-<項>
<項> ::= <因子> | <項>*<因子> | <項>/<因子>
<因子> ::= [+ | -]<一次子>
<一次子> ::=
    <列指定> | <動的パラメタ指定> | <定数> | <集合関数指定> | (<値式>)

参照

“8.3.6 列指定”

“8.3.5 値指定と相手指定”の<動的パラメタ指定>

“8.3.2 定数”

“8.3.7 集合関数指定”

構文規則

単項演算子は、連続して記述することはできません。

一般規則

1. 演算子は、データ型が真数型または概数型の定数にだけ指定することができます。
2. 単項演算子“＋”は、直後に続く要素に何も作用しません。単項演算子“－”は、直後に続く要素に-1を掛けることを示します。
3. 二項演算子“＋”、“－”、“＊”、“/”はそれぞれ、加算、減算、乗算、除算を示します。
4. 評価は、括弧の中の式、単項演算子、乗除算演算および加減算の順に行います。同順位の演算子の評価は、左から右に行います。
5. 除数は、0であってはなりません。

8.3.9 述語

述語は、“真”、“偽”または“不定”の真偽値の表ができるための条件を指定します。述語には、以下の7種類があります。

- 比較述語
- BETWEEN述語
- IN述語
- LIKE述語
- NULL述語
- 限定述語
- EXISTS述語

8.3.9.1 比較述語

2つの値について比較した結果を判定します。

【書き方】

<比較述語> ::= <値式><比較演算子>{<値式> | <副問合せ>}

<比較演算子> ::= {= | < | <= | > | >= | <>}

参照

“8.3.8 値式”

“8.3.14 副問合せ”

一般規則

最初の値式のデータ型と、2番目の値式のデータ型または副問合せは、比較可能でなければなりません。副問合せの結果の基数は、1でなければなりません。

8.3.9.2 BETWEEN述語

範囲比較を指定します。

【書き方】

<BETWEEN述語> ::=

<値式> [NOT] BETWEEN <値式> AND <値式>

参照

"8.3.8 値式"

一般規則

1. 3つの値式のデータ型は、比較可能でなければなりません。
2. 値式には、動的パラメタ指定または変数指定は記述できません。

8.3.9.3 IN述語

ある限定された値の集合について比較を指定します。

【書き方】

<IN述語> ::= <値式> [NOT] IN {(<副問合せ>) | (<限定値リスト>)}

<限定値リスト> ::= <限定値>{,<限定値>} …

<限定値> ::= {<定数> | <変数指定> | USER | <動的パラメタ指定>}

参照

"8.3.8 値式"

"8.3.14 副問合せ"

"8.3.2 定数"

"8.3.5 値指定と相手指定"の<変数指定>

"8.3.5 値指定と相手指定"の<動的パラメタ指定>

一般規則

最初の値式のデータ型と、副問合せまたは限定値リストの中のすべての値指定のデータ型は、比較可能でなければなりません。

8.3.9.4 LIKE述語

文字型データについて、照合比較を指定します。

【書き方】

<LIKE述語> ::= <照合値> [NOT] LIKE <パターン>

<照合値> ::= <列指定>

<パターン> ::= <文字列定数> | <変数指定> | USER | <動的パラメタ指定>

参照

"8.3.6 列指定"

"8.3.2 定数"の<文字列定数>

"8.3.5 値指定と相手指定"の<変数指定>

"8.3.5 値指定と相手指定"の<動的パラメタ指定>

一般規則

1. 列指定は、文字型または各国語文字型の列で参照しなければなりません。
2. 列指定で指定した列データ型が文字列の場合、パターンは文字列でなければなりません。列指定で指定した列データ型が各国語文字列の場合、パターンは各国語文字列でなければなりません。
3. パターンは、文字列と文字の"%"(パーセント記号文字)および"_"(下線文字)を組み合わせて表現することができます。また、各国語文字列と各国語文字のパーセント記号文字および下線文字を組み合わせて表現することができます。
4. パーセント記号文字は、0個以上の任意の文字に対応します。
5. 下線文字は、1個の任意の文字に対応します。

8.3.9.5 NULL述語

ナル値のテストを指定します。

【書き方】

<NULL述語> ::= <列指定> IS [NOT] NULL

参照

"8.3.6 列指定"

8.3.9.6 限定述語

限定された値の集合との比較を指定します。

【書き方】

<限定述語> ::= <値式> <比較演算子> {ALL | ANY} <副問合せ>

参照

"8.3.8 値式"

"8.3.9.1 比較述語"の<比較演算子>

"8.3.14 副問合せ"

一般規則

値式と副問合せのデータ型は、比較可能でなければなりません。

8.3.9.7 EXISTS述語

空集合のテストを実施します。

【書き方】

<EXISTS述語> ::= EXISTS <副問合せ>

参照

"8.3.14 副問合せ"

8.3.10 探索条件

ブール演算子を適用した結果によって、真、偽または不定となる条件を指定します。

【書き方】

<探索条件> ::= <ブール項> [OR <search condition>]

<ブール項> ::= <ブール因子> [AND <ブール項>]

<ブール因子> ::= [NOT] <ブール一次子>

<ブール一次子> ::= ＜述語＞ | (＜探索条件>)

参照

"8.3.9 述語"

8.3.11 表式

表式は、表またはグループ表を指定します。

【書き方】

<表式> ::=

<FROM句>

[<WHERE句>]

[<GROUP BY句>]

[<HAVING句>]

8.3.11.1 FROM句

1つ以上の名前付きの表から導出される表を指定します。

【書き方】

<FROM句> ::= FROM <表参照> [{,<表参照>} …]

<表参照> ::= <表名> [<相関名>]

参照

“表名”

“相関名”

8.3.11.2 WHERE句

先行するFROM句の結果に探索条件を適用することによって、導出される表を指定します。

【書き方】

<WHERE句> ::= WHERE <探索条件>

参照

“8.3.10 探索条件”

8.3.11.3 GROUP BY句

FROM句およびWHERE句によって指定された結果にGROUP BY句を適用することによって、導出されるグループ表を指定します。

【書き方】

<GROUP BY句> ::= GROUP BY <列指定> [{,<列指定>} …]

参照

“8.3.6 列指定”

8.3.11.4 HAVING句

GROUP BY句またはFROM句の結果であるグループ表に対し、探索条件に合うグループだけを選択するように指定します。

【書き方】

<HAVING句> ::= HAVING <探索条件>

参照

“8.3.10 探索条件”

8.3.12 問合せ指定

表式の結果から導出される表を指定します。

【書き方】

SELECT [ALL|DISTINCT] <選択リスト > <表式>

<選択リスト > ::= * | <選択副リスト> [{,<選択副リスト>} …]

<選択副リスト > ::= <値式> | <表名>.* | <相関名>.*

参照

“8.3.11 表式”

“8.3.11.1 FROM句”

“8.3.11.2 WHERE句”

“8.3.8 値式”

“表名”

“相関名”

8.3.13 問合せ式

表を指定します。

【書き方】

<問合せ式 > ::=

{<問合せ指定> | (<問合せ式>)} |

<問合せ式> UNION [ALL] {<問合せ指定> | (<問合せ式>)}

参照

“8.3.12 問合せ指定”

8.3.14 副問合せ

表式から導出される表を指定します。

【書き方】

<副問合せ> ::= (SELECT [ALL | DISTINCT] <選択リスト> <表式>)

参照

“8.3.12 問合せ指定”の<選択リスト>

“8.3.11 表式”

8.3.15 FOR句

SQL文の実行回数またはSQL文で操作する行数を指定します。

【書き方】

<FOR句> ::= FOR <ホスト変数> | <値指定>

構文規則

1. FOR句に指定する値は、1以上の整数でなければなりません。最大値は“付録B システムの定量制限”で定義されるOCCURS句の繰返し回数の上限です。
2. FOR句に指定するホスト変数は、符号付き4バイト2進整数項目でなければなりません。
3. FOR句に指定するホスト変数は、修飾なしで一意でなければなりません。
4. FOR句は、複数行指定ホスト変数または表指定ホスト変数を指定した以下のSQL文にだけ記述できます。
   - SELECT文
   - INSERT文

- FETCH文
- UPDATE文(探索)
- DELETE文(探索)
- EXECUTE文

一般規則

1. データベースから値を取り出すときに指定したFOR句の値は、取り出す行数を示します。データベースの値を操作するときに指定したFOR句の値は、操作回数を示します。
2. FOR句に0または負の値が指定された場合、実行は行われません。
3. FOR句に指定した値が、複数行指定ホスト変数または表指定ホスト変数の反復回数よりも大きい場合、実行は行われません。
4. FOR句を被準備文中に記述してはなりません。

8.4 埋込み例外宣言

埋込み例外宣言は、SQL文に例外事象が発生したときにとる動作を指定します。

【書き方】

WHENEVER { SQLERROR / NOT FOUND } { GO TO : 手続き名 / CONTINUE }

一般規則

1. SQLERRORを指定したとき、埋込み例外宣言は、SQLSTATE/SQLCODEの値が正常終了またはデータなしのどちらも示さないときに実行する文を指定します。
2. NOT FOUNDを指定したとき、埋込み例外宣言は、SQLSTATE/SQLCODEの値がデータなしを示すときに実行する文を指定します。
3. 埋込み例外宣言は、原始プログラム上の次の文以降、同一の条件を指定した埋込み例外宣言が現れるか、またはプログラムの終わりに達するまで、その間にあるすべてのSQL文に対して有効です。
4. 埋込み例外宣言の有効範囲は、プログラム単位です。

8.5 非カーソル系データ操作文

非カーソル系データ操作文は、カーソルを使用せずにデータを操作します。

8.5.1 SELECT文

SELECT文は、表の指定された行から値を取り出します。

【書き方】

[<FOR句 >]
SELECT [ALL|DISTINCT] <選択リスト>
INTO <選択相手リスト>

<表式 >

<選択相手リスト> ::=

<相手指定> [{,<相手指定>} …]

参照

"8.3.12 問合せ指定"の<選択リスト>

"8.3.11 表式"

"8.3.5 値指定と相手指定"の<相手指定>

構文規則

1. 選択リストの次数と、選択相手リストに指定されている相手指定の数は一致していなければなりません。
2. 選択リストの列のデータ型と相手指定のデータ型は、代入可能でなければなりません。
3. FOR句が指定された場合、選択相手リストに指定する相手指定はすべて複数行指定ホスト変数または表指定ホスト変数でなければなりません。

一般規則

1. SELECT文の検索結果は、選択相手リストに指定されているホスト変数に代入されます。代入は、選択リストの左端の列と選択相手リストの先頭に指定されている相手指定から順番に実行されます。
2. 選択相手リストに複数行指定ホスト変数を指定した場合、代入は、検索結果の先頭行と相手指定の配列先頭要素から順番に実行されます。
3. 選択相手リストに表指定ホスト変数を指定した場合、代入は、選択リストの左端の列から順に表指定ホスト変数の従属項目を対応付け、さらに検索結果の先頭行と相手指定の配列先頭要素から順番に実行されます。

8.5.2 DELETE文(探索)

DELETE文(探索)は、探索条件を満たす表の行を削除します。

【書き方】

[<FOR句>]

DELETE FROM <表名> [WHERE <探索条件>]

参照

"表名"

"8.3.10 探索条件"

構文規則

1. 表名で指定する表は、更新可能でなければなりません。
2. FOR句が指定された場合、探索条件中のホスト変数はすべて複数行指定ホスト変数でなければなりません。

一般規則

1. 探索条件を指定した場合、探索条件が真となる行を削除します。
2. 探索条件を省略した場合、表のすべてを削除します。
3. 探索条件は、表の行を削除する前に評価されます。
4. 複数行指定ホスト変数を指定した場合、複数行指定ホスト変数の繰返し回数と同じ数だけSQL文が実行されます。FOR句が指定された場合は、FOR句に指定された値と同じ数だけSQL文が実行されます。実行には、配列要素の先頭の値から順に使用されます。

8.5.3 INSERT文

INSERT文は、既存の表に新しい列を挿入します。

【書き方】

[<FOR句>]

INSERT INTO <表名> [(<挿入列リスト>)]

{VALUES (<挿入値リスト>) | <挿入値リスト> }

<挿入値リスト> ::= <列名> [{,<列名>} …]

<挿入値リスト> ::= <挿入値> [{,<挿入値>} …]

<挿入値> ::= <値指定> | <動的パラメタ指定> | NULL

参照

“8.3.12 問合せ指定”

“表名”

“列名”

“8.3.10 探索条件”

“8.3.5 値指定と相手指定”の<値指定>

“8.3.5 値指定と相手指定”の<動的パラメタ指定>

構文規則

1. 挿入値リストに指定される列名は、表名で示される表の列を指定しなければなりません。また、列名が重複してはなりません。
2. 挿入列リストが省略された場合、表のすべての列に値が設定されます。
3. 挿入列リストが指定された場合、挿入列リストの列数と問合せ指定の次数とは等しくなければなりません。挿入列リストが省略される場合、挿入対象表の次数と問合せ指定の次数は等しくなければなりません。
4. 問合せ指定で示される各列は、挿入する表の各列にそれぞれ代入可能でなければなりません。
5. FOR句が指定された場合、挿入値リストまたは問合わせ指定のホスト変数はすべて複数行指定ホスト変数または表指定ホスト変数でなければなりません。

一般規則

1. 問合せ指定の結果が空であってはなりません。

2. 複数行指定ホスト変数または表指定ホスト変数を指定した場合、複数行指定ホスト変数または表指定ホスト変数の繰返し回数と同じ数だけSQL文が実行されます。FOR句が指定された場合は、FOR句に指定された値と同じ数だけSQL文が実行されます。実行には、配列要素の先頭の値から順に使用されます。

8.5.4 UPDATE文（探索）

UPDATE文(探索)は、表内の特定の行について、探索条件で指定した列を更新します。

【書き方】

[<FOR句>]
UPDATE <表名>
　　SET <設定句:探索> [{,<認定句:探索>} …]
　　[WHERE <探索条件>]

<設定句:探索> ::= <対象列:探索> = {<値式> | NULL}
<認定句:探索> ::= <列名>

参照

“表名”
“8.3.10 探索条件”
“8.3.8 値式”
“列名”

構文規則

1. 表名で指定する表は、更新可能でなければなりません。
2. 設定句で指定する値式に集合関数指定を含んではなりません。
3. 設定句に指定する列名は、更新対象表の列名でなければなりません。
4. 更新対象の列として指定する列名は、更新対象表の列名でなければなりません。
5. 設定句の“＝”の右側の値式は、“＝”の左側の列名で示される列に対して代入可能でなければなりません。
6. FOR句が指定された場合、設定句および探索条件中のホスト変数はすべて複数行指定ホスト変数でなければなりません。

一般規則

1. WHERE句を指定した場合、WHERE句の探索条件が真になる行が変更されます。
2. WHERE句を省略した場合、表のすべての行が変更されます。
3. 設定句は、列の値を“＝”の右側に指定されている値に変更します。
4. 設定句の値式に含まれる列指定の値は、UPDATE文(探索)により変更される前の値です。
5. 複数行指定ホスト変数を指定した場合、複数行指定ホスト変数の繰返し回数と同じ数だけSQL文が実行されます。FOR句が指定された場合は、FOR句に指定した値と同じ数だけSQL文が実行されます。実行には、配列要素の先頭の値から順に使用されます。

8.6 カーソル系データ操作文

カーソル系データ操作文は、カーソルを使用してデータを操作します。

8.6.1 カーソル宣言

カーソル宣言は、カーソルを定義します。

【書き方】

DECLARE <カーソル名> CURSOR FOR <カーソル指定>

<カーソル指定> ::=

{<問合せ式> [<ORDER BY句>]

| SELECT [ALL | DISTINCT] <選択リスト>

<FROM句> [<WHERE句>]

[FOR UPDATE] }

<ORDER BY句> ::= ORDER BY <分類指定> [{,<分類指定>} ….]

<分類指定> ::= {<符号なし整数> | <列指定>} [ASC | DESC]

参照

“カーソル名”

“8.3.13 問合せ式”

“8.3.12 問合せ指定”の<選択リスト>

“8.3.11.1 FROM句”

“8.3.11.2 WHERE句”

“8.3.6 列指定”

構文規則

1. カーソル名は、翻訳単位のプログラム内で一意でなければなりません。
2. ORDER BY句を指定した場合、導出表の行は、ORDER BY句で指定した順番になります。
3. ORDER BY句中の分類指定は、導出表の列を参照していなければなりません。分類指定の効果を次に示します。
   a. 分類指定が列指定を含むならば、分類指定は、その分類指定によって指定される名前を持つ導出表の列の列名を指定しなければなりません。
   b. 分類指定が符号なし整数を含むならば、分類指定は、符号なし整数によって参照される順序位置を持つ導出表の列を指定しなければなりません。符号なし整数は1以上で、かつ、導出表の次数を超えてはなりません。
   c. 列指定と符号なし整数の指定は、混在させることができます。

一般規則

1. カーソル宣言の記述は、そのカーソルを使用するSQL文よりも、ソースコード上の順番で前になければなりません。
2. 導出表は、問合せ式の結果の仮想的な表です。導出表は、カーソル宣言により1度定義されると、翻訳単位のプログラム内でつねに有効になります。

3. カーソル宣言直後のカーソルは、閉じられた状態です。
4. 導出表の列の列名、データ型、精度、位取りおよび長さは、すべてカーソル宣言に記述される問合せ式の値に等しくなります。
5. ORDER BY句を指定したときの効果は次のとおりです。
    a. ASCを指定すると、行の順序は昇順になります。
    b. DESCを指定すると、行の順序は降順になります。
    c. 分類指定間の分類優先順位は、指定した順になります。
    d. 分類対象となる列データが同値である行の順序は、ORDER BY句を指定していない場合の行の順序規則と同じです。
6. 読込み専用となった導出表に対して、DELETE文(位置付け)とUPDATE文(位置付け)は、発行できません。

8.6.2 OPEN文

OPEN文は、カーソルを開いて、カーソルを有効な状態にします。

【書き方】

OPEN <カーソル名>

参照

“カーソル名”

一般規則

1. カーソルは、OPEN文により開かれた状態となります。
2. オープン対象のカーソルは、閉じられた状態になければなりません。
3. オープン直後のカーソルの現在行は、先頭行の直前です。

8.6.3 CLOSE文

CLOSE文は、カーソルを閉じて、カーソルを無効な状態にします。

【書き方】

CLOSE <カーソル名>

参照

“カーソル名”

一般規則

1. クローズ対象のカーソルは、開かれた状態でなければなりません。

2. 閉じたカーソルは、再びOPEN文によって開くまで使用できません。

8.6.4 FETCH文

[Win16][WinIPF][Linux]の場合、FETCH文は、カーソルを指定した行に位置付け、その行から値を取り出します。

[Win32][Winx64][.NET]の場合、FETCH文は、カーソルを指定した行に位置付け、値を取り出します。

【書き方】

[<FOR句>]

FETCH [NEXT | PRIOR | FIRST | LAST] <カーソル名> INTO <取り出し相手リスト>

<取り出し相手リスト> ::= <相手指定> [{,<相手指定>} …]

NEXT、PRIORは、[Win32][Winx64][.NET]固有の機能です。

FIRST、LASTは、[.NET]固有の機能です。

参照

“カーソル名”

“8.3.5 値指定と相手指定”の<相手指定>

構文規則

1. 導出表の次数と、取り出し相手リストに指定されているホスト変数の数は、一致していなければなりません。
2. 導出表の各列は、それぞれ、取り出し相手リストに指定されているホスト変数に代入可能でなければなりません。
3. FOR句が指定された場合、取り出し相手リストに指定する相手指定はすべて複数行指定ホスト変数または表指定ホスト変数でなければなりません。

一般規則

1. カーソルは、開かれた状態でなければなりません。
2. 導出表の行を読み込んだ結果は、導出表の左端の列と取り出し相手リストの先頭に指定されているホスト変数から順に代入されます。
3. 取り出し相手リストに複数行指定ホスト変数を指定した場合、導出表を読み込んだ結果は、導出表の先頭行と取り出し相手リストの配列先頭要素から順に代入されます。
4. 取り出し相手リストに表指定ホスト変数を指定した場合、導出表を読み込んだ結果は、導出表の左端の列から順に表指定ホスト変数の従属項目の並びに対応つけられ、さらに、導出表の先頭行と取り出し相手リストの配列先頭要素から順に代入されます。
5. [Win32][Winx64][.NET]NEXTまたはPRIORはカーソルの方向付けを指定します。カーソル方向付けを指定したときの効果値を以下に示します。
    a. NEXTを指定すると、位置付けられた行から後ろの行の値を取り出します。他のカーソル方向付けが指定されていない場合、NEXTが既定値となります。
    b. PRIORを指定すると、位置付けられた行から前の行の値を取り出します。
6. [.NET]FIRSTまたはLASTはカーソルの位置付けを指定します。カーソル位置付けを指定したときの効果値を以下に示します。
    a. FIRSTを指定すると、カーソルを導出表の先頭行に位置付け、値を取り出します。

b. LASTを指定すると、カーソルを導出表の最終行に位置付け、値を取り出します。

8.6.5 DELETE文(位置付け)

DELETE文(位置付け)は、カーソルによって指定した表の中の1行を削除します。

【書き方】

DELETE FROM <表名>
WHERE CURRENT OF <カーソル名>

参照

"表名"

"カーソル名"

構文規則

表名で指定する表名は、カーソルの導出元の表を示していなければなりません。

一般規則

DELETE文(位置付け)によって、カーソルの導出元の表の行が削除されます。

8.6.6 UPDATE文(位置付け)

UPDATE文(位置付け)は、カーソルによって指定された表内の行を更新します。

【書き方】

UPDATE <表名>
SET <設定句:位置付け> [{,<設定句:位置付け>} …]
WHERE CURRENT OF <カーソル名>

<設定句:位置付け> ::= <対象列:位置付け> = {<値式>|NULL}

<対象列:位置付け> ::= <列名>

参照

"表名"

"カーソル名"

"8.3.8 値式"

"列名"

構文規則

1. 指定する表名は、カーソルの導出元の表を示さなければなりません。
2. 設定句に対する規則を以下に示します。
    a. 設定句に指定する列名は、カーソルの導出元の表の列名でなければなりません。
    b. 設定句に指定する列名は、重複してはなりません。
    c. 設定句に指定する値式は、対応する列に対して代入可能でなければなりません。

一般規則

1. UPDATE文(位置付け)の結果、カーソルの導出元の表の行の値が更新されます。
2. 設定句は、列の値を“＝”の右側に指定されている値に変更します。
3. 値式に含まれる列指定の値は、UPDATE文(位置付け)で変更される前の値です。

8.7 動的SQL

ここでは、動的SQLについて説明します。

8.7.1 INTO句/USING句

INTO句とUSING句は、動的SQL文に対する出力変数と入力パラメタを記述します。

【書き方】

{INTO | USING} {<相手指定> [{,<相手指定>} …]

参照

“8.3.5 値指定と相手指定”の<相手指定>

8.7.2 PREPARE文

PREPARE文は、実行のためのSQL文を用意します。

【書き方】

PREPARE <SQL文識別子> FROM <SQL文変数>

<SQL文変数> ::= <ホスト変数>

参照

“SQL文識別子”

“8.3.5 値指定と相手指定”の<ホスト変数名>

構文規則

SQL文変数は、データ型が英数字項目の固定長文字型または可変長文字型に対応するホスト変数でなければなりません。

一般規則

1. SQL文変数に設定するSQL文には、EXEC SQL、END-EXEC、ホスト変数およびFOR句を含むことはできません。
2. SQL文変数に設定するSQL文に変数を指定する場合には、動的パラメタ指定で指定します。
3. SQL文変数にSQL文を設定して、PREPARE文を実行することにより、そのSQL文の実行の準備が完了します。実行の準備が完了した文を被準備文と呼びます。
4. SQL文変数に設定したSQL文が動的SELECT文の場合、SQL文識別子に関連付けたカーソルは、閉じられた状態でなければなりません。

8.7.3 EXECUTE文

EXECUTE文は、被準備文中の動的パラメタ指定に対して入力パラメタおよび出力変数を関連付け、その文を実行します。

【書き方1】

EXECUTE <SQL文識別子> [<INTO句>] [<USING句>]

【書き方2】

[<FOR句>]

EXECUTE <SQL文識別子> { <INTO句> | <USING句> }

参照

“SQL文識別子”

“8.7.1 INTO句/USING句”の<INTO句>

“8.7.1 INTO句/USING句”の<USING句>

構文規則

書き方2のINTO句またはUSING句には、複数行指定ホスト変数または表指定ホスト変数を指定しなければなりません。

一般規則

1. SQL文識別子に対応する被準備文は、同一翻訳単位内の同名のSQL文識別子に対するPREPARE文で、実行の準備を完了していなければなりません。
2. 書き方1のEXECUTE文で実行可能な被準備文には、以下の6種類があります。
   - SELECT文
   - DELETE文(探索)
   - DELETE文(位置付け)

- INSERT文
- UPDATE文(探索)
- UPDATE文(位置付け)

3. 書き方2のEXECUTE文で実行可能な被準備文には、以下の4種類があります。
   - SELECT文
   - DELETE文(探索)
   - INSERT文
   - UPDATE文(探索)
4. 被準備文中の動的パラメタ指定に値を渡す場合、USING句を指定しなければなりません。
5. USING句に記述する相手指定には、動的パラメタ指定に渡す値を設定しておかなければなりません。
6. 被準備文中の動的パラメタ指定から値を受け取る場合、INTO句を指定しなければなりません。
7. EXECUTE文は、USING句に指定された項目の値を、動的パラメタ指定の値に対応付けて被準備文を実行します。被準備文がSELECT文の場合、INTO句に指定したホスト変数に、実行結果の値を設定します。

8.7.4 EXECUTE IMMEDIATE文

EXECUTE IMMEDIATE文は、SQL文を動的に準備して実行します。

【書き方】

EXECUTE IMMEDIATE <SQL文変数>

参照

"8.7.2 PREPARE文"の<SQL文変数>

構文規則

SQL文変数は、データ型が英数字項目の固定長文字型または可変長文字型に対応するホスト変数でなければなりません。

一般規則

1. SQL文変数に設定できるSQL文には、以下の5種類があります。
   - DELETE文(探索)
   - DELETE文(位置付け)
   - INSERT文
   - UPDATE文(探索)
   - UPDATE文(位置付け)
2. SQL文変数に設定するSQL文には、EXEC SQL、END-EXEC、ホスト変数、動的パラメタ指定およびFOR句を含むことはできません。

8.7.5 動的SELECT文

動的SELECT文は、“8.3.13 問合せ式”および“8.3.14 副問合せ”と同様の書き方で記述します。

動的SELECT文には、動的パラメタ指定を含むことができます。

8.7.6 動的カーソル宣言

動的カーソル宣言は、PREPARE文によって用意された動的SELECT文に従って、カーソルを定義します。

【書き方】

DECLARE <カーソル名> CURSOR FOR <SQL文識別子>

参照

“カーソル名”

“SQL文識別子”

構文規則

1. カーソル名は、翻訳単位内で一意でなければなりません。
2. 動的カーソル宣言で指定するSQL文識別子は、同一翻訳単位内のPREPARE文で指定しなければなりません。

一般規則

1. 動的カーソル宣言に指定するSQL文識別子に対応付ける被準備文は、カーソル指定(動的SELECT文)でなければなりません。SQL文識別子に関連付けるカーソル指定には、動的パラメタ指定を含むことができます。
2. 動的カーソル宣言で定義したカーソルは、カーソル宣言で定義したカーソルと同様に使用することができます。

8.7.7 動的OPEN文

動的OPEN文は、被準備文であるカーソル指定(動的SELECT文)に入力パラメタを関連付け、動的カーソル宣言で宣言したカーソルを開きます。

【書き方】

OPEN <カーソル名> [<USING句>]

参照

“カーソル名”

“8.7.1 INTO句/USING句”の<USING句>

構文規則

カーソル宣言を動的カーソル宣言に、OPEN文を動的OPEN文に置き換えることによって、OPEN文の構文規則が、動的OPEN文の構文規則に適用されます。

一般規則

1. カーソル名に関連付けた被準備文は、カーソル指定でなければなりません。
2. カーソル名と関連付けたカーソル指定が動的パラメタ指定を含む場合、動的OPEN文にはUSING句を指定しなければなりません。
3. USING句の相手指定には、被準備文中の動的パラメタ指定に対応する値を設定します。
4. カーソル宣言を動的カーソル宣言に置き換えることによって、OPEN文の一般規則が、動的OPEN文の一般規則に適用されます。

8.7.8 動的CLOSE文

動的CLOSE文は、動的カーソル宣言で宣言されたカーソルを閉じます。

【書き方】

CLOSE <カーソル名>

参照

"カーソル名"

構文規則

カーソル宣言を動的カーソル宣言に、CLOSE文を動的CLOSE文に置き換えることによって、CLOSE文の構文規則が、動的CLOSE文の構文規則に適用されます。

一般規則

CLOSE文の一般規則が、動的CLOSE文の一般規則に適用されます。

8.7.9 動的FETCH文

動的FETCH文は、動的カーソル宣言で宣言したカーソルを次の行に位置付け、その行から値を取り出します。

【書き方】

[<FOR句>]

FETCH <カーソル名> <INTO句>

参照

"カーソル名"

"8.7.1 INTO句/USING句"の<INTO句>

構文規則

カーソル宣言を動的カーソル宣言に置き換えることによって、FETCH文の構文規則が、動的FETCH文の構文規則に適用されます。

一般規則

カーソル宣言を動的カーソル宣言に置き換えることによって、FETCH文の一般規則が、動的FETCH文の一般規則に適用されます。

8.7.10 動的DELETE文(位置付け)

動的DELETE文(位置付け)は、動的カーソル宣言で宣言したカーソルによって位置付けた1行を削除します。

【書き方】

```
DELETE FROM <表名>
        WHERE CURRENT OF <カーソル名>
```

参照

"表名"

"カーソル名"

構文規則

カーソル宣言を動的カーソル宣言に、DELETE文(位置付け)を動的DELETE文(位置付け)に置き換えることによって、DELETE文(位置付け)の構文規則が、動的DELETE文(位置付け)の構文規則に適用されます。

一般規則

DELETE文(位置付け)を動的DELETE文(位置付け)に置き換えることによって、DELETE文(位置付け)の一般規則が、動的DELETE文(位置付け)の一般規則に適用されます。

8.7.11 動的UPDATE文(位置付け)

動的UPDATE文(位置付け)は、動的カーソル宣言で宣言したカーソルで位置付けた1行を更新します。

【書き方】

```
UPDATE <表名>
        SET <設定句:位置付け> [{,<設定句:位置付け>} …]
        WHERE CURRENT OF <カーソル名>
```

参照

"表名"

"8.6.6 UPDATE文(位置付け)"の<設定句:位置付け>

“カーソル名”

構文規則

カーソル宣言を動的カーソル宣言に、UPDATE文(位置付け)を動的UPDATE文(位置付け)に置き換えることによって、UPDATE文(位置付け)の構文規則が、動的UPDATE文(位置付け)の構文規則に適用されます。

一般規則

UPDATE文(位置付け)を動的UPDATE文(位置付け)に置き換えることによって、UPDATE文(位置付け)の一般規則が、動的UPDATE文(位置付け)の一般規則に適用されます。

8.8 セション制御文

8.8.1 COMMIT文

現行のトランザクションを終了させます。

【書き方】

```
COMMIT [WORK]
```

一般規則

1. 現行のトランザクションを終了させます。
2. 現行のトランザクションによってなされた変更をデータベースに反映します。

8.8.2 ROLLBACK文

現行のトランザクションを終了させます。

【書き方】

```
ROLLBACK [WORK]
```

一般規則

1. 現行のトランザクションを終了させます。
2. 現行のトランザクションによってなされた変更を取り消します。

8.9 コネクション制御文

8.9.1 CONNECT文

SQL環境にSQLセションを確立します。

【書き方】

CONNECT TO <コネクション相手>

<コネクション相手> ::=

{<サーバ名> [AS <コネクション名>]

[USER <利用者名>]

| DEFAULT}

<サーバ名> ::= 一般規則参照

<コネクション名> ::= 一般規則参照

<利用者名> ::= 一般規則参照

一般規則

1. サーバ名、コネクション名、利用者名は、固定長文字列型のホスト変数または文字列定数を指定します。
2. サーバ名は、動作環境ファイルのサーバ情報に設定するサーバ名を指定します。
3. サーバ名の最大長は、32バイトです。ただし、長さに関する規則は、関連システムによって制限を受けることがあります。
4. コネクション名は、コネクションを識別する名前を指定します。
5. コネクション名の最大長は、18バイトです。ただし、長さに関する規則は、関連システムによって制限を受けることがあります。
6. コネクション名を省略した場合は、サーバ名がコネクション名になります。
7. 利用者名は、ユーザIDとパスワードを斜線“/”で区切って指定します。

   ユーザIDは、サーバに登録されているユーザIDを指定します。パスワードは、サーバに登録されているパスワードを指定します。
8. ユーザIDとパスワードの最大長は、それぞれ32バイトです。ただし、長さに関する規則は、関連システムによって制限を受けることがあります。
9. サーバ名、コネクション名、ユーザIDおよびパスワードの、前方および後方の空白は無視されます。
10. USERを省略した場合、動作環境ファイルのデフォルトコネクションの情報が有効となります。
11. DEFAULTを指定した場合、動作環境ファイルのデフォルトコネクションの情報が有効となります。
12. CONNECT文の実行により、サーバとコネクションを接続します。
13. 複数コネクションを確立した場合、最後に実行したCONNECT文のコネクションが現コネクションとなります。
14. コネクションの有効範囲内では、同じコネクション名を指定して、複数のコネクションを接続することはできません。また、DEFAULTを指定して、複数のコネクションを接続することはできません。

8.9.2 SET CONNECTION文

利用可能なコネクションのうち、1つのコネクションを選択します。

【書き方】

SET CONNECTION <コネクション対象>

<コネクション対象> ::= <コネクション名> | DEFAULT

<コネクション名> ::= 一般規則参照

一般規則

1. コネクション名は、固定長文字列型のホスト変数または文字列定数を指定します。
2. コネクション名の最大長は、18バイトです。ただし、長さに関する規則は、関連システムによって制限を受けることがあります。
3. コネクション名は、CONNECT文により接続したコネクション名を指定します。
4. DEFAULTは、先にDEFAULTを指定したCONNECT文が実行されている場合に、指定できます。

8.9.3 DISCONNECT文

利用者とSQL環境間のSQLコネクションを切断します。

【書き方】

DISCONNECT <切断対象>

<切断対象> ::= <コネクション名> | DEFAULT | ALL | CURRENT

<コネクション名> ::= 一般規則参照

一般規則

1. コネクション名は、固定長文字列型のホスト変数または文字列定数を指定します。
2. コネクション名の最大長は、18バイトです。ただし、長さに関する規則は、関連システムによって制限を受けることがあります。
3. コネクション名を指定した場合は、CONNECT文により接続したコネクション名が切断の対象になります。
4. DEFAULTを指定した場合は、デフォルトコネクションが切断の対象になります。
5. ALLを指定した場合は、利用可能なすべてのコネクションが切断の対象になります。
6. CURRENTを指定した場合は、現コネクションが切断の対象になります。
7. DISCONNECT文を実行する場合は、トランザクションは終了した状態でなければなりません。
8. DISCONNECT文で指定したコネクションが現コネクションの場合は、現コネクションがない状態になります。

8.10 ストアドプロシージャ

8.10.1 CALL文

サーバに登録されているストアドプロシージャの呼出しを行います。

この機能は、[Win32][Linux][.NET][WinIPF][Winx64]固有の機能です。

【書き方】

CALL <ストアドプロシージャ名 >([<引数リスト>])

<引数リスト> ::= <引数>[{,<引数>} …]

<引数> ::= <値指定>

参照

“ストアドプロシージャ名”

“8.3.5 値指定と相手指定”の<値指定>

一般規則

1. 指定する引数の数は、ストアドプロシージャ定義で指定したパラメタ宣言の数に一致していなければなりません。
2. 指定する引数の順序は、ストアドプロシージャ定義の対応するパラメタの順序に一致していなければなりません。
3. [Win32][WinIPF][Winx64][.NET]では、ストアドプロシージャ定義の戻り値は、SQLERRDの1番目の要素であるSQLERRD(1)で受け取ることができます。
4. 引数に指定する定数は、ストアドプロシージャ定義の入力パラメタに対してだけ指定可能です。

第9章 通信データベース

通信データベース機能は、クライアントサーバ方式で、通信データベースを介してデータの送受信を行う機能です。

クライアントサーバ方式とは、ある処理を行う場合に、処理の要求と処理の実行とを分離して実現する方式です。処理を要求する側をクライアント、処理を実行する側をサーバと呼びます。ここでは特に、データ処理を実行するためにサーバで動作する応用プログラムをサーバアプリケーションといいます。

クライアントはサーバアプリケーションに未処理のデータを送って処理を要求し、サーバアプリケーションは処理結果をクライアントに返します。この通信は、サーバアプリケーションとクライアントが直接通信するのではなく、それぞれが通信データベースに対してデータの送受信を行います。

クライアントからデータを送る場合、まず、クライアントは通信データベースをあて先としてデータの送信命令を出します。サーバアプリケーションは、通信データベースをあて先としてデータの受信命令を出し、クライアントから送信されたデータを受信します。

また、サーバアプリケーションからデータを送る場合、サーバアプリケーションは、通信データベースをあて先としてデータの送信命令を出します。クライアントは、通信データベースをあて先としてデータの受信命令を出し、サーバアプリケーションから送信されたデータを受信します。

利用者は、通信データベース機能を用いることで、実際の通信先であるサーバアプリケーションあるいはクライアントの所在や内部処理を意識することなく、通信を行うことができます。

通信データベース機能は[Win16]固有の機能です。また、この機能を使用するには、サーバ側とクライアント側のPowerAIMの製品および通信用ソフトウェアが必要となります。

通信データベース

通信データベースは仮想データベースです。SQLを使用してアクセスすることから、データベースに見立てられています。利用者は、通信データベースをあて先としてデータの送信または受信を行います。

通信データベースは、サーバアプリケーションとクライアントの連携のために必要な情報として、サービスとテーブルを持ちます。

サービス

サービスは、利用者が使用できるテーブルについての情報です。サービスは利用者に公開されます。

クライアント側があるサービスを指定すると、対応するサーバアプリケーションが起動され、これと通信できます。

サービスとサーバアプリケーションは1対1に対応付けられています。

テーブル

テーブルは、通信データの属性を定義したものです。テーブルは利用者に公開されます。利用者は、各テーブルの定義に沿ったデータを送信または受信します。

テーブルには、入力側と出力側があります。入力側は、サーバアプリケーションがクライアントから受け取るデータについての属性定義です。出力側は、サーバアプリケーションがクライアントに送るデータについての属性定義です。

埋込みDCSQL

埋込みDCSQLは応用プログラムから通信データベースを使用して通信を行うための記述です。DCSQLとは、通信データベースのアクセスのために使用される、一部のSQLを指します。

COBOLプログラムでは、埋込みDCSQLの記述は“EXEC DCSQL”と“END-EXEC”で囲んで記述します。

ホスト変数

サーバアプリケーションとクライアント間で、データを受け渡すために使用するデータ項目を特にホスト変数と呼びます。

ホスト変数の詳細については“9.1.6 ホスト変数名”および“9.3.1.2 ホスト変数定義”を参照してください。

9.1 埋込みDCSQLの基本要素

COBOLプログラム中に記述する埋込みDCSQLの基本要素について説明します。

9.1.1 使用できる文字

埋込みDCSQLで使用可能な文字には、以下の6種類があります。

数字

0、1、2、3、4、5、6、7、8、9の10文字

英字

英字には英大文字と英小文字があります。

英大文字

A、B、C、…、Zの26文字

英小文字

a、b、c、…、zの26文字

英数字

英数字は、英字と数字の文字集合です。

特殊文字

・、＜、(、＋、＊、)、;、－、/、,、%、_、＞、?、:、”の16文字

拡張文字

@、¥、#の3文字

各国語文字

各国語文字として日本語文字が使用できます。

9.1.2 引用符、キーワードおよび分離符号

埋込みDCSQLでは、引用符はアポストロフィー(’)を使用します。文字列定数中に引用符を書きたい場合は、引用符を2つ続けて指定します。これは文字列定数の中で単一のアポストロフィーとみなされます。

埋込みDCSQL文中のキーワードおよび分離符号の規則は、埋込みSQLでの規則がそのまま適用されます。

9.1.3 通信データベース名

通信データベース名は、通信データベースにつけた名前です。通信データベース名は以下でなければなりません。

- 18文字以下の英数字項目
- 8文字以下の日本語項目

9.1.4 サービス名

サービス名は、サービスにつけた名前です。サービス名は以下でなければなりません。

- 18文字以下の英数字項目
- 18文字以下の日本語項目

9.1.5 テーブル名

テーブル名は、テーブルにつけた名前です。テーブル名は以下でなければなりません。

- 18文字以下の英数字項目
- 18文字以下の日本語項目

9.1.6 ホスト変数名

ホスト変数名は、ホスト変数につけた名前です。ホスト変数は、手続き部の埋込みDCSQL文およびCOBOLプログラム中で、一意名やデータ名を指定してよいところならどこでも指定することができます。

ホスト変数名は、18文字以下の英数字または日本語の並びでなければなりません。その構成文字はCOBOLの利用者語の規則に従います。

9.1.7 定数

埋込みDCSQLで記述できる定数には以下のものがあります。

- 文字列定数
- 漢字列定数
- 真数定数
- 概数定数

9.1.7.1 文字列定数

文字列定数は、その始まりと終わりを引用符で区切られた文字列であり、文字のデータを指定します。

【書き方】

‘{文字-1} …’

構文規則

1. 文字列定数中に引用符を持ちたい場合は、引用符を2つ続けて指定します。これは文字列定数の中で、単一のアポストロフィーと等価に扱われます。
2. 文字列定数のデータ型は文字列です。
3. 文字列定数の長さは、データ型が文字列型であるホスト変数の長さの規則に従います。

一般規則

文字列定数の値は、それが含む文字の並びです。

9.1.7.2 漢字列定数

漢字列定数は、その始まりを“N”と引用符の並びで、終わりを引用符で区切られた日本語文字列であり、日本語のデータを指定します。

【書き方】

N'{日本語文字-1} …'

構文規則

1. 漢字列定数のデータ型は漢字列です。漢字列定数の長さは、それが含む日本語文字の個数です。
2. 漢字列定数の長さは、データ型が漢字列型であるホスト変数の長さの規則に従います。

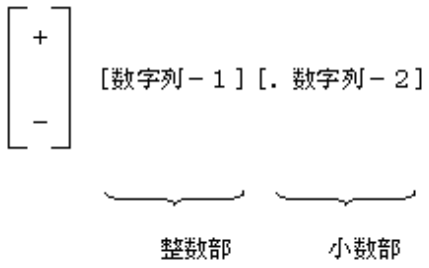
一般規則

漢字列定数の値は、それが含む日本語文字の並びです。

9.1.7.3 真数定数

真数定数は、0～9の数字および小数点からなる文字列です。真数定数には正号または負号を指定することができます。

【書き方】



構文規則

1. 真数定数のデータ型は真数です。真数定数の精度は、それが含む数字の個数です。真数定数の位取りは、小数点の右側の数字の個数です。
2. 真数定数の精度および位取りは、データ型が真数型であるホスト変数の精度および位取りの規則に従います。

一般規則

1. 真数定数は、固定小数点の数値データを表します。
2. 真数定数の数値は、符号付き位取り10進表記法の通常の数学的解釈から導出されます。

9.1.7.4 概数定数

概数定数は、0～9の数字、小数点および文字“E”からなる文字列です。概数定数には正号または負号を指定することができます。

【書き方】

[+|-]仮数E指数

構文規則

1. 概数定数のデータ型は概数です。概数定数の精度は、その仮数の精度です。
2. 仮数は、真数定数と同じ形式で指定します。
3. 指数は次の形式で指定します。

[＋｜－]｛数字｝…

4. 概数定数の精度および位取りは、データ型が概数型であるホスト変数の精度および位取りの規則に従います。

一般規則

1. 概数は、浮動小数点の数値データを表します。
2. 概数定数の数値は、仮数によって表される真数値と、指数によって表される数の10のべき乗によって得られる数の積です。

9.2 埋込みDCSQLの正書法

ここでは、COBOLプログラム中に記述する埋込みDCSQLの正書法について説明します。

9.2.1 記述の全般規定

1. 埋込みDCSQL開始宣言、埋込みDCSQL終了宣言および埋込みDCSQL文は、“EXEC DCSQL”と“END-EXEC”で囲んで記述しなければなりません。
2. 埋込みDCSQL開始宣言、埋込みDCSQL終了宣言および埋込みDCSQL文は、すべてB領域に記述しなければなりません。

9.2.2 行のつなぎ

埋込みDCSQL開始宣言、埋込みDCSQL終了宣言および埋込みDCSQL文の継続(行のつなぎ)の規則は、COBOLの正書法の規則に従います。

9.2.3 COBOLの注記行および行内注記

COBOLの注記行、デバッグ行および空白行は、埋込みDCSQLの記述中のどこにでも書くことができます。

COBOLの行内注記は、埋込みDCSQLの記述中には記述できません。

9.2.4 埋込みDCSQLの注釈

埋込みDCSQLの注釈は連続した2文字“－－”で始まり、境界Rのすぐ左の文字位置で終わる行の部分をいいます。埋込みDCSQLの注釈は、埋込みDCSQL文中のA領域およびB領域のどこから書き始めてもかまいませんが、直前は空白またはコンマ(,)でなければなりません。また、埋込みDCSQLの注釈自体を継続することはできません。埋込みDCSQLの注釈の直前の語、定数またはPICTURE句の文字列は、標識領域にハイフンを記述して継続することができます。

9.3 埋込みDCSQL

埋込みDCSQLは通信データベースを使用して通信を行うための記述です。ここでは、埋込みDCSQL宣言節、埋込みDCSQL文の形式および規則について説明します。

9.3.1 データ部

データ部では、埋込みDCSQL文で使用する変数を定義します。

9.3.1.1 埋込みDCSQL宣言節

埋込みDCSQL宣言節では以下を宣言します。

- サーバアプリケーションとクライアント間の通信で使用するホスト変数
- システムから通知される処理結果の状態を受け取るために使用する変数
- システムから通知されるメッセージを受け取るために使用する変数

【書き方】

EXEC DCSQL

BEGIN DECLARE SECTION END-EXEC.

[{ホスト変数定義} …]

EXEC DCSQL

END DECLARE SECTION END-EXEC.

一般規則

1. 埋込みDCSQL宣言節は、データ部の作業場所節または連絡節に記述することができます。
2. 埋込みDCSQL開始宣言(EXEC DCSQL BEGIN DECLARE SECTION END-EXEC)と埋込みDCSQL終了宣言(EXEC DCSQL END DECLARE SECTION END-EXEC)は省略できません。
3. 埋込みDCSQL開始宣言と埋込みDCSQL終了宣言は、対で指定しなければなりません。また、入れ子であってはなりません。
4. 埋込みDCSQL宣言節は、埋込みSQL宣言節内に含まれてはなりません。また、埋込みDCSQL宣言節内に、埋込みSQL宣言節を含んではなりません。

9.3.1.2 ホスト変数定義

ホスト変数定義は、ホスト変数の性質を指定します。

ホスト変数として浮動小数点項目を定義する場合、USAGE句にCOMP-1またはCOMP-2を指定します。

【書き方】

レベル番号 ホスト変数名

[IS EXTERNAL]

[IS GLOBAL]

[{ PICTURE / PIC } IS 文字列]

[USAGE IS] { BINARY / COMPUTATIONAL / COMP / COMP-1 / COMP-2 / DISPLAY / PACKED-DECIMAL }

[[SIGN IS]

LEADING SEPARATE CHARACTER]

[VALUE IS 定数-1].

構文規則

1. ホスト変数は、レベル番号01またはレベル番号77の基本項目として宣言しなければなりません。

2. データ型CHARACTER VARYINGまたはNATIONAL CHARACTER VARYINGとしてテーブルに定義される項目に対応するホスト変数は、レベル番号01の集団項目として宣言しなければなりません。この場合、レベル番号01の項目に直接従属する項目は、レベル番号49の基本項目でなければなりません。
3. 各句の構文規則は、COBOLのデータ記述項における各句の構文規則に従います。

一般規則

1. PICTURE句は基本項目以外のレベルに書いてはなりません。
2. 各句の一般規則は、COBOLのデータ記述項における各句の一般規則に従います。
3. ホスト変数を指定する場合、以下のいずれかの基本項目でなければなりません。
   a. 4桁または9桁の符号付き2進整数項目
   b. 18桁以下の符号付き内部10進項目
   c. 18桁以下の符号付き外部10進項目(SIGN句指定必要)
   d. 英数字項目
   e. 日本語項目
   f. 単精度内部浮動小数点項目
   g. 倍精度内部浮動小数点項目
4. データ型CHARACTER VARYINGとしてテーブルに定義される項目に対応するホスト変数は、以下でなければなりません。

```
01    ホスト変数名-1.
  49  ホスト変数名-2  PIC S9(9) BINARY.
  49  ホスト変数名-3  PIC X(n).
```

5. データ型NATIONAL CHARACTER VARYINGとしてテーブルに定義される項目に対応するホスト変数は、以下でなければなりません。

```
01    ホスト変数名-1.
  49  ホスト変数名-2  PIC S9(9) BINARY.
  49  ホスト変数名-3  PIC N(n).
```

6. ホスト変数は、無名項目であってはなりません。

9.3.1.3 ホスト変数の参照

ホスト変数は、埋込みDCSQL文の中でも、一般のCOBOL文の中でも参照することができます。埋込みDCSQL文の中でホスト変数を参照する場合、コロン(:)をホスト変数名の前に付加しなければなりません。すなわち、“A”という名前のホスト変数は、埋込みDCSQL文中では“:A”と記述します。一般のCOBOL文中でホスト変数を参照する場合は、コロンを付加してはなりません。

9.3.1.4 DCSQLSTATE

DCSQL文を実行すると、実行の成功または失敗の情報が、コード化されてDCSQLSTATEに設定されます。応用プログラムは、DCSQLSTATEを参照することにより、DCSQL文の実行の状況を知り、処理を組み立てることができます。

DCSQL文を実行すると、サーバシステムは、DCSQLSTATEの値を設定します。

DCSQLSTATEは、変数名が“DCSQLSTATE”である長さ5の英数字項目として、埋込みDCSQL宣言節で必ず定義しなければなりません。

DCSQLSTATEの値については、“PowerAIM/CL ソフトウェア開発キット for Windows オンラインマニュアル”を参照してください。

9.3.1.5 DCSQLMSG

DCSQL文を実行すると、実行の成功または失敗を通知するメッセージがDCSQLMSGに格納されます。利用者は、出力文を利用して、DCSQLMSGに格納されたメッセージを表示または印字することができます。

DCSQLMSGを使用する場合は、変数名が“DCSQLMSG”である英数字項目を埋込みDCSQL宣言節で定義しなければなりません。DCSQLMSGの最大長は255バイトです。

9.3.2 手続き部

手続き部では、データベースに対する接続/切断、トランザクションの管理および、データの送受信を行う文を定義します。

記述形式および規則については、“PowerAIM/CL ソフトウェア開発キット for Windows オンラインマニュアル”を参照してください。

第10章　Micro Focus固有機能

本章では、Micro Focus固有機能の機能や書き方について説明しています。

10.1 名前付き定数

ここでは、Micro Focus固有機能の名前付き定数について説明します。

注意

この機能は、[Win32][WinIPF][Winx64][Solaris][Linux][LinuxIPF][Linux64][.NET]固有の機能です。

10.1.1 利用者語

利用者語には、“1.2.2.1 利用者語”の他に名前付き定数があります。

名前付き定数

名前付き定数は、定数に付ける名前です。定数を名前で参照するために使います。名前付き定数は、データ部にレベル番号78で定義します。名前付き定数は、PICTURE句の反復回数を表す整数および以下を除く【書き方】で“定数-n”、“整数-n”と示しているところに書くことができます。

- 報告書作成機能固有の報告書記述項および報告書集団記述項
- 連結式
- COPY文
- ENTRY文

10.1.2 名前の範囲

名前付き定数は、定義したプログラムのデータ部の利用者語が定義された位置よりも物理的にうしろの記述項、そのプログラムの手続き部中の文、そのプログラムに含まれるプログラム中の記述項および文で参照することができます。

ただし、同じ翻訳単位中の他の利用者語と同じ名前であってはなりません。

10.1.3 データ記述項

ここでは、データ記述項について説明します。

【書き方】

78 名前付き定数-1 VALUE IS { 定数-1 / LENGTH OF 一意名-1 } .

構文規則

1. 名前付き定数は、以下にに書くことができます。
   - ファイル節
   - 作業場所節
   - 画面節

- 定数節
- 基底場所節
- 連絡節

2. データ記述項の先頭には、レベル番号78を書かなければなりません。
3. 定数-1は、記号文字または記号定数であってはなりません。
4. 一意名-1は、添字または部分参照子を付けてはなりません。
5. 一意名-1に、内部ブール項目を指定してはなりません。

一般規則

1. 名前付き定数-1には、VALUE句に指定された定数に付ける名前を指定します。
2. 名前付き定数-1の性質は、定数-1によって決まります。
3. LENGTH指定は、一意名-1に割り当てられる記憶域の長さが指定されたとみなされます。
4. 一意名-1は、このデータ記述項より物理的に前で現れ、かつ完結しているデータ記述項で定義されていなければなりません。
5. 一意名-1に、可変反復データ項目が指定されている場合、OCCURS句の中のDEPENDING ON指定のデータ項目の値に、そのOCCURS句の繰返しの最大数が設定されているかのように扱われます。

10.2 16進数字定数

16進数字定数は、16進文字を数字として扱う定数です。

この機能は、[Win32][WinIPF][Winx64][Solaris][Linux][LinuxIPF][Linux64][.NET]固有の機能です。

【書き方】

H"16進文字-1 …"

---

1. 16進文字-1の並びは、左端を分離符の“H””で区切り、右端を分離符の引用符で区切らなければなりません。
2. 16進文字-1は、“0”～“9”または“A”～“F”でなければなりません。
3. 16進文字-1の文字数は2～16個の偶数個でなければなりません。
4. 16進文字-1の並びは、ビッグエンディアンで表現します。
5. 16進数字定数の項類は、数字です。
6. 16進数字定数は、【書き方】で“定数-n”と示しているところで、数字定数が許されるところに書くことができます。
7. 16進数字定数は、負数および小数を扱えません。
8. 16進数字定数の桁よせは標準桁よせ規則に従って行います。

10.3 スクリーン操作機能

ここでは、Micro Focus固有機能のスクリーン操作機能について説明します。

注意

[WinIPF][Linux][LinuxIPF][Linux64][.NET]では、スクリーン操作機能は使用できません。

10.3.1 環境部

ここでは、特殊名段落について説明します。

10.3.1.1 特殊名段落

特殊名段落では、記号定数、記号文字、符号系名、文字集合、文字の大小順序などを定義したり、機能名と呼び名を対応付けたりします。

スクリーン操作機能としてCONSOLE IS CRT句が指定できます。

【書き方】

SPECIAL-NAMES.

[CONSOLE IS CRT句]

一般規則

特殊名段落のすべての句は、それを明にまたは暗に指定したプログラム、およびその内部プログラムに適用されます。

10.3.1.1.1 CONSOLE IS CRT句

暗に指定された入出力先を定義します。

【書き方】

CONSOLE IS CRT

CONSOLE IS CRT句は、FROM句が指定されていないACCEPT文および、UPON句が指定されていないDISPLAY文の作用対象を、画面項目とみなします。

10.3.2 データ部

ここでは、画面データ記述項について説明します。

10.3.2.1 画面データ記述項

【書き方1】集団画面項目を定義する

レベル番号 [データ名-1 | FILLER]

[AUTO 句]

[BACKGROUND-COLOR 句]

[BLANK SCREEN 句]

[BLINK 句]

[CONTROL 句]

[FOREGROUND-COLOR 句]

[FULL 句]
[GRID 句]
[HIGHLIGHT 句]
[LOWLIGHT 句]
[LEFTLINE 句]
[OCCURS 句]
[OVERLINE 句]
[PROMPT 句]
[REQUIRED 句]
[REVERSE-VIDEO 句]
[SECURE 句]
[SIGN 句]
[UNDERLINE 句]
[USAGE 句]
[ZERO-FILL 句].

【書き方2】定数項目を定義する

レベル番号 [データ名-1] [FILLER]
[BACKGROUND-COLOR 句]
[BELL 句]
[BLANK LINE 句]
[BLANK SCREEN 句]
[BLINK 句]
[COLUMN NUMBER 句]
[ERASE 句]
[FOREGROUND-COLOR 句]
[GRID 句]
[HIGHLIGHT 句]
[LOWLIGHT 句]
[LEFTLINE 句]
[LINE NUMBER 句]
[OCCURS 句]
[OVERLINE 句]
[REVERSE-VIDEO 句]
[SIZE 句]
[UNDERLINE 句]
VALUE 句.

【書き方3】入力項目、出力項目または更新項目を定義する

```
レベル番号 [ データ名-1 ]
           [ FILLER   ]
           [AUTO 句]
           [BACKGROUND-COLOR 句]
           [BELL 句]
           [ BLANK LINE 句  ]
           [ BLANK SCREEN 句 ]
           [BLANK WHEN ZERO 句]
           [BLINK 句]
           [COLUMN NUMBER 句]
           [ERASE 句]
           [FOREGROUND-COLOR 句]
           [FULL 句]
           [GRID 句]
           [ HIGHLIGHT 句 ]
           [ LOWLIGHT 句  ]
           [JUSTIFIED 句]
           [LEFTLINE 句]
           [LINE NUMBER 句]
           [OCCURS 句]
           [OVERLINE 句]
           [PICTURE 句]
           [PROMPT 句]
           [REQUIRED 句]
           [REVERSE-VIDEO 句]
           [SECURE 句]
           [SIGN 句]
           [SIZE 句]
           [UNDERLINE 句]
           [USAGE 句]
           [ZERO-FILL 句].
```

構文規則

構文規則については、“5.5 画面データ記述項”を参照してください。

一般規則

1. 書き方3の入力項目、出力項目または更新項目の区別は、PICTURE句で指定します。
2. 集団画面項目にSECURE句を指定した場合、SECURE句は集団画面項目に従属するすべての入力項目に適用されます。

3. 集団画面項目に以下の句を指定した場合、その句は集団画面項目に従属するすべての入力項目および更新項目に適用されます。

   ー AUTO句

   ー FULL句

   ー REQUIRED句

   ー PROMPT句

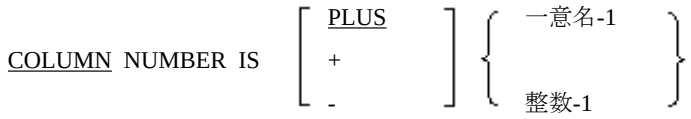
   ー ZERO-FILL句

4. LINE NUMBER句とCOLUMN NUMBER句において、画面上で領域が重なるような指定をすることはできません。また、物理画面の制限を超える指定をすることはできません。
5. JUSTIFIED句とSIZE句を同時に指定することはできません。
6. JUSTIFIED句とPROMPT句を同時に指定した場合、PROMPT句は無効になります。

10.3.2.1.1 COLUMN NUMBER句

画面項目を配置する列を指定します。相対列番号の書き方で“＋”と“ー”が指定できます。

【書き方】



構文規則

“＋”“ー”の書き方および規則は“PLUS”に準じます。構文規則、一般規則については、“5.5.8 COLUMN NUMBER句”を参照してください。

10.3.2.1.2 GRID句

画面項目を構成する各文字の左側に縦線を付加することを指定します。

【書き方】

GRID

構文規則

1. GRID句は、任意の画面項目に指定することができます。
2. GRID句が集団項目レベルに指定されるとき、それは集団項目に従属するすべての基本項目に適用されます。

一般規則

GRID句を指定すると、その画面項目が画面に表示されるとき、画面項目の各文字の左側に縦線が付加されます。

10.3.2.1.3 LEFTLINE句

画面項目の最左端の文字の左側に縦線を付加することを指定します。

【書き方】

LEFTLINE

構文規則

1. LEFTLINE句は、任意の画面項目に指定することができます。
2. LEFTLINE句が集団項目レベルに指定されるとき、それは集団項目に従属するすべての基本項目に適用されます。

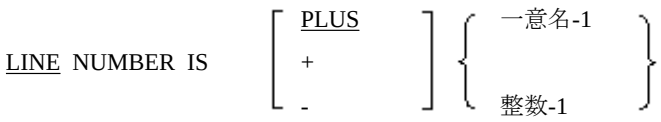
一般規則

LEFTLINE句を指定すると、その画面項目が画面に表示されるとき、最左端の文字の左側に縦線が付加されます。

10.3.2.1.4 LINE NUMBER句

画面項目を配置する行を指定します。相対行番号の書き方で“＋”と“－”が指定できます。

【書き方】



構文規則

“＋”“－”の書き方および規則は“PLUS”に準じます。構文規則、一般規則については、“5.5.14 LINE NUMBER句”を参照してください。

10.3.2.1.5 OCCURS句

同じデータ構造の繰り返しを定義します。

詳細については、“5.4.6 OCCURS句”を参照してください。

【書き方】

OCCURS 整数-1 TIMES

構文規則

OCCURS句は、任意の画面項目に指定することができます。

一般規則

1. OCCURS句を指定すると、従属する画面項目が整数-1回繰り返し定義されます。
2. OCCURS句が指定された集団項目に従属する基本項目に絶対位置が指定された場合、同じ画面位置への繰り返し指定となります。

10.3.2.1.6 OVERLINE句

画面項目の上端に横線を付加することを指示します。

【書き方】

OVERLINE

構文規則

1. OVERLINE句は、任意の画面項目に指定することができます。
2. OVERLINE句が集団項目レベルに指定されるとき、それは集団項目に従属するすべての基本項目に適用されます。

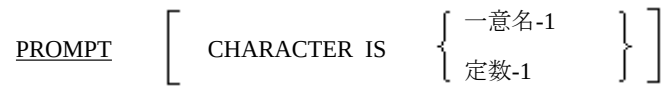
一般規則

OVERLINE句を指定すると、その画面項目が画面に表示されるとき、画面項目の各文字の上端に横線が付加されて表示されます。

10.3.2.1.7 PROMPT句

画面項目の空の文字位置に目印を付加することを指示します。

【書き方】



構文規則

1. PROMPT句は、入力項目および更新項目に書くことができます。
2. PROMPT句が集団項目レベルに指定されるとき、それは集団項目に従属する入力項目および更新項目に適用されます。
3. 一意名-1は、1文字の英字、英数字、日本語項目でなければなりません。
4. 定数-1は、1文字の文字定数、日本語定数および表意定数でなければなりません。

一般規則

1. PROMPT句を指定すると、入力項目および更新項目の空の文字を、指定した文字で埋めます。
2. PROMPT句によって表示された文字は、入力データとして有効にはなりません。
3. 一意名-1または定数-1がPICTURE句で指定した項類と異なる場合、動作は保証できません。
4. CHARACTERの指定がない場合、PROMPT句は意味を持ちません。
5. 一意名-1または定数-1に指定された文字が入力された場合、空白に置き換えられて格納されます。

10.3.2.1.8 SIZE句

画面項目の論理的な大きさを指定します。

【書き方】

SIZE IS { 一意名-1 / 整数-1 }

構文規則

1. SIZE句は、基本項目にだけ書くことができます。
2. SIZE句は、数字項目および数字編集項目に指定できません。
3. 一意名-1は、整数を記述するものでなければなりません。

一般規則

1. SIZE句で指定された値がゼロまたは負の整数の場合、SIZE句は無効になります。
2. SIZE句で指定された値が、関連するPICTURE句またはVALUE句が暗に示す値より小さい場合、画面項目の左側から指定された大きさだけが有効となります。
3. SIZE句で指定された値が、関連するPICTURE句またはVALUE句が暗に示す値より大きい場合、画面項目の右側には空白が表示されます。

10.3.2.1.9 ZERO-FILL句

未入力文字を空白の代わりにゼロで置き換えることを指示します。

【書き方】

ZERO-FILL

構文規則

1. ZERO-FILL句は、入力項目および更新項目に書くことができます。
2. ZERO-FILL句が集団項目レベルに指定されるとき、それは集団項目に従属する入力項目および更新項目に適用されます。

一般規則

ZERO-FILL句は、データを画面から受け取るときに、送出し側データ項目の大きさが画面項目の大きさより小さかった場合、右端の余りにゼロを補います。このとき、JUSTIFIED句が同時に指定された場合は、左側の余りにゼロが補われます。

10.3.3 手続き部

ここでは、以下の文について説明します。

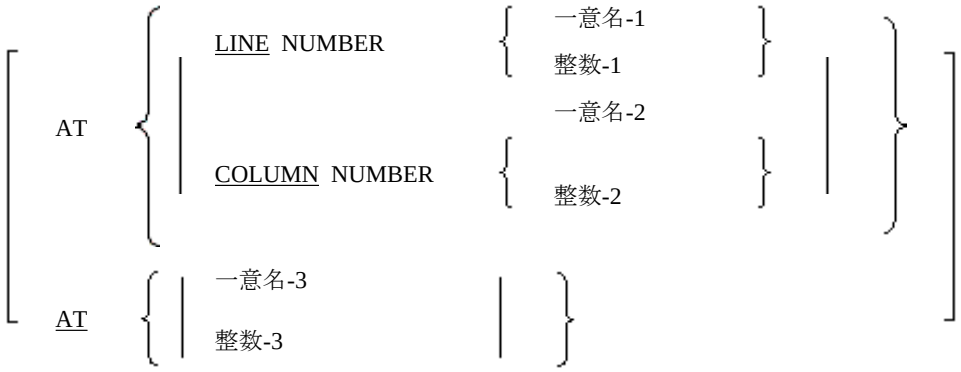
- ACCEPT文(スクリーン操作)
- DISPLAY文(スクリーン操作)

10.3.3.1 ACCEPT文（スクリーン操作）

画面からデータを入力します。

【書き方1】

ACCEPT データ名-1



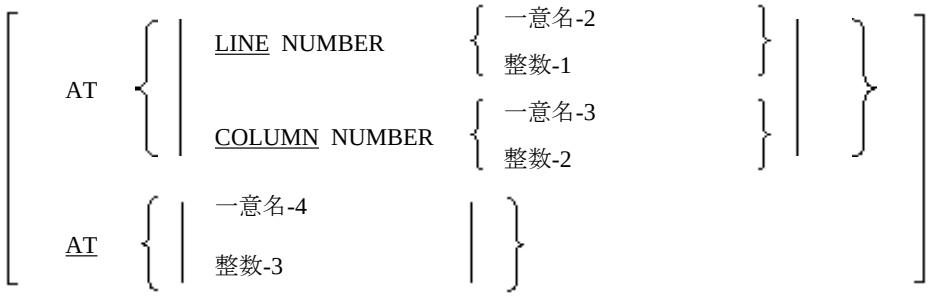
[ON EXCEPTION 無条件文-1]

[NOT ON EXCEPTION 無条件文-2]

[END-ACCEPT]

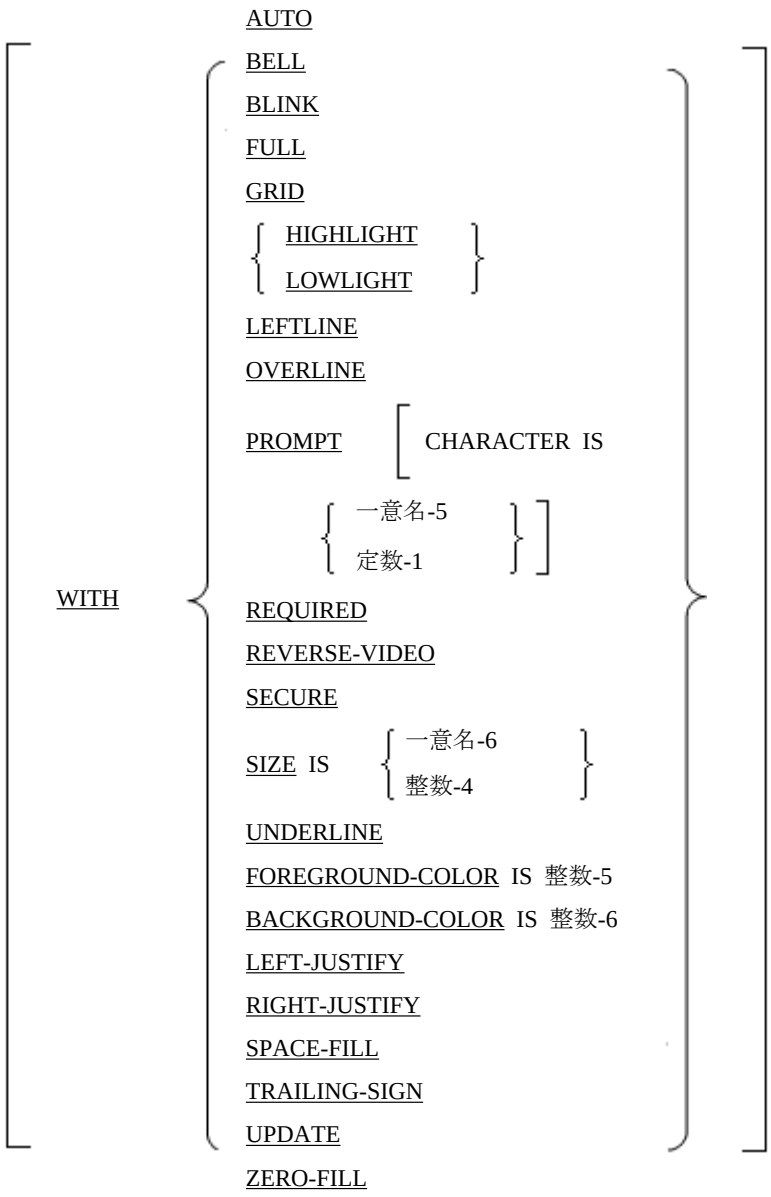
【書き方2】

ACCEPT 一意名-1



[FROM CRT]

[MODE IS BLOCK]



[ON EXCEPTION 無条件文-1]

[NOT ON EXCEPTION 無条件文-2]

[END-ACCEPT]

構文規則

書き方1の規則

1. 一意名-3は、4桁または6桁の符号なし外部10進項目でなければなりません。
2. 整数-3は、4桁または6桁でなければなりません。
3. その他の構文規則については“6.4.2 ACCEPT文（スクリーン操作）”を参照してください。

書き方2の規則

1. 一意名-1は、作業場所節で定義された項目でなければなりません。
2. 一意名-1は、固定長集団項目または基本項目でなければなりません。基本項目として、英字、英数字、外部10進、日本語項目が書けます。

3. 一意名-1の後の語句は任意の順序に指定することができます。
4. 整数-1、整数-2および整数-3は、符号なしでなければなりません。
5. 一意名-2および一意名-3は、符号なし整数項目でなければなりません。
6. 一意名-4は、4桁または6桁の符号なし外部10進項目でなければなりません。
7. 整数-3は、4桁または6桁でなければなりません。
8. SPACE-FILL、ZERO-FILL、LEFT-JUSTIFY、RIGHT-JUSTIFYおよびTRAILING-SIGNの各句は、基本項目にだけ指定することができます。

一般規則

書き方1の規則

一般規則については“6.4.2 ACCEPT文（スクリーン操作）”を参照してください。

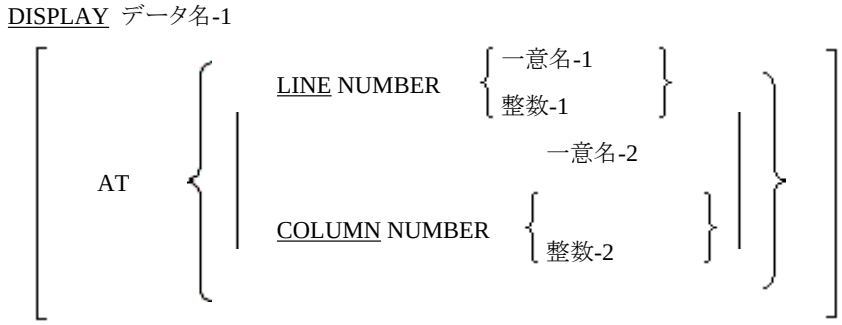
書き方2の規則

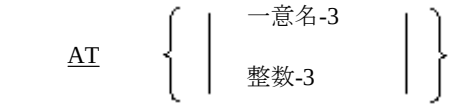
1. ACCEPT文は、一意名-1に指定された項目に対応する画面上の領域からデータを読み込みます。
2. ACCEPT文の実行は、入力キーを押すと完了します。
3. ATは、入力の対象となる項目の物理画面上の位置を指定します。
4. ATを指定しない場合、項目の入力操作は、第1行の第1列から始まります。
5. 一意名-4が4桁の場合、先頭の2桁に行番号を、残りの2桁に列番号を指定します。6桁の場合、先頭の3桁に行番号を、残りの3桁に列番号を指定します。
6. 整数-3が4桁の場合、先頭の2桁に行番号を、残りの2桁に列番号を指定します。6桁の場合、先頭の3桁に行番号を、残りの3桁に列番号を指定します。
7. FROM CRT指定は、ACCEPT文を書き方2で扱うよう指示します。
8. 一意名-1が可変長項目の場合、つねに最大長とみなします。
9. 一意名-1が集団項目で、MODE IS BLOCK句が指定された場合、ひとつの基本項目として扱います。
10. 一意名-1が集団項目で、MODE IS BLOCK句の指定がない場合、従属する基本項目が指定されたものとみなします。ただし、無名項目および表示用以外で定義された項目は無視されます。
11. LEFT-JUSTIFY、RIGHT-JUSTIFY、SPACE-FILL、TRAILING-SIGN、UPDATE指定は、注釈とみなします。
12. その他の句については、“5.5 画面データ記述項”または、“10.3.2.1 画面データ記述項”を参照してください。

10.3.3.2 DISPLAY文（スクリーン操作）

データを画面に表示します。

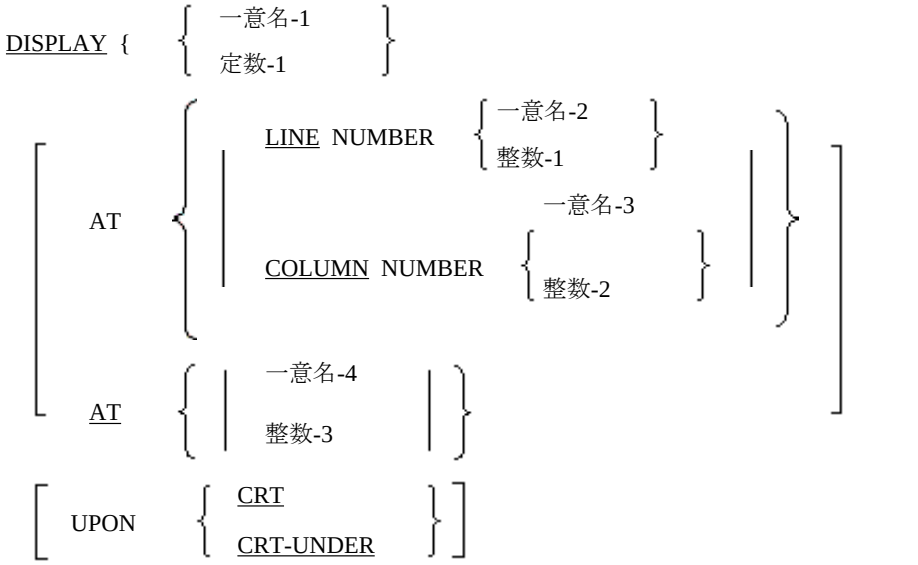
【書き方1】



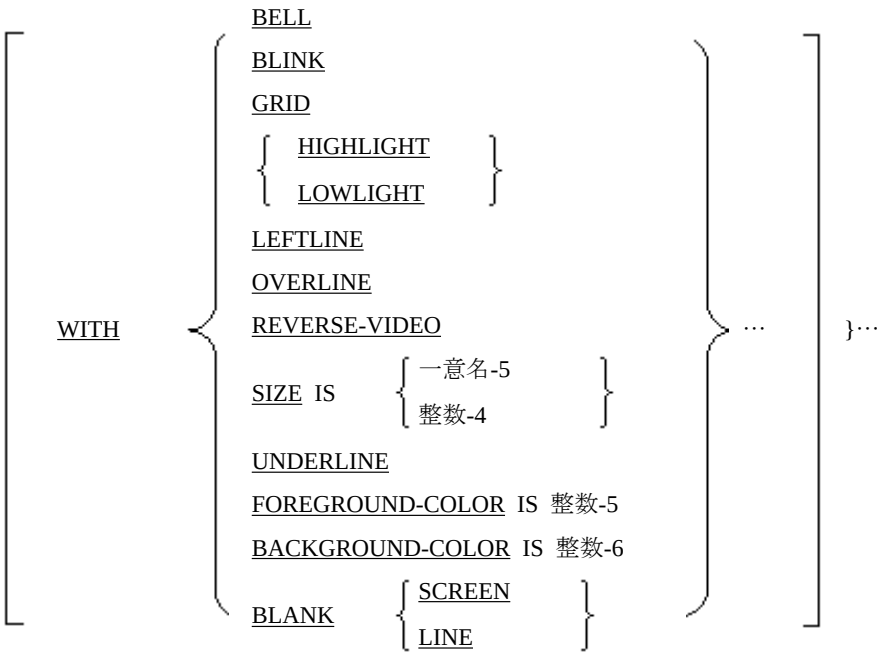


[END-DISPLAY]

【書き方2】



[MODE IS BLOCK]



[END-DISPLAY]

構文規則

書き方1の規則

1. 一意名-3は、4桁または6桁の符号なし外部10進項目でなければなりません。

2. 整数-3は、4桁または6桁でなければなりません。
3. その他の構文規則については“6.4.13 DISPLAY文（スクリーン操作）”を参照してください。

書き方2の規則

1. 一意名-1は、作業場所節で定義された項目でなければなりません。
2. 一意名-1は、固定長集団項目または、基本項目でなければなりません。基本項目として、英字、英数字、外部10進、日本語項目が書けます。
3. 一意名-1の後の語句は任意の順序に指定することができます。
4. 整数-1、整数-2および整数-3は、符号なしでなければなりません。
5. 一意名-2および一意名-3は、符号なし整数項目でなければなりません。
6. 一意名-4は、4桁または6桁の符号なし外部10進項目でなければなりません。
7. 整数-3は、4桁または6桁でなければなりません。

一般規則

書き方1の規則

一般規則については“6.4.13 DISPLAY文（スクリーン操作）”を参照してください。

書き方2の規則

1. DISPLAY文は、一意名-1に指定された項目に対応する画面上の領域にデータを表示します。
2. ATでは、表示の対象となる項目を含む画面の物理画面上の位置を指定します。
3. ATを指定しない場合、項目の表示操作は、第1行の第1列から始まります。
4. 一意名-4が4桁の場合、先頭の2桁に行番号を、残りの2桁に列番号を指定します。6桁の場合、先頭の3桁に行番号を、残りの3桁に列番号を指定します。
5. 整数-3が4桁の場合、先頭の2桁に行番号を、残りの2桁に列番号を指定します。6桁の場合、先頭の3桁に行番号を、残りの3桁に列番号を指定します。
6. UPON CRT指定は、DISPLAY文を書き方2で扱うよう指示します。
7. UPON CRT-UNDER指定は、DISPLAY文を書き方2で扱うよう指示すると同時に、下線を付加して表示します。
8. 一意名-1が可変長項目の場合、つねに最大長とみなします。
9. 一意名-1が集団項目で、MODE IS BLOCK句が指定された場合、ひとつの基本項目として表示します。
10. 一意名-1が集団項目で、MODE IS BLOCK句の指定がない場合、従属する基本項目が指定されたものとみなします。ただし、無名項目および表示用以外で定義された項目は無視されます。
11. その他の句については、“5.5 画面データ記述項”または、“10.3.2.1 画面データ記述項”を参照してください。

10.4 日本語機能

ここでは、Micro Focus固有機能の日本語機能について説明します。

10.4.1 手続き部

ここでは、字類条件について説明します。

10.4.1.1 字類条件

字類条件は、データ項目の内容の字類を検査するために使います。

JAPANESE検査が指定できます。

【書き方】

一意名-1 IS [NOT] { NUMERIC / ALPHABETIC / ALPHABETIC-LOWER / ALPHABETIC-UPPER / JAPANESE / BOOLEAN / 字類名 }

---

1. JAPANESE検査が真になる条件は、一意名-1のデータ項目の内容が、日本語文字および半角カナである場合です。
2. 一意名-1は、以下のいずれかでなければなりません。
   - 用途が表示用のデータ項目
   - 英数字関数および日本語関数の関数一意名
3. JAPANESE検査の場合、一意名-1に以下のデータ項目を指定することはできません。
   - 数字またはブールと指定した項目
4. JAPANESE検査は、動作モードがUnicode(ISO/IEC 10646-1)の場合、使用できません。また、[.NET]では、JAPANESE検査は使用できません。

10.5 入出力文

ここでは、Micro Focus固有機能の入出力文について説明します。

10.5.1 手続き部

ここでは、以下の文について説明します。

- READ文(相対ファイル・索引ファイル)
- START文(相対ファイル)
- START文(索引ファイル)

10.5.1.1 READ文（相対ファイル・索引ファイル）

ファイルからレコードを読みます。

【書き方1】レコードを順に読む（相対ファイル・索引ファイル）

READ ファイル名-1 [{ NEXT / PREVIOUS } RECORD]

[INTO 一意名-1]

[WITH [NO] LOCK]

[AT END 無条件文-1]

[NOT AT END 無条件文-2]

[END-READ]

【書き方2】レコードを乱に読む（相対ファイル）

READ ファイル名-1 RECORD
[INTO 一意名-1]
[WITH [NO] LOCK]
[INVALID KEY 無条件文-3]
[NOT INVALID KEY 無条件文-4]
[END-READ]

【書き方3】レコードを乱に読む（索引ファイル）

READ ファイル名-1 RECORD
[INTO 一意名-1]
[WITH [NO] LOCK]
[KEY IS {データ名-1} …]
[INVALID KEY 無条件文-3]
[NOT INVALID KEY 無条件文-4]
[END-READ]

構文規則

各ファイルに共通する規則

“6.4.34 READ文（順ファイル・相対ファイル・索引ファイル）”を参照してください。

相対ファイルの規則

1. 順呼出し法のファイルを呼び出すときには、書き方1を用います。このとき、PREVIOUS指定を書いてはなりません。
2. 動的呼出し法のファイルから順にレコードを呼び出す場合、書き方1を使います。このとき、NEXT指定を書かなければなりません。
3. 動的呼出し法のファイルのレコードを逆順に呼び出す場合、書き方1を使います。このとき、PREVIOUS指定を書かなければなりません。
4. 乱呼出し法のファイルからレコードを呼び出す場合、または動的呼出し法のファイルから乱にレコードを呼び出す場合、書き方2を使います。

索引ファイルの規則

1. 順呼出し法のファイルを呼び出す場合、書き方1を使います。このとき、PREVIOUS指定を書いてはなりません。
2. 動的呼出し法のファイルから順にレコードを呼び出す場合、書き方1を使います。このとき、NEXT指定は書かなければなりません。
3. 動的呼出し法のファイルのレコードを逆順に呼び出す場合、PREVIOUS指定を書かなければなりません。
4. 乱呼出し法のファイルからレコードを呼び出す場合、または動的呼出し法のファイルから乱にレコードを呼び出す場合、書き方3を使います。

一般規則

書き方1～書き方3に共通する規則

“6.4.34 READ文（順ファイル・相対ファイル・索引ファイル）”を参照してください。

書き方-1の規則

各ファイルに共通する規則

1. ファイル名-1に順呼出し法の相対ファイルまたは順呼出し法の索引ファイルを指定した場合、NEXT指定は省略することができます。NEXT指定はREAD文の実行には影響を与えません。
2. ファイル名-1に順ファイル、順呼出し法の相対ファイルまたは順呼出し法の索引ファイルを指定した場合、ファイル名-1のファイルから次のレコードを呼び出します。
3. ファイル名-1に動的呼出し法の相対ファイルまたは動的呼出し法の索引ファイルを指定した場合、NEXT指定を書くとファイル名-1から次のレコードを呼び出し、PREVIOUS指定を書くとファイル名-1から直前のレコードを呼び出します。
4. READ文を実行すると、使用可能なレコードを定めるための規則に従って、ファイル位置指示子の値が決定されます。ファイル位置指示子の値によって、次にとる動作が決定されます。

PREVIOUS指定を書かない場合の使用可能なレコードを定めるための規則

“6.4.34 READ文（順ファイル・相対ファイル・索引ファイル）”を参照してください。

PREVIOUS指定を書いた場合の使用可能なレコードを定めるための規則(相対ファイルの場合)

1. READ文の実行が開始されるときのファイル位置指示子の値は、以下の規則に従って使用可能なレコードを定めるために使われます。相対ファイルでのレコードの比較は、相対キー番号によって行われます。
    a. ファイル位置指示子が有効な直前のレコードがないことを示しているとき、READ文の実行は不成功になります。
    b. ファイル位置指示子が不定ファイルが存在しないことを示している場合、ファイル終了条件が発生します。
    c. 先に実行されたOPEN文によりファイル位置指示子が設定されている場合、ファイル終了条件が発生します。
    d. 先に実行されたSTART文によりファイル位置指示子が設定されている場合、そのファイル位置指示子と同じ相対レコード番号を持つレコードが選択されます。
    e. 先に実行されたREAD文によりファイル位置指示子が設定されている場合、そのファイル位置指示子より小さい相対レコード番号を持つ最初のレコードが選択されます。
2. 1.のd.またはe.の条件を満足するレコードが見つからなかった場合、ファイル終了条件が発生します。ファイル位置指示子には、次のレコードが存在しないことを示す値が設定されます。
3. 1.のd.またはe.の条件を満足するレコード(「選択されたレコード」という)が見つかった場合、以下の処理が行われます。
    a. ファイル名-1に対してRELATIVE KEY指定が書かれ、選択されたレコードの相対レコード番号の有効桁数が、相対キーデータ項目の大きさより大きいとき、ファイル位置指示子がこの条件を示すように設置され、ファイル終了条件が発生します。
    相対レコード番号の有効桁数が相対キー項目の大きさ以下の場合、選択されたレコードがファイル名-1のレコード領域で使用可能になります。このとき、ファイル位置指示子には、使用可能になったレコードの相対レコード番号が設定されます。また、転記の規則に従って、使用可能になったレコードの相対レコード番号が相対キー項目に転記されます。
    b. ファイル名-1に対してRELATIVE KEY指定を書かなかった場合、選択されたレコードが、ファイル名-1のレコード領域で使用可能になります。このとき、ファイル位置指示子には、使用可能になったレコードの相対レコード番号が設定されます。

PREVIOUS指定を書いた場合の使用可能なレコードを定めるための規則(索引ファイルの場合)

1. READ文の実行が開始されるときのファイル位置指示子の値は、以下の規則に従って、使用可能なレコードを定めるために使われます。索引ファイルでのレコードの比較は、現在の参照キーの値の大小順序によって行われます。

    a. ファイル位置指示子が直前のレコードが存在しないことを示している場合、READ文の実行は不成功になります。

    b. ファイル位置指示子が不定ファイルが存在しないことを示している場合、ファイル終了条件が発生します。

    c. 先に実行されたOPEN文によりファイル位置指示子が設定されている場合、ファイル終了条件が発生します。

    d. 先に実行されたSTART文によりファイル位置指示子が設定された場合、以下のいずれかの条件を満足するレコードが選択されます。

    - START文によって位置付けられたレコード。

    - 前項のレコードが見つからないときまたは選択の対象ではないとき、選択の対象となるレコードのうち、そのレコードの論理的に直後または直前のレコード。

    直後のレコードとなるか直前のレコードとなるかは、START文の比較条件によって決定されます。

    ただし、REVERSED ORDER指定付きのSTART文によりファイル位置指示子が設定された場合には、選択の対象となるレコードのうち、そのレコードの論理的に直前のレコードとなります。すなわち、ファイルの末尾のレコードから先頭のレコードへ向かう方向において、論理的に次のレコードです。

    e. 先に実行されたREAD文によりファイル位置指示子が設定されている場合、以下の条件を満足するレコードが選択されます。

    - REVERSED ORDER指定付きのSTART文の実行により検索方向が逆順であることが指定されている場合、またはPREVIOUS指定が書かれた場合、先に実行されたREAD文により使用可能となったレコードの論理的に直前のレコード。すなわちファイルの末尾のレコードから先頭のレコードへ向かう方向において、論理的に次のレコード。

    - REVERSED ORDER指定付きのSTART文の実行により検索方向が逆順であることが指定されている場合、かつPREVIOUS指定が書かれた場合、先に実行されたREAD文により使用可能となったレコードの論理的に直後のレコード。

2. 1.のd.およびe.で、条件を満足するレコードが見つからなかった場合、ファイル終了条件が発生します。ファイル位置指示子には、次のレコードが存在しないことを示す値が設定されます。

3. 1.のd.およびe.で、条件を満足するレコードが見つかった場合、そのレコードがファイル名-1のレコード領域で使用可能になり、ファイル位置指示子には、使用可能になったレコードの現在の参照キーの値が設定されます。

4. READ文の実行が不成功になった場合、参照キーの値は規定されません。

5. 主レコードキーまたは副レコードキーを参照キーとしてREAD文を実行した場合、参照キーの値を持つレコードが重複して存在することがあります。その場合、以下の順にレコードが呼び出されます。

    a. 先に実行したREVERSED ORDER指定付きのSTART文によりファイル位置指示子が設定されている場合、またはSTART文の比較条件によって、重複した最後のレコードに位置づけられた状態でREAD文にPREVIOUS指定がある場合、WRITE文またはREWRITE文で、参照キーの値が重複するレコードを生成した順と逆の順に、レコードが呼び出されます。

    b. 上記以外の場合、WRITE文またはREWRITE文で、参照キーの値が重複するレコードを生成した順に、レコードが呼び出されます。

6. 動的呼出し法のファイルに対して、書き方3のREAD文に続いて書き方1のREAD文を実行すると、書き方1のREAD文では参照キーが変更されるまで、書き方3のREAD文の実行によって設定された参照キーが使用され、以下に示す検索方向でレコードが呼び出されます。

    a. REVERSED ORDER指定付きのSTART文が実行されるまで、正順でレコードを呼び出します。

    b. REVERSED ORDER指定付きのSTART文が実行されると、逆順にレコードを呼び出します。

c. PREVIOUS指定を書くと、そのファイルから論理的に直前のレコードを呼び出します。ただし、先立つSTART文にREVERSED ORDER指定がされており、論理的に逆順への検索となっていた場合、PREVIOUS指定によって正順のレコードを呼び出します。

例外条件の発生の有無による制御の移行

1. READ文の実行中にファイル終了条件が発生すると、READ文の実行が不成功になります。ファイル名-1の入出力状態にファイル終了条件を示す値が設定された後、"6.3.13 AT END指定"の規則に従って制御が移ります。
2. READ文の実行中にファイル位置指示子が、次のレコードがないこと、相対レコード番号の有効桁数が相対キー項目の大きさより大きいこと、または不定ファイルが存在しないことを示しているとき以下の動作がこの順に行われます。
   ただし、PREVIOUS指定が書かれていた場合は、ファイル位置指示子が、直前のレコードがないこと、または不定ファイルが存在しないことを示しているときに以下の動作がこの順に行われます。
   a. ファイル位置指示子およびファイル名-1の入出力状態が設定されます。
   b. "6.3.13 AT END指定"の規則に従って制御が移ります。
3. READ文の実行中にファイル終了条件もその他の例外条件も発生しなかった場合、以下の処理が順に行われます。
   a. ファイル位置指示子およびファイル名-1の入出力状態が設定されます。
   b. 読み込んだレコードがレコード領域で使用可能になります。INTO指定を書いた場合、暗黙の転記も実行されます。
   c. "6.3.13 AT END指定"の規則に従って制御が移ります。

書き方2および書き方3の規則

"6.4.34 READ文（順ファイル・相対ファイル・索引ファイル）"を参照してください。

10.5.1.2 START文（相対ファイル）

レコードを順呼出しするためにファイルを論理的に位置付けます。

【書き方】

START ファイル名-1

[KEY { IS EQUAL TO / IS = / IS GREATER THAN / IS > / IS NOT LESS THAN / IS NOT < / IS GREATER THAN OR EQUAL TO / IS >= / IS LESS THAN / IS < / IS NOT GREATER THAN / IS NOT > / IS LESS THAN OR EQUAL TO / IS <= } データ名-1]

[INVALID KEY 無条件文-1]

[NOT INVALID KEY 無条件文-2]

[END-START]

注意

“＝”、“＞”、“＜”、“＞＝”および“＜＝”は必要語です。ここでは、他の記号との混同を避けるために下線を付けていません。

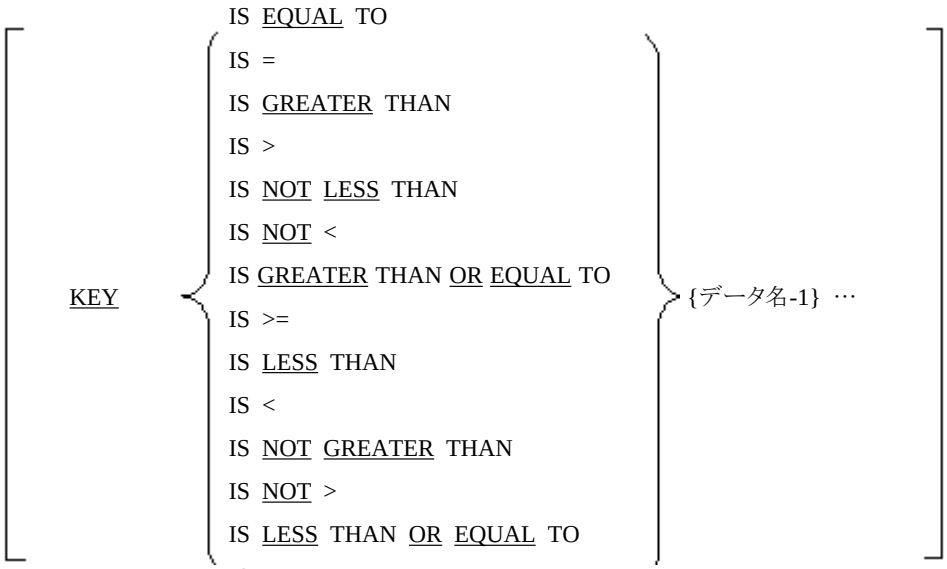
構文規則、一般規則ともに“6.4.42 START文（相対ファイル）”を参照してください。

10.5.1.3 START文（索引ファイル）

レコードを順呼出しするためにファイルを論理的に位置付けます。

【書き方1】正順に順呼出しするために、ファイルを論理的に位置付ける

START ファイル名-1



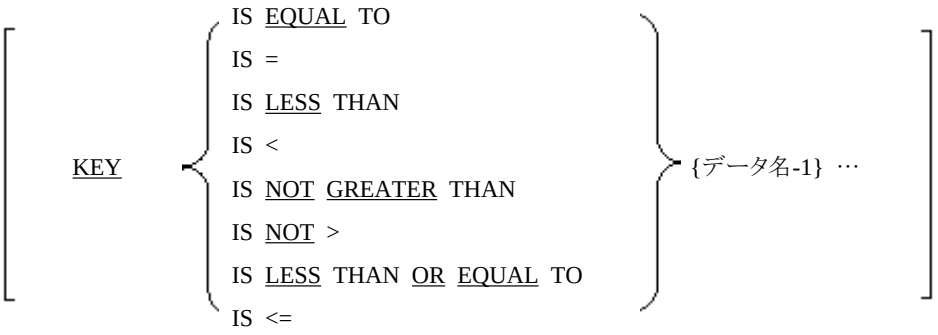
[INVALID KEY 無条件文-1]

[NOT INVALID KEY 無条件文-2]

[END-START]

【書き方2】逆順に順呼出しするために、ファイルを論理的に位置付ける

START ファイル名-1



WITH REVERSED ORDER

[INVALID KEY 無条件文-1]

[NOT INVALID KEY 無条件文-2]

[END-START]

【書き方3】参照キーの先頭または末尾のレコードにファイルを論理的に位置付ける

START ファイル名-1

FIRST RECORD

[KEY IS {データ名-1} …]

[WITH REVERSED ORDER]

[INVALID KEY 無条件文-1]

[NOT INVALID KEY 無条件文-2]

[END-START]

注意

“＝”、“＞”、“＜”、“＞＝”および“＜＝”は必要語です。ここでは、他の記号との混同を避けるために下線を付けていません。

構文規則

“6.4.43 START文（索引ファイル）”を参照してください。

一般規則

書き方1～書き方3に共通する規則

“6.4.43 START文（索引ファイル）”を参照してください。

書き方1および書き方2に共通する規則

1. KEY指定の記述に従って、ファイル名-1のファイルに存在するレコードのキーと、以下のデータ項目が比較されます。
   a. KEY指定を書いた場合、データ名-1のデータ項目が比較されます。
   b. KEY指定を省略した場合、ファイル名-1のRECORD KEY句に書いたデータ項目が比較されます。KEY指定の比較演算子には“IS EQUAL TO”を書いたものとみなされます。

2. 1.の比較は、文字の大小順序に従って昇順に並んでいる、ファイル内のレコードの参照キーに対して行われます。作用対象の長さが異なる場合、長い方の右側が短い方と同じ長さになるように切り捨てられ、文字比較が行われます。比較の結果、以下の処理が行われます。

   a. 書き方1の場合、ファイル位置指示子は、比較条件を満足する最初または最後のレコードの示す値に設定されます。このとき、最初のレコードに設定されるか最後のレコードに設定されるかは、比較演算子によって決定されます。

   b. 書き方2の場合、ファイル位置指示子は、比較条件を満足する最後のレコードの示す値に設定されます。

   c. ファイル中のどのレコードも上記の条件を満足しない場合、無効キー条件が発生し、START文の実行は不成功になります。

書き方3の規則

“6.4.43 START文（索引ファイル）”を参照してください。

10.6 プログラム間連絡機能

10.6.1 手続き部

ここでは、以下の文について説明します。

- CALL文(プログラム間連絡)

10.6.1.1 CALL文（プログラム間連絡）

実行単位中の他のプログラムに制御を移します。

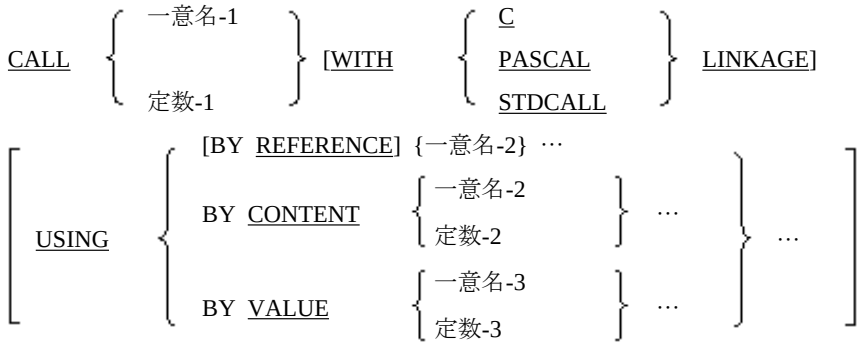
注意

この機能は、[Win32][WinIPF][Winx64][Solaris][Linux][LinuxIPF][Linux64]固有の機能です。

【書き方1】ON OVERFLOW指定

```
CALL { 一意名-1 / 定数-1 } [WITH { C / PASCAL / STDCALL } LINKAGE]
[ USING { [BY REFERENCE] {一意名-2} …
        / BY CONTENT { 一意名-2 / 定数-2 } … } …
        / BY VALUE { 一意名-3 / 定数-3 } … } … ]
  [RETURNING 一意名-4]
  [ON OVERFLOW 無条件文-1]
  [END-CALL]
```

【書き方2】ON EXCEPTION指定



[RETURNING 一意名-4]

[ON OVERFLOW 無条件文-1]

[NOT ON EXCEPTION 無条件文-2]

[END-CALL]

注意

WITH指定は[Win16][Win32]固有機能です。

定数-3は[Win32][WinIPF][Winx64][Solaris][Linux][LinuxIPF][Linux64]固有機能です。

構文規則

1. 定数-3は、9桁以下の数字定数でなければなりません。ただし、[WinIPF][Winx64][LinuxIPF][Linux64]の場合、定数-3は、18桁以下の数字定数でなければなりません。
2. その他の構文規則については、“6.4.6 CALL文(プログラム間連絡)”を参照してください。

一般規則

一般規則については、“6.4.6 CALL文(プログラム間連絡)”を参照してください。

第11章　.NETプログラミング機能

本章では、.NETプログラミング機能を使ってCOBOLプログラムを書くための文法規則について説明します。

.NETプログラミング機能は、[.NET]固有の機能です。

11.1 .NETとは

NetCOBOL for .NETは、Microsoft .NET Frameworkに対応するプログラミング機能(.NETプログラミング機能)を提供します。Microsoft .NET Frameworkは、一般的に.NETと呼ばれます。(ドットネットと読みます。)

.NETは、アプリケーションとWebサービスの統合をサポートするプラットフォームです。.NETでは、多くの異なるプログラミング言語でアプリケーションを開発できます。異なる言語のプログラムから作成されたオブジェクトは、共通のランタイムシステムによって実行することができます。

この共通の環境でCOBOLを利用するために、オブジェクト指向プログラミング機能の概念を取り込んだ.NETプログラミング機能を提供します。

.NETプログラミング機能は、以下の機能を持ちます。

- スタティック定義およびオブジェクト定義を含む、クラスを定義する機能
- スタティックおよびオブジェクトインスタンスのメソッドを定義する機能
- スタティックおよびオブジェクトインスタンス中にカプセル化されたデータを定義する機能
- 単一のクラスを継承し、新しいクラスを定義する機能
- 複数のインタフェースを実装する機能
- 多態を使用する機能
- オブジェクトへの参照を保持するデータ項目を定義する機能
- オブジェクトのメソッドを呼び出す機能
- 引数の数や属性が異なる同名のメソッドを定義または使用する機能
- オブジェクトのフィールドを参照する機能
- 要求されたオブジェクトインスタンスを作成する機能
- デリゲートを扱う機能
- .NET以外のプラットフォームで動作する実行形式ファイルを呼び出す機能
- 総称プログラミングを実現する機能

.NETプログラミング機能では、従来の機能もそのまま利用することができます。

11.2 用語の説明

ここでは、以下の用語を使用します。

11.2.1 オブジェクト指向プログラミング機能の用語

インタフェース(メソッドの)(interface(for a method))

メソッドを正しく呼び出すために必要な情報(名前とパラメタの仕様)です。この情報は、メソッド原型として指定することができます。

インタフェース指定の特殊クラス(special class of interface specification)

特定のインタフェースを持つ特殊オブジェクトが属するクラスです。

オブジェクト(インスタンス)(object(instance))

データおよびそのデータ上で動作するメソッドから構成される単位です。

オブジェクト一意名(object identifier)

オブジェクトを識別する一意名です。オブジェクト一意名は、オブジェクト参照一意名とも呼びます。

オブジェクト参照一意名(object reference identifier)

オブジェクト参照によりオブジェクトを識別する一意名です。

オブジェクト定義(object definition)

オブジェクトを定義するCOBOLソース単位です。

オブジェクトデータ(object data)

オブジェクト定義のデータ部に記述されたデータです。メソッドに記述されたデータは除きます。

オブジェクトプロパティ(object property)

オブジェクトの値を参照および設定するために、オブジェクト参照で修飾して使用される名前です。

オブジェクトメソッド(object method)

オブジェクトのメソッド(ファクトリメソッドに対応)です。

親クラス(parent class)

他のクラスによって継承されるクラスです。2つのクラスが継承の関係にある場合、継承するクラスが子クラスであり、継承されるクラスが親クラスです。スーパクラスと同義語です。

クラス(class)

ゼロ個以上のオブジェクトに対して、共通の動作と属性を定義したものです。同一の性質を持つオブジェクトは、同じクラスのオブジェクトとみなされます。

クラス定義(class definition)

クラスを定義するCOBOLソース単位です。

クラス名(class-name)

クラスを識別する利用者語です。

継承(inheritance)

1つ以上のクラスの属性を他のクラスの属性の基底として使用する方法です。サブクラスは、1つ以上のスーパクラスを継承します。継承したクラスは、継承されたクラスに適合します。

子クラス(child class)

他のクラスを継承するクラスです。2つのクラスが継承の関係にある場合、継承するクラスが子クラスであり、継承されるクラスが親クラスです。サブクラスと同義語です。

サブクラス(sub-class)

他のクラスを継承するクラスです。2つのクラスが継承の関係にある場合、継承するクラスがサブクラスであり、継承されるクラスがスーパクラスです。子クラスと同義語です。

スーパクラス(super-class)

他のクラスによって継承されるクラスです。2つのクラスが継承の関係にある場合、継承するクラスがサブクラスであり、継承されるクラスがスーパクラスです。親クラスと同義語です。

ソース単位(source unit)

見出し部から、終わり見出しまたは翻訳集団の終わりまでで、含まれる他のソース単位も含めた文の並びです。

ソース要素(source element)

ソース単位内に含まれるほかのソース単位を除いた文の並びです。

束縛(binding)

メソッド呼出しを、メソッドの実体と結びつける処理です。翻訳時にオブジェクトのクラスがわかる場合は、翻訳時に束縛されます。この場合を静的束縛または早期束縛と呼びます。翻訳時に束縛できない場合を、動的束縛または遅延束縛と呼びます。この場合、メソッド呼出しを受けたオブジェクトのクラスに従って、メソッドを決定します。

定義済みオブジェクト一意名(predefined object identifier)

特殊なオブジェクトを識別するために使用する予約語です。定義済みオブジェクト一意名には、EXCEPTION-OBJECT、NULL、SELFおよびSUPERがあります。

適合(conformance)

異なるインタフェースのオブジェクトが期待される場所で、オブジェクトがそれ自身のインタフェースで使用されることを許す性質です。適合は、適合したインタフェースが、適合されたインタフェースに対するすべての操作をサポートすることを保証します。

復帰項目(returning item)

プログラムまたはメソッドの実行結果を返すために使われるデータ項目です。

プロパティ名(property-name)

プロパティ名は、オブジェクトから情報を取り出したり、オブジェクトに情報を渡す手段であるオブジェクトプロパティを識別します。

翻訳集団(compilation group)

一緒に翻訳されるCOBOLソース単位の並びです。

メソッド(method)

メソッドの手続き部で宣言された手続きコードです。そのオブジェクトに対するメソッド呼出しによって実行されます。

メソッド定義(method definition)

メソッドを定義するCOBOLソース単位です。

メソッドデータ(method data)

メソッド定義のデータ部で宣言されたデータです。

メソッド名(method-name)

メソッドを識別する利用者語です。

メソッド呼出し(method invocation)、呼出し(invocation)

与えられたオブジェクト上での、名前により指定されたメソッドの実行要求です。メソッド呼出しは、オブジェクト、メソッド名およびメソッド定義で必要とされたパラメタを識別します。

呼出し演算子(invocation operator)

連続したコロンの組“::”です。メソッドの行内呼出しで使われます。

例外オブジェクト(exception object)

例外条件として利用できるオブジェクトです。

例外条件(exception condition)

プログラムの実行中に検出される条件で、正常な処理に対してエラーまたは例外が起きていることを示します。

11.2.2 .NETプログラミング機能の用語

値型(value type)

CLRが扱うオブジェクトの種類の1つです。値型であるオブジェクトの内容は、オブジェクトの値そのものになります。

アトリビュートクラス(attribute class)

.NETシステムが提供するSystem.Attributeクラスから派生したクラスです。

アクセス属性(access attributes)

アクセス属性には、PUBLIC、PRIVATE、PROTECTEDの3種類があります。

PUBLICは、アクセスの制限はありません。PRIVATEは、型定義内部からだけアクセスできます。PROTECTEDは、型定義の内部と派生したクラスからだけアクセスできます。

ENUM(enumeration)

列挙型を表します。列挙型は、名前の付いた整数を列挙したデータ型です。

カスタム属性(custom-attribute)

メタデータ上での注釈であり、実行時の動作を規定する情報になります。

カスタム属性は、利用者が任意に定義することができるカスタム属性と、.NETシステムが用意する擬似カスタム属性の2種類に分けられます。

型パラメタ(type parameter)

型（クラス、インタフェースおよびデリゲート）またはメソッドの定義内で、参照する型を動的に決定可能とするために使用する特殊なパラメタです。COBOLでは、型の名前として参照できます。

型引数(type argument)

型パラメタの表現する型を決定するため、型パラメタを持つ型やメソッドに指定する型です。制約を満たせば、他の総称型の型パラメタで表現される型を含む任意の型を指定できます。

仮想属性(virtual attribute)

オブジェクト定義に含まれるメソッド定義に与えられる属性です。そのメソッド定義で宣言されたメソッドまたはプロパティは仮想属性を持ちます。

(インスタンス)コンストラクタ((instance) constructor)

インスタンスコンストラクタ（単にコンストラクタとも呼びます）は、オブジェクトインスタンスの初期化手続きです。

.NETプログラミング機能では、“NEW”メソッドを、インスタンスコンストラクタとみなします。COBOLソース内での“NEW”メソッドの宣言および参照は、インスタンスコンストラクタの宣言および参照と同義です。

このコンパイラでは、オブジェクト定義のメソッド名“NEW”(すべて大文字)を、インスタンスコンストラクタとみなします。COBOLソース内でのオブジェクトメソッド“NEW”の宣言/参照は、インスタンスコンストラクタの宣言/参照と同義です。

構成型(constructed type)

総称型に型引数を与えて、操作可能とした型です。型引数として与えられた型のインタフェースが確定しているかどうかで、閉じた構成型/閉じていない構成型に分かれます。

COBOL独自型データ(COBOL original type data)

CLSで規定される範囲外のデータ型を持つデータです。

コレクション(collection)

同じ種類のオブジェクトの集まりです。配列、リストなどの形状を持ちます。

最終クラス(sealed class /final class)

クラスが具象クラスで、拡張できないクラスを指します。このクラスを継承することはできません。最終クラスは、COBOLでは定義できません。

参照型(reference type)

CLRが扱うオブジェクトの種類の1つです。参照型であるオブジェクトの内容は、オブジェクトの値への参照です。

CLR(Common Language Runtime)

すべての言語を統合する、.NET共通の実行環境です。

CLR型データ(CLR type data)

CLSが実装を要求しているデータ型に相当する属性を持つデータです。

CLS(Common Language Specification)

.NETにおいて、異なる言語間の相互運用性を高めるための規定です。

(メソッドの)シグネチャ((method) signature)

メソッドの名前、引数の個数と型、および、復帰値の型で構成されるメソッドを識別するための情報です。

実体化(instantiation)

総称性を持つ型やメソッドに型引数を指定して、操作可能とすることです。

スタティック(static)

.NETでの“static”は、メソッドまたはデータを修飾する語です。スタティックなメソッドやデータは、クラスで一度だけ生成されます。(インスタンスごとに生成されるものではありません)

.NETプログラミング機能では、スタティックなメソッドまたはデータは、スタティック定義内で宣言します。スタティック定義は、オブジェクト指向プログラミング機能のファクトリ定義と似ていますが、親クラスから継承したメソッドを置き換えることはできません。また、動的にメソッドを呼び出すこともできません。

スタティックコンストラクタ(static constructor)

スタティックオブジェクトの初期化手続きです。

このコンパイラでは、スタティック定義のメソッド名“NEW”(すべて大文字)を、スタティックコンストラクタとみなします。COBOLソース内でのスタティックメソッド“NEW”の宣言は、スタティックコンストラクタの宣言と同義です。

スタティックデータ(static data)

スタティック定義のデータ部で宣言されたデータです。オブジェクトが削除されるまでの間、最後に更新された状態を保持します。

スタティック定義(static definition)

スタティックデータおよびスタティックメソッドを定義するCOBOLソースの単位です。

スタティックメソッド(static method)

スタティック定義内で宣言されたメソッドです。

制約(constraint)

型引数に指定可能な型が許容すべき条件です。型パラメタに対して指定します。

総称性(generics)

定義済みの手続きに対し、手続き中で参照するデータの型を動的に決定可能にして、手続きの再利用を可能とする機能(ジェネリック)です。COBOLでは型パラメタ/型引数によって実現されます。

総称型(generic type)

1つ以上の型パラメタを持ち、総称性を持つ型です。
クラス、インタフェースおよびデリゲートが総称型となることができます。総称型は、そのままでは操作の対象にできません。

総称クラス(generic class)

1つ以上の型パラメタを持ち、総称性を持つクラスです。

総称インタフェース(generic interface)

1つ以上の型パラメタを持ち、総称性を持つインタフェースです。

総称デリゲート(generic delegate)

1つ以上の型パラメタを持ち、総称性を持つデリゲートです。

総称メソッド(generic method)

1つ以上の型パラメタを持つメソッドです。
型パラメタを持つメソッドは、それが含む型と独立に総称性を持ちます。
これに対して、総称型内に含まれるメソッドは、メソッド自身の型パラメタの有無にかかわらず、総称性を持ちます。そのため、広義には、総称型内に含まれるメソッドも、総称メソッドと呼ぶことがあります。後者を区別して、“型パラメタを持つメソッド”呼ぶことがあります。

閉じた(bounded)/閉じていない(unbounded)

型引数によって、型パラメタの表現する型のインタフェースが確定したかどうかを示します。型引数として指定された型の中に、型パラメタで表現される型がない場合、これを閉じていると呼びます。

DLLIMPORT属性(dllimport attribute)

.NETが提供しているDllImportAttributeクラスに対応付けたカスタム属性です。

.NETプログラミング機能では、.NET上から非管理コードを呼び出す際にDLLIMPORT属性を定義し、利用します。

名前空間(name space)

名前によってアプリケーションを階層的に管理する方法、または、その管理のために付けられた名前そのものを指します。後者の場合、たとえば、.NET上には「System」という名前空間が与えられています。

非管理コード(プログラム)(unmanaged code(program))

CLR上で実行されるコードを管理コードと呼ぶのに対して、それ以外のコード、たとえば、Windows関数などCLRに依存しないで実行されるコードを非管理コードと呼びます。

非公開クラス(private class)

他のオブジェクトに公開しないクラスです。非公開クラスはCOBOLでは定義できません。

プログラム原型(program prototype)

プログラム名、仮引数および復帰項目で定義されるプログラムの型です。

プログラム原型定義(program prototype definition)

プログラムの型を定義するCOBOLソース単位です。.NETプログラミング機能では、プログラム原型定義によって非管理コードのプログラム原型を定義します。

メタデータ(metadata)

自己記述型の型情報であり、型情報を公開するための共通フォーマットとして利用されます。メタデータはDLL、EXEなどの実行形式ファイルの中に含まれます。

タイプオブジェクト(type object)

.NETが提供するSystem.Typeクラスのオブジェクトインスタンスです。

11.3 全般規定

ここでは、.NETプログラミング機能における全般規定について説明します。

11.3.1 言語の基本要素

ここでは、.NETプログラミング機能におけるCOBOLの語および表意定数について説明します。

11.3.1.1 COBOLの語

以下のCOBOLの語について示します。

- 利用者語
- 予約語

11.3.1.1.1 利用者語

.NETプログラミング機能固有の利用者語を、以下に示します。

クラス名

クラス名は、ゼロ個以上のオブジェクトに対して、共通の動作と属性を定義した存在であるクラスを識別します。

メソッド名

メソッド名は、メソッドを識別します。

プロパティ名

プロパティ名は、オブジェクトから情報を取り出したり、オブジェクトに情報を渡す手段であるオブジェクトプロパティを識別します。

インタフェース名

インタフェース名は、メソッド原型の集合であるインタフェースを識別します。

デリゲート名

デリゲート名は、手続きとその手続きが実行されるオブジェクトとをカプセル化した存在であるデリゲートを識別します。

ENUM名

ENUM名は、列挙型を識別します。

フィールド名

フィールド名は、型をもつ領域であるフィールドを識別します。

カスタム属性名

カスタム属性名は、カスタムアトリビュートに関連付けたカスタム属性を識別します。

プログラム原型名

プログラム原型名は、プログラムの型の名前であり、呼び出すプログラムを識別するために利用します。

メソッド識別名

メソッド識別名は、型パラメタを持つメソッドとその型引数として指定される型の組合せを識別します。

11.3.1.1.2 予約語

予約語の一覧については、“付録A 予約語一覧”を参照してください。以下は、.NETプログラミング機能固有の予約語です。

- CUSTOM-ATTRIBUTE
- DELEGATE
- DELEGATE-ID
- ENUM
- ENUM-ID
- INTERFACE
- INTERFACE-ID
- PRIVATE
- PUBLIC
- STATIC

特殊用途語

特殊用途語は、次のとおりです。

― 定義済みオブジェクト一意名

定義済みオブジェクト一意名

いくつかの予約語は、定義済みオブジェクト一意名として使用されます。

定義済みオブジェクト一意名は、次のとおりです。

― EXCEPTION-OBJECT

― NULL

― SELF

― SUPER

11.3.1.2 表意定数

1. プログラム定義以外では、表意定数のALL定数は、定数の長さが2以上の場合、数字または数字編集項目に関連付けてはなりません。
2. 表意定数は、文字定数がメソッドを識別するために使用されるところには指定してはなりません。

11.3.2 データ記述の概念

ここでは、.NETプログラミング機能固有のデータ記述項の概念を説明します。

字類の概念

すべての基本データ項目、すべての定数およびすべての関数は、字類および項類を持ちます。

“表11.1 オブジェクトデータ項目の字類および項類の関係”に、オブジェクトデータ項目の字類および項類の関係を示します。

表11.1 オブジェクトデータ項目の字類および項類の関係

項類	字類
オブジェクト参照	オブジェクト

11.3.3 一意参照

ここでは、.NETプログラミング機能における一意参照について説明します。

11.3.3.1 一意名

【書き方1】(関数一意名)

関数一意名-1

【書き方2】(添字付きの修飾されたデータ名)

データ名-1 [{ IN / OF } データ名-2] … [{ IN / OF } ファイル名-1]

[{添え字-1} …]

【書き方3】(部分参照)

一意名-1 部分参照子-1

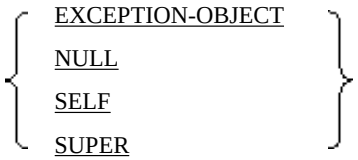
【書き方4】(メソッドの行内呼出し)

行内呼出し-1

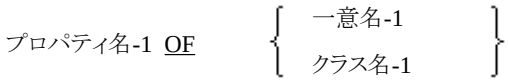
【書き方5】(オブジェクト指定子)

一意名-2 オブジェクト参照子-1

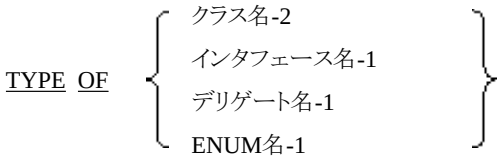
【書き方6】(定義済みオブジェクト一意名)



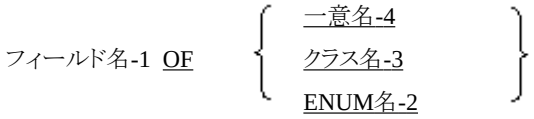
【書き方7】(オブジェクトプロパティ)



【書き方8】(TYPE OF特殊レジスタ)



【書き方9】(フィールド参照)



構文規則

書き方1～書き方9に共通する規則

一意名は再帰的に定義されます。一意名の書き方が、ほかの一意名の指定を許していれば、その一意名はどの書き方の一意名も指定できます。

書き方1の規則

関数一意名-1は、“1.4.7 関数一意名”で定義される関数一意名です。

書き方2の規則

1. INとOFは、同義語です。
2. データ名-1が、現在のソース要素の、名前の有効範囲で一意でないとき、その後に一意参照に必要な修飾と添字付けの構文的に正しい組合せを書かなければなりません。“11.3.5.2 名前の範囲”を参照してください。
3. 添字は、“1.4.2 添字付け”で定義されます。

書き方3の規則

部分参照子-1は、“1.4.3 部分参照”で定義されます。

書き方4の規則

行内呼出し-1は、11.3.3.2 メソッドの行内呼出し”で定義されます。

書き方5の規則

オブジェクト指定子は、11.3.3.3 オブジェクト指定子”で定義されます。

書き方6の規則

定義済みオブジェクト一意名は、11.3.3.4 定義済みオブジェクト一意名”で定義されます。

書き方7の規則

オブジェクトプロパティは、11.3.3.5 オブジェクトプロパティ”で定義されます。

書き方8の規則

TYPE OF特殊レジスタは、11.3.3.6 TYPE OF特殊レジスタ”で定義されます。

書き方9の規則

フィールド参照は、11.3.3.7 フィールド参照”で定義されます。

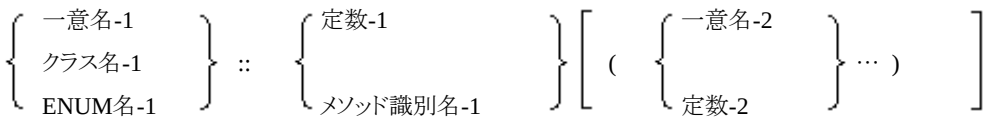
一般規則

1. 一意参照の規則の適用順序は以下のとおりであり、最初の規則から適用します。
    a. 定義済みオブジェクト一意名は、基本一意名です。
    b. TYPE OF特殊レジスタは、右側のクラス名、インタフェース名、デリゲート名、またはENUM名に適用されます。
    c. データ名修飾のOFまたはINは、ファイル名または右側のデータ名により、左側のデータ名(すでに修飾されていてもよい)を修飾します。
    d. 添字は、左側の完全に修飾されたデータ名に適用されます。
    e. オブジェクト指定子は、左側のオブジェクト一意名に適用されます。
    f. オブジェクトプロパティのOFは、左側のプロパティ名を右側のオブジェクト一意名に適用します。
    g. フィールド参照のOFは、左側のフィールド名を右側のオブジェクト一意名に適用します。
    h. メソッドの行内呼出しのための呼出し演算子は、右側に省略可能な括弧で囲まれた引数を持つ定数メソッド名を、左側のオブジェクト一意名に適用します。
    i. 関数一意名は、引数がない場合、基本一意名とみなされます。引数が指定されている場合は、関数名を括弧で囲まれた引数の並びに適用します。
    j. 部分参照は、左側の一意名に適用されます。
2. 以下は、オブジェクト一意名です。
    - USAGE OBJECT REFERENCE句が指定されたデータ記述項の参照
    - 行内呼出し
    - オブジェクトプロパティ
    - 定数オブジェクト一意名
    - フィールド参照

11.3.3.2 メソッドの行内呼出し

メソッドの行内呼出しは、呼び出されたメソッドから返された値を参照します。

【書き方】



構文規則

1. メソッドの行内呼出しは、受取り側項目に記述できません。
2. 一意名-1は、オブジェクト一意名でなければなりません。ただし、NULLであってはなりません。
3. クラス名-1、ENUM名-1はそれぞれ、リポジトリ段落のクラス指定子、ENUM指定子で指定した名前でなければなりません。
4. 定数-1は、その値が一意名-1、クラス名-1またはENUM名-1に対して指定されたメソッド名である文字列、または、日本語文字定数でなければなりません。
5. この形式で識別されるメソッドは、復帰項目を持たなければなりません。
6. 一意名-2は、SELF、EXCEPTION-OBJECT、NULL、あるいは、ファイル節、作業場所節または連絡節で定義されたデータ項目でなければなりません。
7. 定数-2は、この形式で指定されたメソッドに対応する引数と同じ字類および項類の項目へのMOVE文の送出し側として正しくなければなりません。
8. 一意名-2は、引数の適合にしたがって宣言されなければなりません。
9. 一意名-2に内部ブール項目を指定する場合、内部ブール項目はバイト境界で始まるように定義しなければなりません。
10. 一意名-2は、部分参照付けしてはなりません。

一般規則

1. 一意名-1、クラス名-1またはENUM名-1は、その上でメソッドが呼び出されるオブジェクトです。
2. 定数-1またはメソッド識別名-1は、呼び出すメソッドを識別します。
3. メソッドの行内呼出しが使われた場合の動作は、以下のどちらかの文に続いて、一時データ名が使われた場合と同じです。

```
INVOKE 呼び出しオブジェクト“メソッド名”
                            USING  引数の並び
                            RETURNING  一時データ名
INVOKE 呼び出しオブジェクト“メソッド名”
                            RETURNING  一時データ名
```

   a. 引数がある場合、それは行内呼出しのために指定された引数です。
   b. 一時データ名は、一意名-1、クラス名-1またはENUM名-1と定数-1で識別されるメソッドの、宣言中のRETURNING引数と同じ記述、字類および項類を持ちます。
   c. 一時データ名は、行内呼出しのためだけに存在する一時的な項目です。

11.3.3.3 オブジェクト指定子

オブジェクト指定子は、指定された型でオブジェクトを参照するオブジェクト参照を一意に参照する、文字列と分離符の構文的に正しい組合せです。

【書き方】

一意名-1 AS { クラス名-1 / インタフェース名-1 / デリゲート名-1 / ENUM名-1 }

構文規則

1. 一意名-1は、オブジェクト参照一意名でなければなりません。ただし、定義済みオブジェクト一意名SUPERまたはNULLであってはなりません。また、クラス名などのオブジェクトの型を表すものであってはなりません。
2. クラス名-1、インタフェース名-1、デリゲート名-1、ENUM名-1は、それぞれ、リポジトリ段落のクラス指定子、インタフェース指定子、デリゲート指定子、ENUM指定子で指定した名前でなければなりません。
3. 一意名-1は、受取り側項目として参照されていてはなりません。

一般規則

1. オブジェクト指定子は、ASの右側に指定された型を持った、一意名-1より参照されるオブジェクトに対するオブジェクト参照を返します。
2. クラス名-1が指定された場合、結果の暗黙の記述は、USAGE IS OBJECT REFERENCE クラス名-1です。もし、一意名-1により識別されるオブジェクトが、クラス名-1のオブジェクトでも、クラス名-1を継承したクラスのオブジェクトでもなければ、この記述を含む文の実行は、例外を発生させます。
3. インタフェース名-1が指定された場合、結果の暗黙の記述は、USAGE IS OBJECT REFERENCE インタフェース名-1です。もし、一意名-1により識別されるオブジェクトが、インタフェース名-1を実装したクラスのオブジェクトでなければ、この記述を含む文の実行は、例外を発生させます。
4. デリゲート名-1が指定された場合、結果の暗黙の記述は、USAGE IS OBJECT REFERENCE デリゲート名-1です。もし、一意名-1により識別されるオブジェクトが、デリゲート名-1のオブジェクトでなければ、この記述を含む文の実行は、例外を発生させます。
5. ENUM名-1が指定された場合、結果の暗黙の記述は、USAGE IS OBJECT REFERENCE ENUM名-1です。もし、一意名-1により識別されるオブジェクトがENUM名-1のオブジェクトでなければ、この記述を含む文の実行は、例外を発生させます。

11.3.3.4 定義済みオブジェクト一意名

定義済みオブジェクト一意名を以下に示します。

- EXCEPTION-OBJECT
- NULL
- SELF
- SUPER

11.3.3.4.1 EXCEPTION-OBJECT

EXCEPTION-OBJECTは、宣言部分で現在の例外オブジェクトを参照するために使用する定義済みオブジェクト一意名です。

【書き方】

EXCEPTION-OBJECT

構文規則

1. EXCEPTION-OBJECTは、宣言部分でだけ使用できます。
2. EXCEPTION-OBJECTは、受取り側作用対象として使用してはなりません。

一般規則

1. EXCEPTION-OBJECTは、現在の例外オブジェクトを参照します。例外オブジェクトが現在の例外と関係ない場合は、EXCEPTION-OBJECTはNULLです。
2. クラス名指定のUSE文に関連付けられた宣言手続きで使用されているEXCEPTION-OBJECTは、そのクラス名を指定したオブジェクト参照項目データとして暗に宣言されます。

11.3.3.4.2 NULL

NULLは、NULLオブジェクトを参照する定義済みオブジェクト一意名です。

【書き方】

NULL

構文規則

NULLは、受取り側項目として指定してはなりません。

一般規則

NULLは、常に同じオブジェクト、NULLオブジェクトを参照します。

11.3.3.4.3 SELFとSUPER

SELFおよびSUPERは、現在のメソッドが実行されているオブジェクトを参照する、定義済みオブジェクト一意名です。

【書き方】

{ SELF / SUPER }

構文規則

1. この書き方は、オブジェクト定義のメソッドの手続き部でだけ使用できます。
2. この書き方は、受取り側項目として指定してはなりません。
3. SUPERは、メソッドの行内呼出し、オブジェクトプロパティのオブジェクトとして、または、INVOKE文によるメソッド呼出しのオブジェクトとして使用できます。

一般規則

1. 定義済みオブジェクト一意名SELFまたはSUPERは、SELFまたはSUPERが現れるメソッドの呼出しで使われるオブジェクトを参照します。
2. メソッドの呼出しでSELFが指定された場合、メソッドの探索は、そのオブジェクトに対して明または暗に定義されているすべてのメソッドを含んで行われます。
3. メソッドの呼出しでSUPERが指定された場合、メソッドの探索は、メソッドを実行しているのと同じクラスで定義されたすべてのメソッドを無視します。さらに、その同じクラスを継承しているクラスで定義されているすべてのメソッドも無視します。

11.3.3.5 オブジェクトプロパティ

オブジェクトプロパティは、オブジェクトの情報を参照したり、オブジェクトに情報を設定したりするための特殊な構文を提供します。

【書き方】

プロパティ名-1 OF { 一意名-1 / クラス名-1 }

構文規則

1. 一意名-1は、オブジェクト一意名でなければなりません。ただし、NULLであってはなりません。
2. オブジェクトプロパティが送出し側項目として使用された場合、一意名-1により参照されるオブジェクトまたはクラス名-1により参照されるクラスに、プロパティ名-1のプロパティ取出しメソッドが存在していなければなりません。
3. オブジェクトプロパティが受取り側項目として使用された場合、一意名-1により参照されるオブジェクトまたはクラス名-1により参照されるクラスに、プロパティ名-1のプロパティ設定メソッドが存在していなければなりません。
4. 送出し側項目として使用されるオブジェクトプロパティの記述は、プロパティ取出しメソッドの復帰項目の記述と同じです。このオブジェクトプロパティは、送出し側としてその記述のデータ項目が書けるところなら、どこにでも書けます。
5. 受取り側項目として使用されるオブジェクトプロパティの記述は、プロパティ設定メソッドのUSING引数の記述と同じです。このオブジェクトプロパティは、受取り側としてその記述のデータ項目が書けるところなら、どこにでも書けます。
6. プロパティ取出しメソッドのRETURNING指定に書かれた項目のデータ記述は、プロパティ設定メソッドのUSING指定に書かれた項目のデータ記述と同じでなければなりません。
7. オブジェクトプロパティは、部分参照してはなりません。
8. 受取り側項目として使用されるオブジェクトプロパティは、以下の場所にだけ書けます。
   - CORRESPONDING指定のないMOVE文の受取り側
   - SET文の受取り側

一般規則

1. オブジェクトプロパティが送出し側項目としてだけ使われた場合、概念上の一時データ項目、temp-1がその場所で使われます。プロパティの値は、関連付けられたプロパティ取出しメソッドが実行され、復帰値がtemp-1に格納されることによって決められます。temp-1のデータ記述は、プロパティ取出しメソッドのRETURNING指定に書かれたデータ記述と同じです。
2. オブジェクトプロパティが受取り側項目としてだけ使われた場合、概念上の一時データ項目、temp-2がその場所で使われます。プロパティの値は、関連付けられたプロパティ設定メソッドが実行され、temp-2の内容を引数に設定することによって、代入されます。temp-2のデータ記述は、プロパティ設定メソッドのUSING指定に書かれたデータ記述と同じです。

11.3.3.6 TYPE OF特殊レジスタ

TYPE OF特殊レジスタは、指定したオブジェクトを示すタイプオブジェクトを保持します。

【書き方】

TYPE OF { クラス名-1 / インタフェース名-1 / デリゲート名-1 / ENUM名-1 }

構文規則

1. クラス名-1、インタフェース名-1、デリゲート名-1、ENUM名-1は、それぞれリポジトリ段落のクラス指定子、インタフェース指定子、デリゲート指定子、ENUM指定子で指定した名前でなければなりません。
2. この書き方は、受取り側項目として指定してはなりません。

一般規則

TYPE OF特殊レジスタは、クラス名-1、インタフェース名-1、デリゲート名-1またはENUM名-1として指定したオブジェクトを示すタイプオブジェクトを保持します。

11.3.3.7 フィールド参照

フィールド参照は、フィールドを参照する機能を提供します。

【書き方】

フィールド名-1 OF { 一意名-1 / クラス名-1 / ENUM名-1 }

構文規則

一意名-1は、オブジェクト参照一意名でなければなりません。ただし、SUPERおよびNULLを指定することはできません。

一般規則

1. 一意名-1を指定した場合、その一意名-1が参照するオブジェクトに対して明または暗に宣言された、スタティック属性でないフィールドを参照します。
2. クラス名-1を指定した場合、そのクラスで明または暗に宣言されたスタティック属性のフィールドを参照します。
3. ENUM名-1を指定した場合、そのENUMのフィールドを参照します。
4. COBOLでは、スタティック定義およびオブジェクト定義の作業場所節で宣言されるレベル番号01または77の項目がフィールドに相当します。スタティック定義のフィールドはスタティック属性を持ち、オブジェクト定義のフィールドはスタティック属性を持ちません。
5. 長さが可変であるフィールドを参照することはできません。
6. フィールド参照は、部分参照できません。
7. 定数フィールドを参照するフィールド参照は、受け取り側に指定できません。

11.3.4 外部名と内部名

ここでは、.NETプログラミング機能における外部名と内部名について説明します。

外部名は、メタデータに格納される名前であり、ほかのプログラムやクラスから参照されるときの名前です。外部名は定数でも表記できるため、COBOLの語の範囲外の名前を使用するための手段になります。

内部名はCOBOLの利用者語の規則にしたがって命名される名前であり、翻訳単位内の要素の名前を任意に表現します。

以下の要素は外部名と内部名を持ちます。

- クラス
- デリゲート
- インタフェース
- ENUM
- プログラム(最外部のプログラムのみ)
- プログラム原型
- メソッド
- プロパティ
- フィールド
- パラメタ・復帰値

外部名が明に指定されていない場合は、内部名と同名の外部名が暗に指定されたものとみなされます。また、リポジトリ段落で外部名を指定する場合、その外部名は翻訳単位の中で一意な名前でなければなりません。

なお、本書中で上記の名前について特に外部名であることが説明されていなければ、それは内部名を指します。

11.3.4.1 外部名の表記の規則

外部名を記述する場合で、次のようなときに修飾が必要となります。

- 名前空間や他の型の入れ子となっているとき
- 型が配列型であることを示すとき
- 型やメソッドが型パラメタを持つことを示すとき

11.3.4.1.1 名前空間や他の型の入れ子となっている場合の修飾

クラス、インタフェース、デリゲート、ENUM、プログラムおよびプログラム原型の外部名は、各要素の名前を、名前空間や入れ子の上位階層要素の名前で完全修飾して示します。名前空間や入れ子がなければ修飾は不要です。

この名前空間や入れ子の上位階層要素の名前は、次の規則に従って記述する必要があります。

a. 名前空間による修飾は“.”(ピリオド)で示します。

b. 入れ子の上位階層要素による修飾は“+”(プラス)で示します。

c. 入れ子の上位階層要素が型パラメタを持つ場合、“11.3.4.1.3 型やメソッドが型パラメタを持つことを示すための修飾”で示す規則に従った修飾が必要です。

a～cの条件を満たさない場合、あるいはこれらの修飾を付けることができないメソッド、プロパティおよびフィールドの外部名では、外部名に含まれる“.”や“+”は、単なる文字として解釈されます。

```
CLASS CLASS-A  AS "FJnamespace.ClassA"          *> (1)
ENUM  ENUM-B   AS "FJnamespace.ClassA+enumB"    *> (2)
```

たとえば、名前空間FJnamespaceに所属するClassAの場合は、上記の(1)のように記述します。また、enumBがそのクラスClassAに含まれる場合、上記の(2)のように記述します。

11.3.4.1.2 型が配列型であることを示すための修飾

型が配列型であることは、クラス指定子に記述する外部名で指定できます。

配列型であることを示すには、外部名を型の名前の後ろに角括弧([])を付加した文字列として記述します。

この外部名は、次の規則に従って記述する必要があります。

a. 型名に続く角括弧の組は一組しか記述できません。

b. 角括弧の中には、カンマ(,)しか記述できません。(カンマは配列の次元数を表し、括弧内にカンマがない場合は、1次元配列と解釈されます。)

c. 定数-1は角括弧の組で終わっていなければなりません。

d. 型名はクラス名、デリゲート名で、必要なら名前空間や入れ子の上位階層の型の名前で修飾する必要があります。

e. 指定する型名が型パラメタを持つ型である場合、“11.3.4.1.3 型やメソッドが型パラメタを持つことを示すための修飾”で示す規則に従った修飾が必要です。

a～eの条件を満たさない場合は、角括弧([])は配列型の指定と見なされず、単なる文字として解釈されます。

```
CLASS ARRAY-OBJECT AS "System.Object[]"    *> (1)
CLASS D3ARRAY-INT  AS "System.Int32[,,]"   *> (2)
```

たとえば、“System.Object”型の1次元配列は上記の(1)のように記述します。また、“System.Int32”型の3次元配列は上記の(2)のように記述します。

11.3.4.1.3 型やメソッドが型パラメタを持つことを示すための修飾

型またはメソッドが型パラメタを持つことは、クラス指定子、インタフェース指定子、デリゲート指定子またはメソッド指定子に記述することで指定できます。

型パラメタを持つ型であることを示すには、外部名を型の名前の後ろに鋭角括弧(<>)を付加した文字列として記述します。同じように型パラメタを持つメソッドであることを示すには、外部名をメソッド型の名前の後ろに鋭角括弧(<>)を付加した文字列として記述します。

この外部名は、次の規則に従って記述する必要があります。

a. 型名に続く鋭角括弧の組は、一組しか記述できません。

b. 鋭角括弧の中には、カンマ(,)しか記述できません。(カンマは型パラメタの個数を表し、括弧内にカンマがない場合、型パラメタは1つと解釈されます。)

c. 型名は、必要なら名前空間や入れ子の上位階層の型の名前で修飾する必要があります。

```
CLASS G-List  AS  "System.Collections.Generic.List<>"         *> (1)
CLASS G-Dict  AS  "System.Collections.Generic.Dictionary<,>"  *> (2)
```

たとえば、型パラメタを1つ持つSystem.Collections.Generic.List型は、上記の(1)のように記述します。また、型パラメタを2つ持つSystem.Collections.Generic.Dictionary型は上記の(2)のように記述します。

一方、型パラメタを1つ持つ“System.Collections.Generic.List<>"1次元の配列型は以下の(3)のように、入れ子のクラスInnerClassの上位階層のクラスが型パラメタを2つ持つ場合には(4)のように記述します。

```
CLASS G-List-ARY  AS  "System.Collections.Generic.List<>[]"  *> (3)
CLASS G-IN-CLASS  AS  "OuterClass<,>+InnerClass"             *> (4)
```

11.3.5 正書法

.NETプログラミング機能の正書法は、オブジェクト指向プログラミング機能の正書法と同じです。ここでは、オブジェクト指向プログラミング機能固有の行のつなぎ、名前の有効範囲について説明します。

11.3.5.1 行のつなぎ

呼出し演算子の2つの文字“::”は同じ行になければなりません。

11.3.5.2 名前の範囲

以下に、クラス名およびメソッド名の名前の範囲について説明します。

クラス名の名前の範囲

ソース要素中で使用されるクラスのクラス名は、それ自身またはそのソース要素を含むリポジトリ段落で宣言しなければなりません。

翻訳集団内では、1つのクラス名に対するクラス定義は1つでなければなりません。

ソース要素中のリポジトリ段落で宣言されたクラス名は、そのソース要素および含まれるすべてのソース単位で使用できます。

メソッド名の名前の範囲

メソッドのメソッド名は、メソッド名段落で宣言されます。メソッド名は、INVOKE文、行内呼出しおよびメソッドの終わり見出しでだけ参照できます。

クラス定義またはインタフェース定義で宣言されたメソッドは、オーバロードされる場合を除き、そのクラス定義内またはインタフェース定義内で一意なメソッド名を持たなければなりません。子クラスで宣言されたメソッドは、“11.5.1.5 メソッド名段落（METHOD-ID）”の条件に従い、親クラスのメソッドと同じ名前を持つことができます。

11.4 プログラム構造

ここでは、.NETプログラミング機能固有のプログラム構造について説明します。

11.4.1 プログラムの構成

1つの翻訳集団は、1つ以上のCOBOLソース単位を含むことができます。

原始文操作文および終わり見出しを除いて、ソース単位の文、項、段落および節は、以下の順序で並ぶ4つの部にグループ分けされます。

- 見出し部
- 環境部
- データ部
- 手続き部

ソース単位の部の始まりは、適切な部見出しで示されます。見出し部の見出しは、見出し部で許されている段落の見出しの1つにより示すこともできます。

ソース単位中の部の終わりは、以下のどれかで示されます。

- ソース単位中で次に続く部の始まり
- そのソース単位の終わり見出し
- その後にソースプログラム行がない物理的位置

COBOLソース単位の終わりは、指定されていれば終わり見出し、または翻訳集団にそれ以上のプログラム行がないことにより示されます。他のソース単位に直接または間接的に含まれるソース単位は、含んでいるソース単位の資源を追加参照する別のソース単位であるとみなされます。

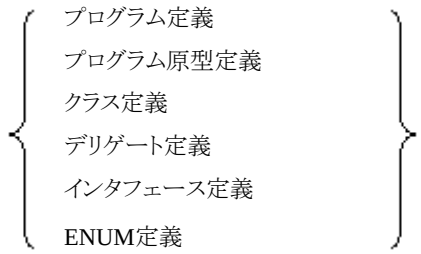
11.4.1.1 COBOL翻訳集団

.NETプログラミング機能におけるCOBOL翻訳集団について説明します。

【書き方】

[{ソース単位} …]

ここで、ソース単位は、



ここで、プログラム定義は、

[IDENTIFICATION DIVISION.]
PROGRAM-ID段落
[環境部]
[データ文]
[手続き部]
[{プログラム定義} …]
[END PROGRAM { プログラム名-1 / プログラム名定数-1 } .]

ここで、プログラム原型定義は、

[IDENTIFICATION DIVISION.]
PROGRAM-ID段落
[環境部]
[データ部]
[手続き部]
END PROGRAM プログラム原型名-1.

ここで、クラス定義は、

[IDENTIFICATION DIVISION.]
CLASS-ID段落
[環境部]
[スタティック定義]
[オブジェクト定義]
END CLASS クラス名-1.

ここで、スタティック定義は、

```
[ [IDENTIFICATION DIVISION.]
  STATIC.
  [環境部]
  [データ部]
  [ [PROCEDURE DIVISION.
  [ [{メソッド定義} … ]
  END STATIC. ]
```

ここで、オブジェクト定義は、

```
[ [IDENTIFICATION DIVISION.]
  OBJECT段落
  [環境部]
  [データ部]
  [ [PROCEDURE DIVISION.
  [ [{メソッド定義} … ]
  END OBJECT. ]
```

ここで、メソッド定義は、

```
[ [IDENTIFICATION DIVISION.]
  METHOD-ID段落
  [環境部]
  [データ部]
  [手続き部]
  END METHOD [メソッド名-1]. ]
```

ここで、デリゲート定義は、

```
[IDENTIFICATION DIVISION.]
DELEGATE-ID段落
[環境部]
[データ部]
[手続き部]
END DELEGATE デリゲート名-1.
```

ここで、インタフェース定義は、

```
[IDENTIFICATION DIVISION.]
INTERFACE-ID段落
C
```

[PROCEDURE DIVISION.]
[[{メソッド定義} …]]

END INTERFACE インタフェース名-1.

ここで、ENUM定義は、

[IDENTIFICATION DIVISION.]
ENUM-ID段落
[環境部]
[データ部]
END ENUM ENUM名-1.

構文規則

1. PROGRAM-ID段落、CLASS-ID段落、OBJECT段落、METHOD-ID段落、DELEGATE-ID段落、STATIC段落、INTERFACE-ID段落、ENUM-ID段落の書き方および規則は、“11.5 見出し部と終わり見出し”で定義されます。
2. プログラム定義は、PROGRAM-ID段落を含む見出し部で始まるソース単位です。プログラム定義は、環境部、データ部および手続き部を含むことができます。プログラム定義は、他のプログラム定義を含むことができます。次のいずれかの場合、プログラム定義の終わりにはプログラム終わり見出しが必要です。
    a. 1つ以上の他のプログラム定義を含むプログラム定義
    b. 他のプログラム定義内に含まれるプログラム定義
3. プログラム原型定義は、PROTOTYPE属性を持つプログラム定義です。プログラム原型定義のデータ部は、連絡節だけを含むことができます。プログラム原型定義の手続き部は、手続き部の見出しだけを含むことができます。プログラム原型定義は、プログラム終わり見出しで終わらなければなりません。
4. クラス定義は、CLASS-ID段落を含む見出し部で始まるソース単位です。クラス定義は、環境部、スタティック定義およびオブジェクト定義を含むことができます。環境部での定義は、そのクラス定義および含まれる定義に適用されます。環境部は、入出力節を含んでいてはなりません。クラス定義は、クラス終わり見出しで終わらなければなりません。
5. スタティック定義は、STATIC段落を含む見出し部で始まるソース単位です。スタティック定義は、環境部、データ部および手続き部を含むことができます。環境部での定義は、そのスタティック定義およびスタティック定義で宣言されるすべてのメソッドに適用されます。環境部は、構成節を含んでいてはなりません。データ部は、基底場所節、ファイル節、作業場所節、定数節だけを含むことができます。手続き部は、メソッド定義だけを含むことができ、ほかの文、節の見出しまたは段落の見出しを含むことはできません。スタティック定義は、スタティック終わり見出しで終わらなければなりません。
6. オブジェクト定義は、OBJECT段落を含む見出し部で始まるソース単位です。オブジェクト定義は、環境部、データ部および手続き部を含むことができます。環境部での定義は、そのオブジェクト定義およびオブジェクト定義で宣言されるすべてのメソッドに適用されます。環境部は、構成節を含んでいてはなりません。データ部は、基底場所節、ファイル節、作業場所節、定数節だけを含むことができます。手続き部は、メソッド定義だけを含むことができ、ほかの文、節の見出しまたは段落の見出しを含むことはできません。オブジェクト定義は、オブジェクト終わり見出しで終わらなければなりません。
7. メソッド定義は、METHOD-ID段落を含む見出し部で始まるソース単位です。メソッド定義は、環境部、データ部および手続き部を含むことができます。環境部は、構成節を含んでいてはなりません。データ部は基底場所節、ファイル節、作業場所節、定数節および連絡節だけを含むことができます。手続き部は、プログラムの手続き部の規則に従います。メソッド定義は、メソッド終わり見出しで終わらなければなりません。
8. デリゲート定義は、DELEGATE-ID段落を含む見出し部で始まるソース単位です。デリゲート定義は、環境部、データ部および手続き部を含むことができます。環境部での定義は、そのデリゲート定義に適用されます。環境部は、

構成節以外を含んでいてはなりません。データ部は、連絡節だけを含むことができます。手続き部は、手続き部見出しだけを含むことができます。デリゲート定義は、デリゲート終わり見出しで終わらなければなりません。

9. インタフェース定義は、INTERFACE-ID段落を含む見出し部で始まるソースの単位です。インタフェース定義の環境部は構成節以外を含んでいてはなりません。インタフェース定義はデータ部を含んでいてはなりません。インタフェース定義の手続き部は、メソッド定義のみを含むことができ、他の文、節の見出しまたは段落の見出しを含むことはできません。インタフェース定義は、インタフェース終わり見出しで終わらなければなりません。
10. ENUM定義は、ENUM-ID段落を含む見出し部で始まるソース単位です。ENUM定義の環境部は構成節以外を含んでいてはなりません。データ部は、作業場所節のみを含むことができます。ENUM定義は手続き部を含んでいてはなりません。ENUM定義は、ENUM終わり見出しで終わらなければなりません。

11.4.2 継承

クラスの継承

クラス定義において、1つの定義は他の1つの定義を継承することができます。継承した定義は、継承された定義のすべてのメソッドを持ちます。これらには、継承された定義が継承したメソッドも含まれます。継承したクラスは、継承されたクラスで定義されたすべてのデータを持ちます。これらには、継承されたクラスが継承したデータも含まれます。継承されたデータは、継承したクラスの各のオブジェクトの一部です。継承された定義で定義されたメソッド名は、継承する定義の利用者語になります。これは、継承する定義でメソッド名を宣言するようなものです。継承された定義の他の利用者語は、継承されません。さらに、この利用者語は、継承された定義で定義されていないのと同じように、継承するクラスで使用することができます。

インタフェースの継承

インタフェース定義において、1つの定義は他の1つ以上の定義を継承することができます。継承した定義は、継承された定義のすべてのメソッドを持ちます。これらには、継承された定義が継承したメソッドも含まれます。

11.4.3 総称性

型パラメタを使用すると、型の決定されていないデータ項目を含むソース要素を定義できます。このソース要素の中で、型パラメタは未決定の型を表現するプレースホルダとして使用できます。

型パラメタを持つソース要素には、外から型パラメタの表現する型を指定できます。このとき、型パラメタを含むソース要素は、型パラメタの指定がすべて指定された型で置き換えられたかのようにふるまいます。型パラメタの表現する型を決定するために外部から指定する型を、型引数と呼びます。型パラメタの持つ制約という条件を満たせば、型引数に任意の型を指定できます。

このように、同一のソース要素の中で使用する型を後から指定可能とすることで、ソース要素を再利用する機構を総称性(ジェネリック)と呼びます。

総称型

1つ以上の型パラメタを持つ型のことです。次のどれかが総称型になります。

- クラス
- インタフェース
- デリゲート

総称型を定義するソース要素の中で、型パラメタは次のような目的で使用できます。

- スタティック/オブジェクトやデータ項目の型を指定する
- メソッドのパラメタや戻り値の型を指定する
- メソッドのローカルなデータ項目の型を指定する
- 他の型パラメタの制約や型引数に指定する

総称型は、型が未決定である要素を含むため、そのままでは通常の型のようには使用できません。総称型に適切な型引数を指定すれば、総称型を雛型として、その型パラメタを型引数の型で置き換えた型が仮想的に作成されます。これを構成型と呼びます。構成型は、通常の型と同じように使用できます。

総称メソッド

1つ以上の型パラメタを持つメソッドです。型パラメタはメソッド定義内で、次のような目的で使用できます。

- メソッドのパラメタや戻り値の型を指定する
- メソッドのローカルなデータ項目の型を指定する
- 他の型パラメタの制約や型引数に指定する

総称メソッドは、型が未決定である要素を含むため、そのままでは通常のメソッドのようには使用できません。総称メソッドに適切な型引数を指定すれば、総称メソッドを雛型として、その型パラメタを型引数の型で置き換えたメソッドが仮想的に作成されます。このメソッドは通常のメソッドと同じように使用できます。

総称メソッドへの型引数の指定は、総称型へ型引数を指定する場合と同じように明示的に指定できます。また、次の条件に該当すれば、メソッド呼び出し時に指定された実パラメタの並びから、型引数の型を推測し、暗黙的に型引数に渡すことができます。

1. 型パラメタが仮パラメタの型として使用されている
2. 実パラメタの並びに、COBOL独自型の実パラメタが含まれない
3. 実パラメタの並びから型引数の型が一意に決定できる
4. メソッドを束縛できる

11.5 見出し部と終わり見出し

ここでは、.NETプログラミング機能で使用する見出し部および終わり見出しを説明します。

11.5.1 見出し部の構成

ここでは、.NETプログラミング機能における、見出し部の構成について説明します。

見出し部は、プログラム定義、プログラム原型定義、クラス定義、スタティック定義、オブジェクト定義、メソッド定義、デリゲート定義、インタフェース定義およびENUM定義を識別します。

【書き方】

[IDENTIFICATION DIVISION.]

{ PROGRAM-ID段落 / CLASS-ID段落 / STATIC段落 / OBJECT段落 / METHOD-ID段落 / DELEGATE-ID段落 / INTERFACE-ID段落 / ENUM-ID段落 }

11.5.1.1 プログラム名段落（PROGRAM-ID）

PROGRAM-ID段落は、この見出し部がプログラム定義を始めることを示し、プログラムを識別する名前を指定し、プログラム属性をプログラムに割り当てます。

【書き方1】

```
PROGRAM-ID.  { プログラム名-1 [AS 定数-1]   }
             { プログラム名定数             }
  [ IS { | COMMON   | } PROGRAM ]
       { | INITIAL  | }
  [CUSTOM-ATTRIBUTE句].
```

【書き方2】

```
PROGRAM-ID. プログラム名-2 [AS 定数-1]
    IS PROTOTYPE
    CUSTOM-ATTRIBUTE句.
```

構文規則

書き方1および書き方2に共通する規則

定数-1は、文字定数または日本語定数でなければなりません。ただし、16進文字定数や日本語16進文字定数であってはなりません。

書き方1の規則

1. 定数-1はこのプログラムが他のプログラムに含まれる場合には指定できません。
2. COMMON句はこのプログラムが他のプログラムに含まれない場合、指定できません。

書き方2の規則

CUSTOM-ATTRIBUTE句には、DLLIMPORT属性を定義するカスタム属性名を指定しなければなりません。

一般規則

書き方1の規則

1. プログラム名-1は、このプログラム定義で宣言されたプログラムに名前を付けます。定数-1が指定されている場合、それがプログラムの外部名になり、プログラム名-1が内部名になります。定数-1が指定されなかった場合、プログラム名-1が内部名および外部名になります。
2. INITIAL句が指定された場合、プログラムに初期化属性が与えられます。初期化プログラム(初期化属性を持つプログラム)が呼び出されると、そのプログラムは初期状態になります。
3. COMMON句が指定された場合、プログラムに共通属性が与えられます。共通プログラム(共通属性を持つプログラム)は、共通プログラムを含むプログラム以外からも呼び出すことができます。

書き方2の規則

1. プログラム名-2は、プログラム原型名を定義します。プログラム原型名は、プログラムのインタフェースを識別する名前であり、この定義と同じインタフェースを持つ非管理プログラムコードの呼出しに使用します。
2. プログラム原型名と、それによって識別されるプログラムの実体は、DLLIMPORT属性で定義します。DLLIMPORT属性のPROPERTY指定で、EntryPointを指定しなかった場合、呼び出されるプログラムの名前はプログラム原型名と等しいものとみなされます。
3. プログラム原型定義のデータ部は、連絡節だけを含むことができます。
4. プログラム原型定義の手続き部は、手続き部の見出しだけを含むことができます。

11.5.1.2 クラス名段落（CLASS-ID）

CLASS-ID段落は、この見出し部がクラス定義を始めることを示します。また、クラスを識別する名前を指定し、クラス属性をクラスに割り当てます。

【書き方】

CLASS-ID. クラス名-1 [AS 定数-1]

[IS PARTIAL]

[INHERITS クラス名-2]

[CUSTOM-ATTRIBUTE句].

構文規則

1. 定数-1は、文字定数または日本語定数でなければなりません。ただし、16進文字定数や日本語16進文字定数であってはなりません。
2. クラス名-2は、このソース要素のリポジトリ段落で指定されたクラスの名前でなければなりません。
3. クラス名-2は、このクラス定義で宣言されたクラスと同じ名前であってはなりません。
4. クラス名-2は、クラス名-1を直接または間接的に継承してはなりません。
5. クラス名-2は、最終クラス(sealedクラス(finalクラス))や非公開クラス(privateクラス)であってはなりません。(※sealedクラスやprivateクラスは、COBOLでは定義できません。)
6. クラス名.ASSEMBLYは、アセンブリに対するカスタムアトリビュートであるため、コンパイラが予約します。このため、定数-1に.ASSEMBLYが指定された場合、上記の規則に加えて、次の規則が適用されます。
    a. アセンブリに対するカスタムアトリビュートのクラス名段落には、クラス名のほかにはCUSTOM-ATTRIBUTE句だけが記述できます。
    b. アセンブリに対するカスタムアトリビュートのクラスでは、クラス名段落のほかには環境部の構成節だけが記述できます。
7. クラス名.MODULEは、モジュールに対するカスタムアトリビュートのため、コンパイラが予約します。このため、定数-1に.MODULEが指定された場合、上記の規則に加えて、次の規則が適用されます。
    a. モジュールに対するカスタムアトリビュートのクラス名段落には、クラス名のほかにはCUSTOM-ATTRIBUTE句だけが記述できます。
    b. モジュールに対するカスタムアトリビュートのクラスでは、クラス名段落のほかには環境部の構成節だけが記述できます。

一般規則

1. クラス名-1は、このクラス定義で宣言されたクラスに名前を付けます。定数-1が指定されている場合、それがクラスの外部名になり、クラス名-1が内部名になります。定数-1が指定されなかった場合、クラス名-1が内部名および外部名になります。
2. INHERITS句は、継承の規則に従い、クラス名-1により継承されるクラスの名前を指定します。
3. INHERITS句が省略された場合、System.Objectクラスが暗黙のうちに継承されます。
4. クラス名-2が抽象クラスであった場合、クラス名-2に含まれるすべての抽象メソッドおよび抽象プロパティを、このクラス定義中で再定義しなければなりません。
5. PARTIAL句は、このクラスを部分型形式として扱うことを指定します。
6. 定数-1に、.ASSEMBLYまたは.MODULEが指定されたクラスは、継承および参照することはできません。
7. 定数-1に.ASSEMBLYが指定されたクラスは、アセンブリに付与するカスタム属性を指定するために使用します。

8. 定数-1に.MODULEが指定されたクラスは、モジュールに付与するカスタム属性を指定するために使用します。

11.5.1.3 スタティック段落（STATIC）

STATIC段落は、この見出し部からスタティック定義を始めることを示します。

【書き方】

STATIC.

構文規則

クラス定義中に複数のスタティック定義が書かれていてはなりません。

11.5.1.4 オブジェクト段落（OBJECT）

OBJECT段落は、この見出し部からオブジェクト定義を始めることを示します。

【書き方】

OBJECT. [IMPLEMENTS {インタフェース名-1} …].

構文規則

1. インタフェース名-1は、このソース要素のクラス定義のリポジトリ段落で指定されたインタフェースの名前でなければなりません。
2. クラス定義中に複数のオブジェクト定義が書かれていてはなりません。また、クラス定義中にスタティック定義が書かれている場合、スタティック定義に続けて書かれなければなりません。

一般規則

IMPLEMENTS句は、このオブジェクトがインタフェース-1によって指定されたインタフェースを実装していることの保証を与えます。なぜなら、インタフェース名-1に含まれるすべてのメソッドおよびプロパティは、このオブジェクト定義内で明または暗に実装されるからです。

11.5.1.5 メソッド名段落（METHOD-ID）

METHOD-ID段落は、この見出し部からメソッド定義を始めることを示します。また、METHOD-ID段落は、メソッドを識別する名前を指定し、メソッドにメソッド属性を割り当てます。

【書き方】

METHOD-ID.

```
{ メソッド名-1 [AS 定数-1]
{ { GET } PROPERTY プロパティ名-1 [AS 定数-1] }
{ { SET } }
[ { { OF インタフェース名-1 } ] ]
[ { { OVERRIDE } } ]
```

```
[    IS  { PUBLIC    }  ]
         { PRIVATE   }
         { PROTECTED }
```

[CUSTOM-ATTRIBUTE句].

構文規則

1. 定数-1は、文字定数または日本語定数でなければなりません。ただし、16進文字定数や日本語16進文字定数であってはなりません。
2. OVERRIDE指定は、オブジェクト定義中のメソッド定義にだけ指定できます。
3. OVERRIDE指定が書かれた場合、このメソッド定義で宣言されたメソッドと同じシグネチャを持つ継承可能なメソッドが、親クラスで定義されていなければなりません。シグネチャが一致するかどうかは、次の規則によって判定されます。
    a. メソッド名（外部名）が一致するか。
    b. 仮引数(COBOLでは、手続き部見出しのUSING指定で記述)の個数が同じか。
    c. 同じ順序位置にある仮引数の受取り方(COBOLでは、手続き部見出しのUSING BY指定で記述)が同じか。たとえば、このメソッドの引数にBY VALUEが指定されているなら、対応する引数は値渡しでなければなりません。あるいは、このメソッドの引数に明または暗にBY REFERENCEが指定されているなら、対応する引数は参照渡しでなければなりません。
    d. 同じ順序位置にある仮引数のデータ属性が同じか。
    CLR型なら、それらは同じ型でなければなりません。COBOL独自型なら、項類、項目長、用途、符号、PICTURE句の編集文字の並びが一致し、かつ、BLANK WHEN ZEROおよびJUSTIFIEDの各句の指定が一致していなければなりません。
    e. 復帰項目(COBOLでは、手続き部見出しのRETURNING指定で記述)の有無およびデータ属性が同じか。データ属性が同じかどうかの判定基準は上述のとおりです。ただし、復帰項目が無いことと、復帰項目がvoid型（C#やC++のキーワード）であることは同義とします。
4. さらに、OVERRIDEに関しては以下の規則が適用されます。
    a. 仮想属性を持たないメソッドをOVERRIDEすることはできません。
    b. パラメタ配列(System.ParamArrayAttribute属性を持つパラメタ)をパラメタに持つメソッドをOVERRIDEすることはできません。
5. OVERRIDEするメソッドとOVERRIDEされるメソッドのアクセス属性は同じでなければなりません。
6. OF指定はオブジェクト定義中のメソッド定義にだけ指定できます。
7. OF指定が書かれた場合、このメソッド定義で宣言されたメソッドと同じシグネチャを持つメソッドが、OF指定で指定したインタフェース内で定義されていなければなりません。シグネチャが一致するかどうかの判定規則は、構文規則 3)と同じです。
8. プロパティ名-1は、スタティック定義またはオブジェクト定義の作業場所節の01または77レベル項目として定義したデータ名と同じ名前であってはなりません。
9. GET指定が書かれた場合、メソッドは、手続き部の見出しにUSING指定があってはなりません。また、RETURNING指定を持たなければなりません。
10. SET指定が書かれた場合、メソッドの手続き部の見出しには、BY VALUE指定のUSING引数が1つ指定されなければなりません。また、RETURNING指定があってはなりません。
11. 同じプロパティ名-1のためのプロパティ取出しメソッドとプロパティ設定メソッドの両方が自クラス内で明または暗に定義される場合、それらは以下の点ですべて一致していなければなりません。
    ー static属性の有無
    ー アクセス属性

- 取出しメソッドのRETURNING引数の属性と、設定メソッドのUSING引数の属性

12. メソッド名NEWは、コンストラクタのためにコンパイラが予約します。このため、メソッド名としてNEWを指定した場合、上記の規則に加えて、以下の規則が適用されます。

    a. コンストラクタをスタティックメソッドに定義した場合、そのコンストラクタはスタティックコンストラクタになります。

    b. コンストラクタをオブジェクトメソッドに定義した場合、そのコンストラクタはインスタンスコンストラクタになります。

    c. スタティックコンストラクタのメソッド名段落には、メソッド名のほかにはCUSTOM-ATTRIBUTE句だけが指定できます。

    d. インスタンスコンストラクタのメソッド名段落には、メソッド名のほかには以下の指定だけできます。

        - CUSTOM-ATTRIBUTE句

        - アクセス属性指示

13. 以下に示すものは予約されたメソッド名であり、いかなるメソッド定義にもこれらを外部名として指定することはできません。

    - .ctor

    - .cctor

    .cctorは、スタティックコンストラクタ.COBOLソース内では、NEWメソッドとして宣言します。スタティックコンストラクタは、クラスがロードされる際にCLRが自動的に呼び出すため、参照できません。

    また、以下のプレフィクスを持つメソッドは、同名のものをコンパイラが自動生成する可能性があるため、利用者語としての使用はしないでください。

    - get_

    - set_

    - add_

    - remove_

    - raise_

    - op_

    - Item

14. インタフェースを実装するメソッドのアクセス属性は、PUBLICでなければなりません。ただし、OF指定によってインタフェース名を明に指定して実装する場合、このメソッドのアクセス属性はPRIVATEでなければなりません。

一般規則

1. このメソッド定義がオブジェクト定義に含まれる場合、このメソッド定義により宣言されるメソッドまたはプロパティは、仮想属性を持ちます。また、このメソッド定義がスタティック定義に含まれる場合、このメソッド定義により宣言されるメソッドまたはプロパティは、スタティック属性を持ちます。

2. メソッド名-1は、このメソッド定義で宣言されるメソッドに名前を付けます。定数-1が指定されている場合、それがメソッドの外部名になり、メソッド名-1が内部名になります。定数-1が指定されなかった場合、メソッド名-1が内部名および外部名になります。

3. プロパティ名-1は、このメソッド定義で宣言されるプロパティに名前を付けます。定数-1が指定されている場合、それがプロパティの外部名になり、プロパティ名-1が内部名になります。定数-1が指定されなかった場合、プロパティ名-1が内部名および外部名になります。

4. OVERRIDE指定は、このメソッドが継承された仮想属性を持つメソッドを置き換えることを示します。スタティックメソッドにOVERRIDE指定を書くことはできませんが、同じシグネチャを持つスタティックメソッドが継承されているなら、それを自動的に置き換えます。

5. メソッドの外部名は、このメソッドを含むオブジェクトを参照するオブジェクト一意名を伴うメソッド呼出しで使用できます。

6. プロパティの外部名は、このメソッドを含むオブジェクトを参照するオブジェクト一意名を伴うオブジェクトプロパティで使用できます。ただしプロパティが内部名を持つならば、オブジェクトプロパティで参照できるのは内部名です。
7. 利用者語が、このメソッド名定義のデータ部、および、それを含むスタティック定義またはオブジェクト定義で定義された場合、このメソッド中でその語を使用すると、このメソッド中の宣言を参照します。含んでいるスタティック定義またはオブジェクト定義内での宣言は、このメソッドから修飾なしでアクセスできません。
8. GET指定が書かれた場合、このメソッドは、プロパティ名-1または定数-1で示されるプロパティの、プロパティ取出しメソッドです。
9. SET指定が書かれた場合、このメソッドは、プロパティ名-1または定数-1で示されるプロパティの、プロパティ設定メソッドです。
10. OF指定は、このメソッド定義が実装したメソッド定義を含むインタフェース定義を明示的に指定します。複数のインタフェースを実装するクラスのメソッドの定義で、インタフェース名指定を省略した場合、そのメソッド定義はインタフェースから引き継いだ同じ名前と引数インタフェースを持つメソッドを実装するものとみなされます。
11. PUBLIC、PRIVATE、PROTECTEDは、メソッドのアクセス属性を指定します。省略した場合は、PUBLICが指定されたものとみなします。
12. RETURNING項目を除くシグネチャが一致しないならば、同じ名前を持つメソッドを複数定義することができます。これらのメソッドはどれも同じ名前で呼び出しますが、実際に呼び出されるメソッドはUSING引数の個数や型(実引数の形式)によって決まります。これをメソッドのオーバロードといいます。

例

オーバロードの例

```
*> (a) 引数なしのMETHOD1の宣言
 METHOD-ID. METHOD1.
 PROCEDURE  DIVISION.
*>      :
 END METHOD METHOD1.

*> (b) 引数を1個持つMETHOD1の宣言
 METHOD-ID. METHOD1.                    *>…[1]
 DATA       DIVISION.
 LINKAGE    SECTION.
 01 P1      PIC S9(9) COMP-5.
 PROCEDURE DIVISION USING BY VALUE P1.
*>           :
 END METHOD METHOD1.
*>
 METHOD-ID. MAIN.
 PROCEDURE  DIVISION.
  *> 引数なしのMETHOD1を呼び出します。
     INVOKE SELF "METHOD1"              *>…[2]
 END METHOD MAIN.
```

[1](b)のMETHOD1は、(a)のMETHOD1と異なるシグネチャを持つので、オーバロードが成立し、宣言することができます。
[2]INVOKE文に引数(USING引数)が指定されていないので、引数を持たない方の(a)のMETHOD1が選択され、呼び出されます。

13. COBOL独自型は、オーバロード時のメソッドシグネチャの識別においては、すべて同じ型とみなされます。したがって、COBOL独自型引数の内容の違いだけでメソッドをオーバロードすることはできません。
14. 引数がCLR型の場合、引数の受渡し方法の違いだけでメソッドをオーバロードすることができます。
15. プロパティメソッドはオーバロードできません。
16. スタティックコンストラクタはオーバロードできません。

11.5.1.6 デリゲート名段落（DELEGATE-ID）

DELEGATE-ID段落は、見出し部がデリゲート定義を始めることを示し、デリゲートを識別する名前を指定し、デリゲート属性をデリゲートに割り当てます。

【書き方】

DELEGATE-ID. デリゲート名-1 [AS 定数-1].

構文規則

定数-1は、文字定数または日本語定数でなければなりません。ただし、16進文字定数や日本語16進文字定数であってはなりません。

一般規則

1. デリゲート名-1は、このデリゲート定義で宣言されたデリゲートに名前を付けます。定数-1が指定されている場合、それがデリゲートの外部名になり、デリゲート名-1が内部名になります。定数-1が指定されなかった場合、デリゲート名-1が内部名および外部名になります。
2. デリゲート定義のデータ部は連絡節だけを含むことができます。
3. デリゲート定義の手続き部は、手続き部の見出しだけを含むことができます。

11.5.1.7 インタフェース名段落（INTERFACE-ID）

INTERFACE-ID段落は、見出し部がインタフェース定義を始めることを示し、インタフェースを識別する名前を指定し、インタフェース属性をインタフェースに割り当てます。

【書き方】

INTERFACE-ID. インタフェース名-1 [AS 定数-1]
[INHERITS {インタフェース名-2} …]
[CUSTOM-ATTRIBUTE句]
[IS PARTIAL].

構文規則

1. 定数-1は、文字定数または日本語定数でなければなりません。ただし、16進文字定数や日本語16進文字定数であってはなりません。
2. インタフェース名-2はこのソース要素のリポジトリ段落で指定されたインタフェースの名前でなければなりません。
3. インタフェース名-2は、インタフェース名-1を直接または間接的に継承してはなりません。

一般規則

1. インタフェース名-1は、このインタフェース定義で宣言されるインタフェースに名前を付けます。定数-1が指定されている場合、それがインタフェースの外部名になります。
2. INHERITS句は、インタフェースの継承規則に従い、インタフェース名-1により継承されるインタフェースの名前を指定します。
3. PARTIAL句は、このインタフェースを部分型形式として扱うことを指定します。

11.5.1.8 ENUM名段落（ENUM-ID）

ENUM-ID段落は、見出し部がENUM定義を始めることを示し、ENUMを識別する名前を指定し、ENUM属性をENUMに割り当てます。

【書き方】

ENUM-ID. ENUM名-1 [AS 定数-1]
[INHERITS クラス名-1]
[CUSTOM-ATTRIBUTE句].

構文規則

1. 定数-1は、文字定数または日本語定数でなければなりません。ただし16進文字定数や日本語16進文字定数であってはなりません。
2. クラス名-1は、このソース要素のリポジトリ段落で指定された以下のクラスでなければなりません。
   - System.Byte(byte)
   - System.Int16(short)
   - System.Int32(int)
   - System.Int64(long)

一般規則

1. ENUM名-1は、このENUM定義で宣言されるENUMに名前を付けます。定数-1が指定されている場合、それがENUMの外部名になります。
2. INHERITS句は、ENUMの要素の型を指定します。
3. INHERITS句が省略された場合、System.Int32クラスが指定されたものとみなします。
4. ENUM定義に手続き部を記述してはなりません。

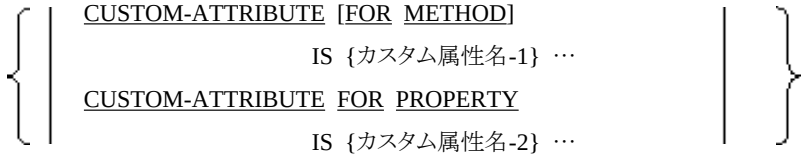
11.5.1.9 CUSTOM-ATTRIBUTE句

CUSTOM-ATTRIBUTE句は、この句を指定した定義に、カスタム属性を与える機能を提供します。

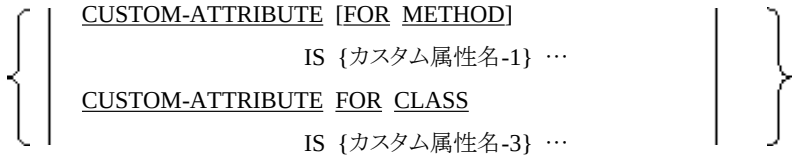
【書き方1】

CUSTOM-ATTRIBUTE IS {カスタム属性名-1} …

【書き方2】



【書き方3】



構文規則

1. CUSTOM-ATTRIBUTE句は、以下の定義にだけ指定することができます。
   - CLASS-ID段落
   - METHOD-ID段落
   - PROGRAM-ID段落（最外部のプログラムのみ）
   - INTERFACE-ID段落
   - ENUM-ID段落
   - スタティック定義およびオブジェクト定義の作業場所節で宣言されるCLR型項目
   - 連絡節のデータ定義
2. カスタム属性名-1を定義するアトリビュートクラスが、同一定義への複数指定を許さないものである場合、同じアトリビュートクラスで定義したカスタム属性名を複数指定することはできません。
3. カスタム属性名-1は、この句を指定した定義に指定可能な属性を持つアトリビュートクラスによって定義したものでなければなりません。
4. 書き方2は、GET指定またはSET指定のあるメソッド名段落でだけ記述できます。
5. 同じプロパティのプロパティ設定メソッドとプロパティ取得メソッドを両方定義する場合、FOR PROPERTYの指定を含むCUSTOM-ATTRIBUTE句は、どちらにも記述しないか、あるいは両方に同じカスタム属性名を記述しなければなりません。
6. 書き方3は、PROTOTYPE句を含まないプログラム名段落でだけ記述できます。

一般規則

書き方1および書き方2に共通する規則

CUSTOM-ATTRIBUTE句は、この句を指定した定義に、カスタム属性を与える機能を提供します。

書き方2の規則

1. FOR指定のないCUSTOM-ATTRIBUTE句、またはFOR METHOD指定を含むCUSTOM-ATTRIBUTE句は、メソッドに付与するカスタム属性を指定するために使用します。
2. FOR PROPERTYの指定を含むCUSTOM-ATTRIBUTE句は、プロパティに付与するカスタム属性を指定するために使用します。

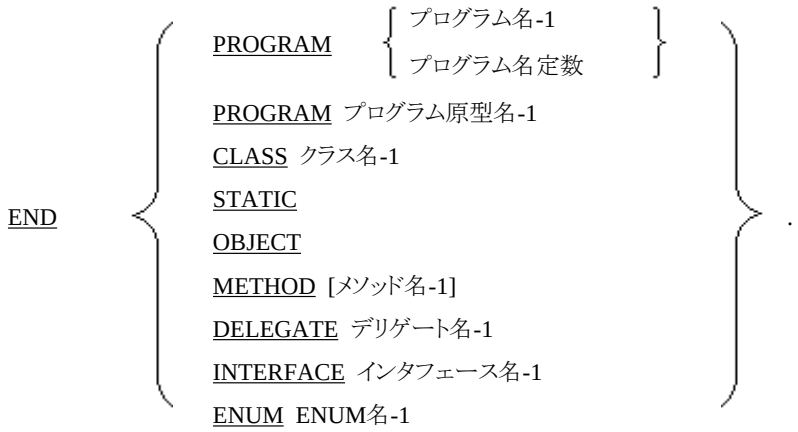
書き方3の規則

1. FOR指定のないCUSTOM-ATTRIBUTE句、またはFOR METHOD指定を含むCUSTOM-ATTRIBUTE句は、メソッドに付与するカスタム属性を指定するために使用します。
2. FOR CLASSの指定を含むCUSTOM-ATTRIBUTE句は、クラスに付与するカスタム属性を指定するために使用します。

11.5.2 終わり見出し

ここでは、.NETプログラミング機能における、終わり見出しについて説明します。

【書き方】



構文規則

1. 終わり見出しは、ほかのソース単位を含むソース単位、ほかのソース単位に含まれるソース単位、または、ほかのソース単位が続くソース単位に書かなければなりません。
2. プログラム名-1は、対応するPROGRAM-ID段落で指定したプログラム名と同じでなければなりません。
3. プログラム原型名-1は、対応するPROGRAM-ID段落で指定したプログラム原型名と同じでなければなりません。
4. クラス名-1は、対応するCLASS-ID段落で指定したクラス名と同じでなければなりません。
5. メソッド名-1は、対応するMETHOD-ID段落で指定したメソッド名と同じでなければなりません。METHOD-ID段落にPROTOTYPE指定を書いた場合、メソッド名-1は省略しなければなりません。
6. デリゲート名-1は、対応するDELEGATE-ID段落で指定したデリゲート名と同じでなければなりません。
7. インタフェース名-1は、対応するINTERFACE-ID段落で指定したインタフェース名と同じでなければなりません。
8. ENUM名-1は、対応するENUM-ID段落で指定したENUM名と同じでなければなりません。

一般規則

1. 終わり見出しは、指定されたソース単位の終わりを示します。
2. 終わり見出しで終わるソース単位が、ほかのソース単位に含まれる場合、次の文は、ソース単位の最初の文、または含んでいるソース単位を終了させる終わり見出しでなければなりません。
3. 終わり見出しで終わるプログラムがほかのソース単位に含まれていない場合、ソース記述が続くなら、次の文は、別に翻訳されるソース単位の最初の文でなければなりません。

11.6 環境部

ここでは、.NETプログラミング機能固有の構成節および入出力節について説明します。

11.6.1 構成節（CONFIGURATION SECTION）

ここでは、.NETプログラミング機能における、構成節について説明します。

その他の規則については、“4.2 構成節（CONFIGURATION SECTION）”を参照してください。

【書き方】

CONFIGURATION SECTION.

[翻訳用計算機段落]

[実行用計算機段落]

[特殊名段落]

[リポジトリ段落]

構文規則

1. 構成節は、プログラム定義、プログラム原型定義、クラス定義、デリゲート定義、インタフェース定義またはENUM定義の中に指定できますが、スタティック定義、オブジェクト定義およびメソッド定義の中には指定できません。
2. 翻訳用計算機段落および実行用計算機段落は、クラス定義、デリゲート定義、インタフェース定義またはENUM定義の構成節に指定できません。

11.6.1.1 特殊名段落（SPECIAL-NAMES）

SPECIAL-NAMES段落では、通常のCOBOLプログラミングの場合と同様に、記号定数、記号文字、符号系名、文字集合、文字の大小順序などを定義したり、機能名と呼び名を対応付けたりします。.NETプログラミング機能では、これに加えてカスタム属性名を定義することができます。

【書き方】

SPECIAL-NAMES.

[{機能名-1句} …]

[{機能名-2句} …]

[{機能名-3句} …]

[{ALPHABET句} …]

[{CLASS句} …]

[CURRENCY SIGN句]

[{CUSTOM-ATTRIBUTE句} …]

[DECIMAL-POINT IS COMMA句]

[POSITIONING UNIT句]

[PRINTING MODE句]

[{SYMBOLIC CHARACTERS句} …]

[{SYMBOLIC CONSTANT句} …]

.

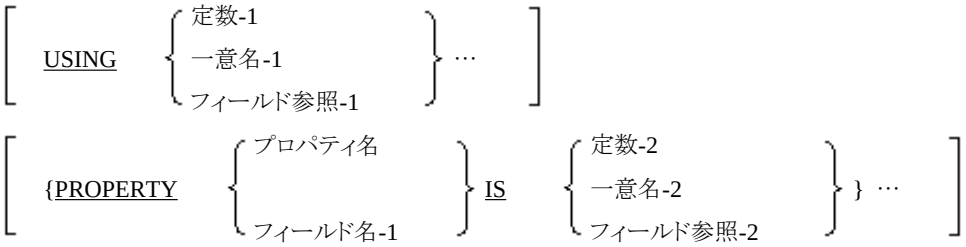
11.6.1.2 CUSTOM-ATTRIBUTE句

CUSTOM-ATTRIBUTE句は、アトリビュートクラスからカスタム属性名を定義します。

【書き方】

CUSTOM-ATTRIBUTE カスタム属性名-1

CLASS クラス名-1



構文規則

1. 一意名-1および一意名-2は、フィールド参照、関数一意名、またはTYPE OF特殊レジスタのいずれかでなければなりません。
2. クラス名-1は、アトリビュートクラス(System.Attributeを継承したクラス)のクラス名でなければなりません。
3. クラス名-1は、このソース要素のリポジトリ段落で指定されたクラスの名前でなければなりません。
4. プロパティ名-1は、クラス名-1に指定したクラスのプロパティ名でなければなりません。
5. フィールド名-1は、クラス名-1に指定したクラスのフィールド名でなければなりません。
6. フィールド参照-1およびフィールド参照-2は、定数を示すフィールド参照でなければなりません。
7. 関数一意名は、以下の関数でなければなりません。
   - FUNCTION ENUM-OR
   - FUNCTION ENUM-AND
   - FUNCTION ENUM-NOT

一般規則

1. 定数-1、フィールド参照-1および関数一意名-1は、クラス名-1に指定されたクラスのコンストラクタの引数に適合していなければなりません。
2. 定数-2、フィールド参照-2および関数一意名-2は、それに対応したフィールド名-1およびプロパティ名-1の型に一致しなければなりません。
3. 定数-1、フィールド参照-1および関数一意名-1により束縛される、クラス名-1のクラスのコンストラクタは、保持する引数の型が、次のいずれかでなければなりません。
   - uint8
   - int16
   - int32
   - int64
   - float32
   - float64
   - char
   - boolean
   - object
   - string
   - ENUM型
   - type

4. フィールド名-1またはプロパティ名-1の型は、次のいずれかでなければなりません。

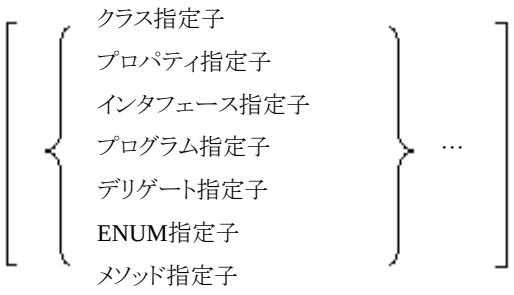
   - uint8
   - int16
   - int32
   - int64
   - float32
   - float64
   - char
   - boolean
   - object
   - string
   - ENUM型
   - type

11.6.1.3 リポジトリ段落（REPOSITORY）

リポジトリ段落は、翻訳単位内で使われるクラス名、プロパティ名、フィールド名、インタフェース名、プログラム原型名、デリゲート名、ENUM名およびメソッド識別名を指定します。

【書き方】

REPOSITORY.



ここで、クラス指定子は、

CLASS クラス名-1 [AS 定数-1]
　　[EXPANDS クラス名-2
　　USING指定]

ここで、プロパティ指定子は、

PROPERTY { プロパティ名-1 / フィールド名-1 } [AS 定数-2]

ここで、インタフェース指定子は、

INTERFACE インタフェース名-1 [AS 定数-3]
[EXPANDS インタフェース名-2
USING指定]

ここで、プログラム指定子は、

PROGRAM プログラム原型名-1 [AS 定数-4]

ここで、デリゲート指定子は、

DELEGATE デリゲート名-1 [AS 定数-5]
[EXPANDS デリゲート名-2
USING指定]

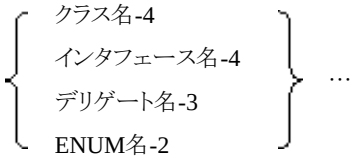
ここで、ENUM指定子は、

ENUM ENUM名-1 [AS 定数-6]

ここで、メソッド指定子は、

METHOD メソッド識別名-1 [AS 定数-7]
[EXPANDS メソッド識別名-2
USING指定]

ここで、USING指定は、



構文規則

1. クラス名-1、プロパティ名-1、インタフェース名-1、プログラム原型名-1、デリゲート名-1、ENUM名-1およびメソッド識別名-1は、リポジトリ段落で2度以上指定されてはなりません。
2. 定数-1、定数-2、定数-3、定数-4、定数-5、定数-6および定数-7は、文字定数または日本語定数でなければなりません。ただし、16進文字定数や日本語16進文字定数であってはなりません。
3. 定数-1、定数-2、定数-3、定数-4、定数-5および定数-6は、アスタリスク(*)で始まる文字列であってはなりません。
4. リポジトリ段落の各指定子で指定する名前は、そのリポジトリ段落を含むソース単位の名前であってはなりません。
5. クラス指定子で配列型を示すことができます。配列型であることを示すには、型の名前の後ろに角括弧([])を付加した文字列を定数-1に記述します。
詳細な規則については“11.3.4.1 外部名の表記の規則”を参照してください。

6. クラス指定子、インタフェース指定子、デリゲート指定子またはメソッド指定子で総称性を持つクラス、インタフェース、デリゲートまたはメソッドを指定する場合、型またはメソッドの名前の後ろに鋭角括弧(<>)を付加した文字列を定数-1に記述します。
   詳細な規則については“11.3.4.1 外部名の表記の規則”を参照してください。

7. AS指定とEXPANDS指定は同時には指定できません。

クラス指定子固有の規則

1. クラス名-2、クラス名-4、インタフェース名-4、デリゲート名-3およびENUM名-2は、同じリポジトリ段落で宣言した名前でなければなりません。

2. クラス名-2は、型パラメタを持つクラスを指定する名前でなければなりません。

3. USING指定に指定する名前の個数は、EXPANDS指定に記述したクラス名-2で指定されるクラスの持つ型パラメタの個数と一致していなければなりません。

プロパティ指定子固有の規則

1. プロパティ指定子は、この構成節の有効範囲内でプロパティあるいはフィールドを参照する際の名前を宣言します。プロパティおよびフィールドは、その実体を定義する名前と別の名前を付けて参照することが可能であり、その場合、プロパティ指定子で宣言した名前は、この構成節の有効範囲内でだけ有効な別名となります。このプロパティ指定子で宣言した名前でプロパティまたはフィールドを参照する場合、以下の規則にしたがわなければなりません。

   a. この構成節の有効範囲内で定義するプロパティまたはフィールドは、このプロパティ指定子の宣言にしたがって定義する名前と、参照する名前の組合せが、ただ1通りとなるように定義しなければなりません。

例

```
REPORITORY.
PROPERTY P1
PROPERTY P2 AS "P".                  … [a]
 :
01  P1  PIC N(1) PROPERTY.          … ○
01  P2  PIC N(1) PROPERTY.          … × [1]
01  P   PIC N(1) PROPERTY.          … ○
   :
METHOD-ID. GET PROPERTY P1.         … ○
   :
METHOD-ID. GET PROPERTY P2.         … × [2]
   :
METHOD-ID. GET PROPERTY P2 AS "P".  … ○ [3]
```

[a]は、Pという名前で定義したプロパティまたはフィールドを、P2いう名前で参照するという宣言です。
[1]および[2]は、P2という名前で定義したプロパティをP2という名前で参照する宣言ですので、翻訳エラーになります。
[3]は、Pという名前で定義したプロパティを、P2という名前で参照する宣言であり、[a]の宣言と矛盾しないため翻訳エラーになりません。

   b. 定数-2が指定されない場合、この構成節を含むソース単位、この構成節のリポジトリ段落での宣言から明または暗に参照可能なクラスまたはインタフェースのいずれかでプロパティまたはフィールドがこのプロパティ指定子に指定した名前で宣言されていることを示します。

   c. 定数-2が指定された場合、この構成節を含むソース単位、この構成節のリポジトリ段落での宣言から明または暗に参照可能なクラスまたはインタフェースのいずれかでプロパティまたはフィールドがこのプロパティ指定子の定数-2で指定した名前で宣言されていることを示します。

インタフェース指定子固有の規則

1. インタフェース名-2、クラス名-4、インタフェース名-4、デリゲート名-3およびENUM名-2は、同じリポジトリ段落で宣言した名前でなければなりません。

2. インタフェース名-2は、型パラメタを持つインタフェースを指定する名前でなければなりません。

3. USING指定に指定する名前の個数は、EXPANDS指定に記述したインタフェース名-2で指定されるインタフェースの持つ型パラメタの個数と、一致していなければなりません。

デリゲート指定子固有の規則

1. デリゲート名-2、クラス名-4、インタフェース名-4、デリゲート名-3およびENUM名-2は、同じリポジトリ段落で宣言した名前でなければなりません。
2. デリゲート名-2は、型パラメタを持つデリゲートを指定する名前でなければなりません。
3. USING指定に指定する名前の個数は、EXPANDS指定に記述したデリゲート名-2で指定されるデリゲートの持つ型パラメタの個数と、一致していなければなりません。

メソッド指定子固有の規則

1. メソッド識別名-2、クラス名-4、インタフェース名-4、デリゲート名-3およびENUM名-2は、同じリポジトリ段落で宣言した名前でなければなりません。
2. メソッド識別名-2は、型パラメタを持つメソッド名を指定する名前でなければなりません。
3. USING指定に指定する名前の個数は、EXPANDS指定に記述したメソッド識別名-2で指定されるメソッドの持つ型パラメタの個数と、一致していなければなりません。

一般規則

1. ASが指定されているならば、その定数で示した名前が外部名とみなされ、AS指定に先行する利用者語は内部名とみなされます。
2. プロパティ指定子を除き、ASに指定する定数は、名前空間や入れ子の上位階層要素の名前で完全修飾された名前でなければなりません。

クラス指定子固有の規則

1. クラス指定子は、これを含む環境部の有効範囲内で使用されるクラスの名前を、クラス名-1として宣言します。
2. クラス名-1が総称クラスを指定する場合、他のクラス指定子のEXPANDS指定にだけ指定できます。
3. USING指定に指定した型は、指定した順に、クラス名-2の指定する総称クラスに型引数として渡されます。クラス名-2の型パラメタが制約を持つ場合、対応する型引数となる型は、制約を満たす型でなければなりません。
4. EXPANDS指定を持つクラス指定子は、EXPANDS指定に記述したクラス名-2の指定する総称クラスと、USING指定で指定される型引数の組合せに名前を付けます。

プロパティ指定子固有の規則

1. プロパティ指定子は、これを含む環境部の有効範囲内で使用されるプロパティまたはフィールドの名前を、プロパティ名-1またはフィールド名-1として宣言します。

インタフェース指定子固有の規則

1. インタフェース指定子は、これを含む環境部の有効範囲内で使用されるインタフェースの名前を、インタフェース名-1として宣言します。
2. インタフェース名-1が、総称インタフェースを指定する場合、他のインタフェース指定子のEXPANDS指定だけに指定できます。
3. USING指定に指定した型は、指定した順に、インタフェース名-2の指定する総称インタフェースに型引数として渡されます。インタフェース名-2の型パラメタが制約を持つ場合、対応する型引数となる型は、制約を満たす型でなければなりません。
4. EXPANDS指定を持つインタフェース指定子は、EXPANDS指定に記述したインタフェース名-2の指定する総称インタフェースと、USING指定で指定される型引数の組合せに名前を付けます。

プログラム指定子固有の規則

1. プログラム指定子は、これを含む環境部の有効範囲内で使用されるプログラム原型の名前を、プログラム原型名-1として宣言します。
2. プログラム指定子を指定する場合、そのプログラム指定子を含むソース単位と同じ翻訳集団単位内に、プログラム原型名-1で示されるプログラム原型定義が存在していなければなりません。

デリゲート指定子固有の規則

1. デリゲート指定子は、これを含む環境部の有効範囲内で使用されるデリゲートの名前を、デリゲート名-1として宣言します。
2. デリゲート名-1が、総称デリゲートを指定する場合、他のデリゲート指定子のEXPANDS指定にだけ指定できます。
3. USING指定に指定した型は、指定した順に、デリゲート名-2の指定する総称デリゲートに型引数として渡されます。デリゲート名-2の型パラメタが制約を持つ場合、対応する型引数となる型は、制約を満たす型でなければなりません。
4. EXPANDS指定を持つデリゲート指定子は、EXPANDS指定に記述したデリゲート名-2の指定する総称デリゲートと、USING指定で指定される型引数の組合せに名前を付けます。

ENUM指定子固有の規則

1. ENUM指定子は、これを含む環境部の有効範囲内で使用されるENUMの名前を、ENUM名-1として宣言します。

メソッド指定子固有の規則

1. メソッド指定子は、これを含む環境部の有効範囲内で使用されるメソッドと型引数の組合せに名前を付け、これをメソッド識別名-1として宣言します。
2. メソッド識別名-1が総称メソッドを指定する場合、他のメソッド指定子のEXPANDS指定にだけ指定できます。
3. USING指定に指定した型は、指定した順に、メソッド識別名-2の指定する総称メソッドに型引数として渡されます。メソッド識別名-2の型パラメタが制約を持つ場合、対応する型引数となる型は、制約を満たす型でなければなりません。
4. EXPANDS指定を持つメソッド指定子は、EXPANDS指定に記述したメソッド識別名-2の指定する総称メソッドと、USING指定で指定される型引数の組合せに名前を付けます。

11.6.2 入出力節（INPUT-OUTPUT SECTION）

ここでは、.NETプログラミング機能における、入出力節について説明します。その他の規則については、“4.3 入出力節（INPUT-OUTPUT SECTION）”を参照してください。

【書き方】

INPUT-OUTPUT SECTION.
[ファイル管理段落]
[入出力管理段落]

構文規則

入出力節は、プログラム定義、スタティック定義、オブジェクト定義およびメソッド定義の中に指定できます。ただし、インタフェース定義に含まれるメソッド定義に指定することはできません。

11.6.2.1 ファイル管理段落（FILE-CONTROL）

11.6.2.1.1 ファイル管理記述項

順ファイル、相対ファイルおよび索引ファイル機能のファイル管理記述項については、“4.3.1 ファイル管理段落（FILE-CONTROL）”を参照してください。

【表示ファイルのファイル管理記述項の書き方】

SELECT句

[ACCESS MODE句]

ASSIGN句

［DESTINATION句］

[FILE STATUS区]

[FORMAT句]

[GROUP句]

[ORGANIZATION句]

[PROCESSING CONTROL句]

[PROCESSING MODE句]

[SELECTED FUNCTION句]

[SESSION CONTROL区]

SYMBOLIC DESTINATION句

[UNIT CONTROL句]

11.6.2.1.2 SYMBOLIC DESTINATION句

構文規則

あて先種別として指定できる値は“PRT”のみです。

11.6.2.2 入出力管理段落（I-O-CONTROL）

ここでは、.NETプログラミング機能における、入出力管理段落の規則についてだけ説明します。その他の規則の詳細は、“4.3.2 入出力管理段落（I-O-CONTROL）”を参照してください。

構文規則

1. I-O-CONTROL段落には、APPLY MULTICONVERSATION-MODE句またはAPPLY SAVED-AREA句を指定してはなりません。
2. スタティック定義、オブジェクト定義およびメソッド定義のI-O-CONTROL段落には、RERUN句およびMULTIPLE FILE TAPE句を書いてはなりません。
3. APPLY SAVED-AREA句のデータ名-1およびデータ名-1に従属するデータ項目に、USAGE OBJECT REFERENCE句を指定してはなりません。

11.7 データ部

ここでは、.NETプログラミング機能固有のデータ部について説明します。

11.7.1 ファイル節（FILE SECTION）

ここでは、.NETプログラミング機能における、ファイル節の規則についてだけ説明します。その他の規則の詳細は、“5.1 データ部の構成（DATA DIVISION）”を参照してください。

構文規則

1. ファイル節はプログラム定義、スタティック定義、オブジェクト定義またはメソッド定義の中にだけ指定できます。ただし、インタフェース定義に含まれるメソッド定義に指定することはできません。
2. スタティック定義、オブジェクト定義およびメソッド定義のファイル節では、LABEL RECORD句、VALUE OF句およびDATA RECORDS句を書いてはなりません。その他の句は指定することができます。ただし、スタティック定義、オブジェクト定義では、GLOBAL句およびEXERNAL句を書いてはなりません。

3. メソッド定義では、EXTERNAL句とLINAGE句を同時に指定することはできません。
4. 報告書記述項は宣言できません。

11.7.2 連絡節（LINKAGE SECTION）

ここでは、.NETプログラミング機能における、連絡節の規則についてだけ説明します。その他の規則の詳細は、“5.1 データ部の構成（DATA DIVISION）”を参照してください。

構文規則

1. ソース要素の連絡節で定義されたデータ項目は、以下の条件のいずれかを満たす場合にだけ、そのソース要素の手続き部で参照できます。
   a. 手続き部の見出しのUSING指定またはRETURNING指定のオペランドである
   b. 手続き部の見出しのUSING指定またはRETURNING指定のオペランドに従属している
   c. 上記の条件を満たす目的語に対するREDEFINES句またはRENAMES句が定義されている
   d. 規則 1.の c)の条件を満たす項目に従属している
   e. 上記の条件の1つを満たすデータ項目に関連付けられた条件名または指標名である
2. 連絡節データの名前は、ソース単位に含まれる他の連絡節データの名前と重複してはなりません。

11.7.3 データ記述項

ここでは、.NETプログラミング機能における、データ記述項について説明します。

【書き方】

```
レベル番号 [ データ名-1 ]
           [ FILLER    ]
[REDEFINES データ名-2]
[IS TYPEDEF]
[IS EXTERNAL]
[IS GLOBAL]
[ { PICTURE } IS PICTURE句の文字列 ]
[ { PIC     }                      ]
[ VALUE IS { 定数-1 } ]
[          { AUTO   } ]
[USAGE句]
[ [SIGN IS] { LEADING  } ]
[           { TRAILING } ]
[ [SEPARATE CHARACTER]   ]
[OCCURS句]
[ { SYNCHRONIZED } ][ LEFT  ] ]
[ { SYNC        } ][ RIGHT ] ]
[ { JUSTIFIED } RIGHT ]
[ { JUST      }       ]
```

[BLANK WHEN { ZERO / ZEROS / ZEROES }]

[PROPERTY [WITH NO { GET / SET }]]

[TYPE 型名-1]

[CHARACTER TYPE句]

[PRINTING POSITION句]

[CUSTOM-ATTRIBUTE句]

[IS { PUBLIC / PROTOTECTED / PRIVATE } [AS 定数-2]]

.

ここで、USAGE句は、

[USAGE IS] {
BINARY
BIT
COMPUTATIONAL
COMP
COMPUTATIONAL-1
COMP-1
COMPUTATIONAL-2
COMP-2
COMPUTATIONAL-3
COMP-3
COMPUTATIONAL-4
COMP-4
COMPUTATIONAL-5
COMP-5
DISPLAY
INDEX
PACKED-DECIMAL
POINTER
OBJECT [REFERENCE] { クラス名-1 / インタフェース名-1 / デリゲート名-1 / ENUM名-1 }
BINARY-CHAR UNSOGNED
BINARY-SHORT [SIGNED]
BINARY-LONG [SIGNED]
BINARY-DOUBLE [SIGNED]
}

構文規則

1. スタティック定義およびオブジェクト定義の作業場所節で宣言されるレベル番号01または77のデータ項目は、無名項目であってはなりません。また、その名前は、ほかの01または77レベルのスタティックデータおよびオブジェクトデータの名前と重複してはなりません。
2. ENUM定義の場合、ENUMの要素は作業場所節で宣言されるレベル番号01の基本項目でなければなりません。このとき、VALUE句は省略できません。また、VALUE句以外の句は記述してはなりません。
3. ENUM定義の場合、VALUE句には数字定数またはAUTO指定を指定します。数字定数のとき、数字定数の値はINHERITS句で示される値の範囲に入るものでなければなりません。

11.7.3.1 CHARACTER TYPE句

ここでは、.NETプログラミング機能における、CHARACTER TYPE句の規則について説明します。その他の規則については、“5.4.2 CHARACTER TYPE句”を参照してください。

【書き方1】日本語項目または日本語編集項目の文字サイズを指定する

```
[CHARACTER TYPE IS] { MODE-1 | MODE-2 | MODE-3 } [BY 呼び名-1]
```

【書き方2】日本語項目または日本語編集項目の文字サイズ、文字ピッチ、文字書体、文字回転量および文字形態を指定する

```
CHARACTER TYPE IS 呼び名-2
```

【書き方3】任意のデータ項目の文字サイズ、文字ピッチ、文字書体、文字回転量および文字形態を指定する

```
CHARACTER TYPE IS
    { 印字モード名-1                                  }
    { [印字モード名-2] … DEPENDING ON データ名-1      }
  [OR
    { 印字モード名-3                                  }
    { [印字モード名-4] … DEPENDING ON データ名-2      }  ]
```

構文規則

1. CHARACTER TYPE句は、プログラム定義およびクラス定義内のデータ部でのみ使用できます。
2. スタティック定義またはオブジェクト定義のデータ部には、DEPENDING ON指定を含むCHARACTER TYPE句を書いてはなりません。

11.7.3.2 CUSTOM-ATTRIBUTE句

CUSTOM-ATTRIBUTE句は、この句を指定した定義に、カスタム属性を与える機能を提供します。

【書き方】

CUSTOM-ATTRIBUTE IS {カスタム属性名-1} …

構文規則

1. CUSTOM-ATTRIBUTE句は、以下のデータ記述項にだけ指定できます。
   - ー スタティック定義およびオブジェクト定義の作業場所節で宣言されるCLR型項目
   - ー 連絡節のデータ定義
2. カスタム属性名-1は、この句を指定した定義に指定可能な属性を持つアトリビュートクラスによって定義したものでなければなりません。
3. アトリビュートクラスが”Systen.ParamArrayAttribute”クラスである場合、さらに以下の規則が適合されます。
   a. メソッド(プロパティ設定メソッドおよびプロパティ取得メソッドを除く)の連絡節のデータ項目である。
   b. データ項目が手続き部見出しのUSING指定に最後に指定される項目である。
   c. データ項目が1次元の配列である。
4. カスタム属性名-1を定義するアトリビュートクラスが、同一定義への複数指定を許さないものである場合、同じアトリビュートクラスで定義したカスタム属性名を複数指定することはできません。

11.7.3.3 EXTERNAL句

ここでは、.NETプログラミング機能における、EXTERNAL句の規則について説明します。その他の規則については、“5.4.3 EXTERNAL句”を参照してください。

【書き方】

IS EXTERNAL

構文規則

1. EXTERNAL句は、スタティック定義およびオブジェクト定義のデータ部に指定してはなりません。
2. EXTERNAL句は、用途がオブジェクト参照であるデータ項目には指定してはなりません。

11.7.3.4 GLOBAL句

ここでは、.NETプログラミング機能における、GLOBAL句の規則について説明します。その他の規則については、“5.4.4 GLOBAL句”を参照してください。

【書き方】

IS GLOBAL

構文規則

1. GLOBAL句は、スタティック定義およびオブジェクト定義のデータ部に指定してはなりません。
2. GLOBAL句は、用途がオブジェクト参照である連絡節のデータ項目には指定してはなりません。

11.7.3.5 OCCURS句

ここでは、.NETプログラミング機能における、OCCURS句の規則について説明します。その他の規則については、“5.4.6 OCCURS句”を参照してください。

【書き方1】(一定の繰返し回数を指定する)

```
OCCURS 整数-2 TIMES
    [ { ASCENDING  } KEY IS {データ名-2} … ]
      { DESCENDING }
  [INDEXED BY {指標名-1} … ]
```

【書き方2】(可変の繰返し回数を指定する)

```
OCCURS      [整数-1]          TO 整数-2 TIMES
          DEPENDING ON データ名-1
    [ { ASCENDING  } KEY IS {データ名-2} … ]
      { DESCENDING }
  [INDEXED BY {指標名-1} … ]
```

構文規則

KEY IS指定には、字類がブールまたはオブジェクトのデータ項目を指定してはなりません。

11.7.3.6 PRINTING POSITION句

構文規則

PRINTING POSITION句は、プログラム定義およびクラス定義内のデータ部でのみ使用できます。

11.7.3.7 PROPERTY句

ここでは、.NETプログラミング機能における、PROPERTY句の規則について説明します。

PROPERTY句は、このデータ項目がこのオブジェクトのプロパティであるとみなされ、それにより、GETまたはSETメソッドが生成されることを示します。

【書き方】

```
PROPERTY [ WITH NO { GET } ]
           { SET }
```

構文規則

1. PROPERTY句は、スタティック定義またはオブジェクト定義の作業場所節だけで使用できます。
2. PROPERTY句は、TYPEDEF句が書かれたデータ項目またはそれに従属するデータ項目に指定できません。

3. PROPERTY句は、レベル番号01または77の基本項目にだけ指定できます。
4. PROPERTY句を、TYPE句と同時に参照する場合、TYPE句の参照する型は上記構文規則 3.に違反しない項目として定義されていなければなりません。
5. SET指定が書かれない場合、このデータ項目は、数字項目、オブジェクト参照項目または1文字の日本語項目でなければなりません。

一般規則

1. GET指定が書かれなかった場合、PROPERTY句はそれを含むスタティックまたはオブジェクト用に定義されたメソッドを生成します。データ項目の用途がオブジェクト参照または指標の場合、このメソッドの暗黙の定義は、以下のとおりです。

```
METHOD-ID.  GET  PROPERTY  データ名.
DATA        DIVISION.
LINKAGE     SECTION.
01  LS-データ名  データ記述.
PROCEDURE   DIVISION RETURNING LS-データ名.
段落名.
     SET LS-データ名 TO データ名
     EXIT METHOD.
END METHOD.
```

データ項目の字類が英数字または日本語の場合、このメソッドの暗黙の定義は、以下のとおりです。

```
METHOD-ID.  GET  PROPERTY  データ名.
DATA        DIVISION.
LINKAGE     SECTION.
01  LS-データ名  データ記述.
PROCEDURE   DIVISION RETURNING LS-データ名.
段落名.
     MOVE データ名 TO LS-データ名(1:)
     EXIT METHOD.
END METHOD.
```

さもなければ、このメソッドの暗黙の定義は、以下のようになります。

```
METHOD-ID.  GET  PROPERTY  データ名.
DATA        DIVISION.
LINKAGE     SECTION.
01  LS-データ名  データ記述.
PROCEDURE   DIVISION RETURNING LS-データ名.
段落名.
     MOVE データ名 TO LS-データ名
     EXIT METHOD.
END METHOD.
```

ここで、LS-データ名は、従属する項目も含めて、このデータ項目と同じデータ記述項を持ちます。ただし、以下を除きます。

- INDEXED BY句
- PROPERTY句
- VALUE句
- REDEFINES句
- CUSTOM-ATTRIBUTE句
- アクセス指定句

2. SET指定が書かれなかった場合、PROPERTY句はそれを含むスタティックまたはオブジェクト用に定義されたメソッドを生成します。データ項目の用途がオブジェクト参照の場合、このメソッドの暗黙の定義は、以下のとおりです。

```
METHOD-ID.  SET  PROPERTY  データ名.
DATA        DIVISION.
LINKAGE     SECTION.
01  LS-データ名  データ記述.
PROCEDURE   DIVISION USING BY VALUE LS-データ名.
段落名.
      SET データ名 TO LS-データ名
      EXIT METHOD.
END METHOD.
```

データ項目の字類が日本語の場合、このメソッドの暗黙の定義は、以下のとおりです。

```
METHOD-ID.  SET  PROPERTY  データ名.
DATA        DIVISION.
LINKAGE     SECTION.
01  LS-データ名  データ記述.
PROCEDURE   DIVISION USING BY VALUE LS-データ名.
段落名.
      MOVE LS-データ名 TO データ名(1:)
      EXIT METHOD.
END METHOD.
```

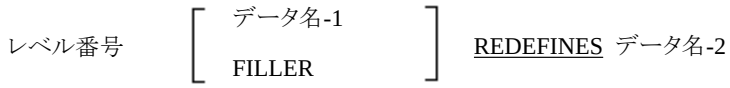
さもなければ、このメソッドの暗黙の定義は、以下のようになります。

```
METHOD-ID.  SET  PROPERTY  データ名.
DATA        DIVISION.
LINKAGE     SECTION.
01  LS-データ名  データ記述.
PROCEDURE   DIVISION USING BY VALUE LS-データ名.
段落名.
      MOVE LS-データ名 TO データ名
      EXIT METHOD.
END METHOD.
```

11.7.3.8 REDEFINES句

ここでは、.NETプログラミング機能における、REDEFINES句の規則について説明します。その他の規則については、“5.4.9 REDEFINES句”を参照してください。

【書き方】



構文規則

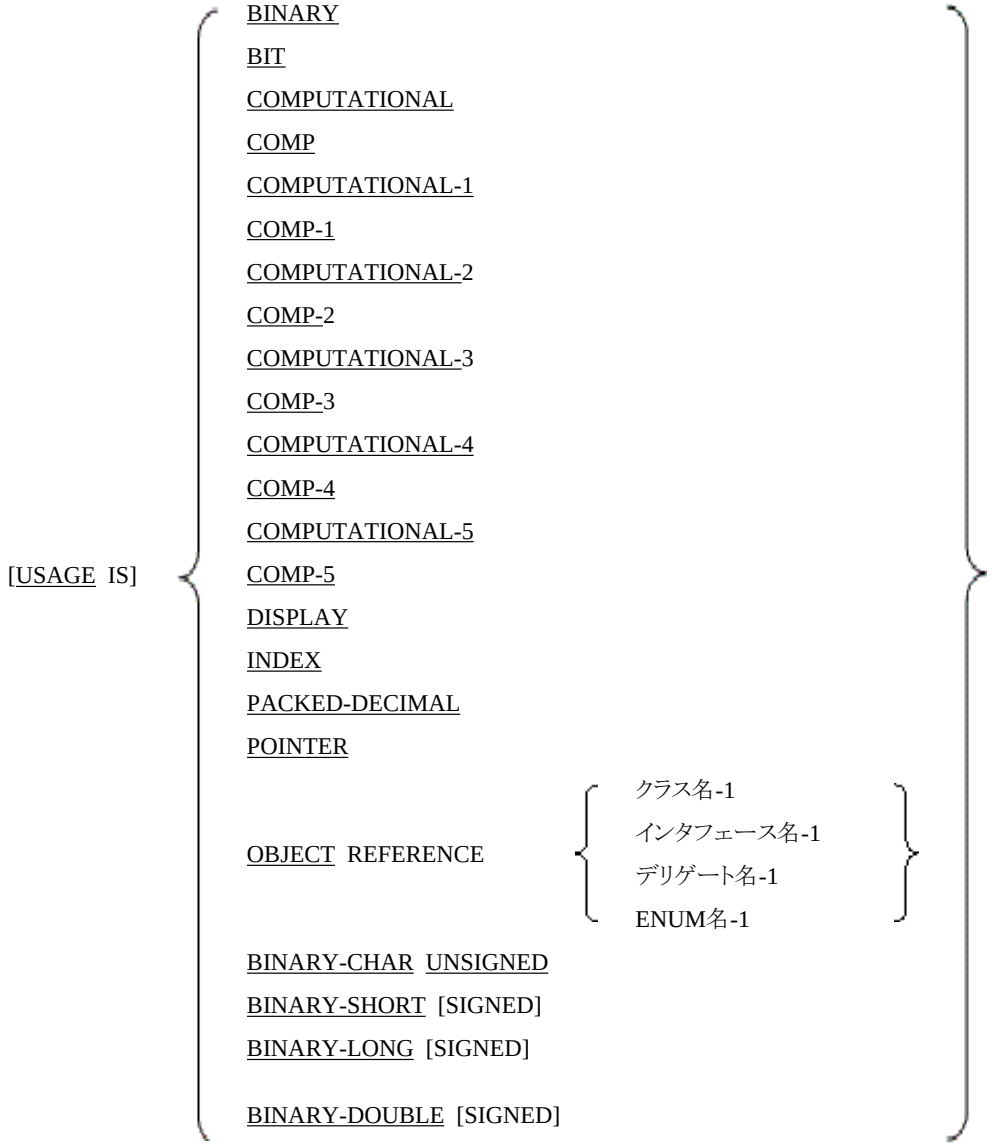
1. REDEFINES句は、用途がオブジェクト参照として宣言されたデータ項目に対して指定できません。
2. データ名-2のデータ記述項は、用途がオブジェクト参照であってはなりません。

11.7.3.9 USAGE句

ここでは、.NETプログラミング機能における、USAGE句について説明します。その他の規則については、“5.4.15 USAGE句”を参照してください。

USAGE句は、データ項目の用途を指定します。

【書き方】



構文規則

1. BLANK WHEN ZERO、JUSTIFIED、PICTUREおよびSYNCHRONIZED句は用途が指標またはオブジェクト参照のデータ項目に指定してはなりません。
2. USAGE OBJECT REFERENCE句は、レベル番号が01または77の基本項目にだけ指定できます。
3. USAGE OBJECT REFERENCE句は、作業場所節および連絡節にだけ書けます。
4. クラス名-1は、そのデータ記述項を含むソース単位中のクラス指定子で宣言したクラス名、またはそのデータ記述項を含むクラス定義のクラス名でなければなりません。
5. クラス名-1が以下のクラスを示す場合、そのデータ記述項はプログラム定義およびメソッド定義の作業場所節にしか記述できません。
   - System.Byte

- System.Int16
- System.Int32
- System.Int64
- System.Single
- System.Double
- System.Char

クラス名-1が以下のクラスを示す場合、OBJECT REFERENCE句に指定できません。

- System.Void

6. インタフェース名-1は、そのデータ記述項を含むソース単位中のインタフェース指定子で宣言しなければなりません。
7. デリゲート名-1は、そのデータ記述項を含むソース単位中のデリゲート指定子で宣言しなければなりません。
8. ENUM名-1は、そのデータ記述項を含むソース単位中のENUM指定子で宣言しなければなりません。

一般規則

1. USAGE OBJECT REFERENCE句を含むデータ記述は、オブジェクト参照を保持するのに十分な領域を割りつけます。
2. USAGE OBJECT REFERENCEを指定したデータ記述項は、動的オブジェクトを参照するのに使われるオブジェクト参照一意名です。
3. CLRの基本データ型とCOBOLのデータ型の対応を、以下の表に示します。

表11.2 CLR型とCOBOLのデータ型の対応

CLR型	COBOLのデータ型	PICTURE
int8	OBJECT REFERENCE "System.SByte"	-
unsigned int8	BINARY-CHAR UNSIGNED	-
int16	COMP-5	S9(4)
	BINARY-SHORT SIGNED	-
unsigned int16	OBJECT REFERENCE "System.UInt16"	-
int32	COMP-5	S9(9)
	BINARY-LONG SIGNED	-
unsigned int32	OBJECT REFERENCE "System.UInt32"	-
int64	COMP-5	S9(18)
	BINARY-DOUBLE SIGNED	-
unsigned int64	OBJECT REFERENCE "System.UInt64"	-
float32	COMP-1	-
float64	COMP-2	-
char	[DISPLAY]	N
decimal	OBJECT REFERENCE "System.Decimal"	-
Boolean	OBJECT REFERENCE "System.Boolean"	-
object	OBJECT REFERENCE "System.Object"	-
string	OBJECT REFERENCE "System.String"	-

上記表の“System.～”は、実際にはリポジトリ段落で定義した名前です。

4. CLR型のデータを定義するには、以下の条件をすべて満たさなければなりません。ひとつでも満足しない場合はCOBOL独自型のデータとなります。

    a. 作業場所節または連絡節で定義されたデータ項目でなければなりません。ただしプログラム定義の連絡節データは、用途がオブジェクト参照であるものを除いて、COBOL独自型です。

    b. 01または77レベルの基本項目でなければなりません。

    c. ほかの項目を再定義せず、ほかの項目から再定義されません。

    d. 再命名されてはなりません。

    e. EXTERNAL句が指定されていてはなりません。

    f. SQLホスト変数であってはなりません。

    g. データの属性は、以下のいずれかでなければなりません。

        - OBJECT REFERENCE
        - BINARY-CHAR UNSIGNED
        - BINARY-SHORT SIGNED
        - BINARY-LONG SIGNED
        - BINARY-DOUBLE SIGNED
        - S9(4) COMP-5
        - S9(9) COMP-5
        - S9(18) COMP-5
        - COMP-1
        - COMP-2
        - PIC N

    h. CHARACTER TYPE句、PRINTING POSITION句が指定されていてはなりません。

11.7.3.10 VALUE句

ここでは、.NETプログラミング機能における、VALUE句の規則について説明します。その他の規則については、“5.4.16 VALUE句”を参照してください。

【書き方1】(データ項目に初期値を与える)

```
VALUE IS { 定数-1 }
         { AUTO   }
```

【書き方2】(条件名の値を指定する)

```
{ VALUE  IS } [定数-2 [ { THROUGH } 定数-3 ] ] …
{ VALUES ARE }        [ { THRU    }        ]
```

構文規則

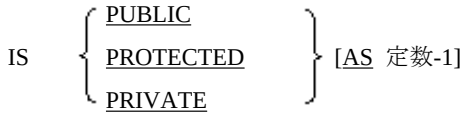
1. VALUE句は、用途がオブジェクト参照であるデータ項目には指定できません。
2. AUTO指定はENUM定義の要素を定義する項目のみ指定できます。

一般規則

1. USAGE OBJECT REFERENCE句の指定されたデータ項目は、このデータがReference Typeの場合にはNULLオブジェクトを参照するように初期化されます。この初期値は、VALUE句が作用するのと同じ時に設定されます。このデータがValue Typeの場合、初期値は不定です。
2. AUTO指定が指定された場合、要素の値は自動的に割り振られます。
3. 最初に定義された要素のVALUE句にAUTO指定が指定された場合、その要素の値の初期値は0です。それ以外のAUTO指定の値は1つ前に定義された要素の値に1を加算することによって決定されます。

11.7.3.11 アクセス指定句（PUBLIC/PRIVATE/PROTECTED）

【書き方】



構文規則

1. アクセス指定句は、スタティック定義またはオブジェクト定義の作業場所節あるいはメソッド定義の連絡節で宣言されたCLR型データ項目にだけ指定できます。
2. PUBLIC/PROTECTED/PRIVATEを同時に指定することはできません。
3. 連絡節データにはPRIVATE指定しか記述できません。
4. 定数-1は文字定数または日本語文字定数でなければなりません。ただし、16進文字定数や日本語16進文字定数であってはなりません。
5. 定数-1が作業場所節データに対して指定された場合、定数-1で示される名前は、このクラス内で宣言された他のフィールド／プロパティの外部名と重複してはなりません。
6. 定数-1が連絡節データに対して指定された場合、定数-1で示される名前は、この連絡節内で宣言された他のデータ項目の外部名と重複してはなりません。
7. アクセス指定句が記述されなかった場合は、PRIVATEが指定されたとみなします。ただし、PROPERTY句が指定されたデータ項目であればPUBLICが指定されたとみなします。

一般規則

1. PROPERTY句が指定された項目にアクセス指定句が指定された場合は、それらはプロパティに対する指定であるとみなします。
2. AS指定を記述した場合、定数-1が、このフィールド、プロパティあるいは連絡節データ項目の外部名になります。
3. PUBLICを指定したフィールドまたはプロパティは、それを含むクラス内またはそのクラスを参照するソース単位から利用できます。
   手続き部でこれらにアクセスするときはフィールド参照またはオブジェクトプロパティの構文を記述します。
4. PROTECTEDを指定したフィールドまたはプロパティは、それを含むクラス内またはそのクラスを継承するクラスから利用できます。
   手続き部でこれらにアクセスするときはフィールド参照またはオブジェクトプロパティの構文を記述します。
5. PRIVATEが明または暗に指定されたフィールドまたはプロパティは、それを含むクラス内でだけ利用できます。
   手続き部でこれらにアクセスするときはフィールド参照またはオブジェクトプロパティの構文を記述します。
   また、PRIVATEが明または暗に指定された連絡節データ項目は、それを含むメソッド定義内でだけ参照できます。

11.8 手続き部

ここでは、.NETプログラミング機能固有の手続き部について説明します。

11.8.1 手続き部の見出し

ここでは、.NETプログラミング機能における、手続き部の見出しについて説明します。

メソッドまたはプログラムの手続き部は、実行される手続きを含みます。

スタティック定義とオブジェクト定義の手続き部は、スタティックまたはオブジェクト上で呼び出されるメソッドを含みます。

【書き方1】(節を持つ手続き部)

```
PROCEDURE DIVISION
        [USING指定]
        [RETURNING データ名-2]
        [RAISING {クラス名-1} … ].
[DECLARATIVES.
        {節名 SECTION.
USE文.
        [段落名. [完結文] … ] … }
END DECLARATIVES.]
{節名 SECTION.
        [段落名. [完結文] … ] … } …
```

【書き方2】(節を持たない手続き部)

```
PROCEDURE DIVISION
        [USING指定]
        [RETURNING データ名-2]
        [RAISING {クラス名-1} … ].
          [段落名. [完結文] … ] … } …
```

【書き方3】(手続きを持たない手続き部)

```
PROCEDURE DIVISION
      [USING指定]
      [RETURNING データ名-2]
      [RAISING {クラス名-1} … ].
```

【書き方4】(スタティック定義、オブジェクト定義、インタフェース定義の手続き部)

```
PROCEDURE DIVISION.
      [{メソッド定義} …]
```

ここで、USING指定は、

```
USING  { [BY REFERENCE] {データ名-1} …  }
       { BY VALUE {データ名-1} …        }  …
```

構文規則

書き方1～書き方3に共通する規則

1. データ名-1は、連絡節でレベル番号01または77の項目として定義されていなければなりません。1つの利用者語が、データ名-1として2回以上現れてはなりません。データ名-1のデータ記述項は、REDEFINES句を含んでいてはなりません。ただし、連絡節内の、ほかの場所でのデータ項目には、REDEFINES データ名-1と指定できます。
2. BY VALUE指定に書かれたそれぞれのデータ名-1は、CLR型データ項目または数字項目として定義されていなければなりません。
3. データ名-2は、連絡節のレベル番号01または77のデータ項目として定義されなければなりません。データ名-2のデータ記述項は、REDEFINES句を含んでいてはなりません。ただし、連絡節の、ほかの場所のデータ項目には、REDEFINES データ名-2と指定できます。
4. データ名-2は、データ名-1と同じであってはなりません。
5. クラス名-1は、リポジトリ段落に指定されたクラスの名前でなければなりません。
6. プログラム定義の手続き部の見出しには、BY VALUE指定は記述できません。
7. スタティックコンストラクタの手続き部の見出しには、USINGおよびRETURNINGは指定できません。
8. インスタンスコンストラクタの手続き部の見出しには、RETURNINGは指定できません。

書き方3の規則

プログラム原型定義およびデリゲート定義の手続き部の記述は、書き方3でなければなりません。

書き方4の規則

書き方4は、スタティック定義、オブジェクト定義またはインタフェース定義でだけ使用できます。

一般規則

書き方1および書き方2に共通する規則

実行は、宣言部分を除く手続き部の最初の文から始まります。文は、規則がほかの順序を指定している場合を除いて、翻訳時に現れた順に実行されます。

書き方1～書き方3に共通する規則

1. USING指定は、メソッドまたはプログラムで使用される仮引数の名前を識別します。それに渡される引数は、呼び出すソース要素の以下のいずれかで識別されます。
   - CALL文のUSING指定
   - INVOKE文のUSING指定
   - メソッドの行内呼出しに指定された作用対象
2. この仮引数の並びとソース要素の呼び出す文のUSING指定に記述した実引数の対応付けは、USING指定中の位置を基準として行われます。
3. BY REFERENCE指定とBY VALUE指定はともに、それらに続く引数に対して、別のBY REFERENCEまたはBY VALUEが現れるまで有効です。最初の引数の前にBY REFERENCE指定もBY VALUE指定も書かなかった場合、BY REFERENCE指定が書かれたものとみなされます。
4. データ名-1は、メソッドまたはプログラムの仮引数です。
5. データ名-2は、メソッドの結果、プログラムの結果にしたがって、呼び出した要素に結果を返すために、メソッドまたはプログラム内で使われる名前です。

6. 復帰項目であるデータ名-2の初期値は、不定です。
7. 引数がBY REFERENCEで渡される場合、呼び出された側は、連絡節のデータ項目が呼出し側の引数と同じ領域を占有するかのように処理します。
8. 引数がBY CONTENT指定で渡される場合、呼び出された側は、呼出しの初期化時に呼び出す側により連絡節のレコードが割り付けられたように処理します。このレコードは、呼出し側の引数と同じ領域を占有しません。
   呼び出されるのがプログラムの場合、この割り付けられたレコードは、引数と正確に同じ長さを持ちます。その引数は、この割り付けられたレコードに変換なしに転記されます。このレコードは、呼び出された側により、それが引数でありBY REFERENCEにより渡されたのと同じように扱われます。
   呼び出されるのがプログラム原型またはメソッドの場合、この割り付けられたレコードは、連絡節のレコードと正確に同じ長さを持ちます。引数は送出し側項目として、連絡節に指定された記述を持つ、割り付けられたレコードは、受取り側項目として、以下の文で使用されます。
   - 仮引数が数字の場合、COMPUTE文
   - 仮引数が指標データ項目または用途がオブジェクト参照の場合、SET文
   - それ以外であれば、MOVE文

   割り付けられたレコードは、それが引数でありBY REFERENCEにより渡されたのと同じように扱われます。
9. 引数がBY VALUEで渡される場合、呼出しの初期化時に呼び出す側により連絡節のレコードが割り付けられたように処理します。
   割り付けられたレコードは、長さにおいて、連絡節の領域と正確に同じ英数字文字位置を持ちます。ROUNDED指定のないCOMPUTE文で、引数が送出し側、連絡節で指定された記述を持つ、割り付けられたレコードが受取り側として使用されます。呼び出された側は、割り付けられたレコードにアクセスできます。
10. 呼び出された側では常に、データ名-1およびデータ名-2の参照は、連絡節中のそれらの記述により解決されます。
11. クラス名-1は、EXIT文において発生させる可能性のある例外オブジェクトのクラス名またはそのクラスを継承するクラス名を指定します。

11.8.2 文に関する共通の規則

ここでは、.NETプログラミング機能における、共通の規則および文に共通して書くことができる指定について説明します。

11.8.2.1 呼出し演算子

呼出し演算子は、分離符の空白に続く2つの連続したCOBOL文字“::”です。それらは分離符の空白に挟まれていなければなりません。呼出し演算子の使い方については、“11.3.3.2 メソッドの行内呼出し”を参照してください。

11.8.2.2 メソッドおよびオブジェクトの状態

以下に、メソッドおよびオブジェクトの状態について説明します。

11.8.2.2.1 メソッドの状態

メソッドの状態は、実行単位のどんな時点でも活性または不活性のどちらかです。メソッドは、活性化されるたびに初期状態になります。

活性状態

メソッドの実体は、メソッドの呼出しが成功したとき、活性状態になります。そして、このメソッドの実体中のSTOP文の実行、または明または暗のEXIT METHOD文が実行されるまで活性状態のままです。

メソッドは、再帰的に活性状態になります。しかし、それぞれのメソッドの実体は、一度しか活性化されません。

メソッドが活性化されたときはいつも、メソッドに含まれるPERFORMの制御機構は初期状態に設定されます。

データの初期状態

メソッドの内部データは、それが記述されたメソッドが活性化されるたびに初期状態になります。

メソッドのデータが初期状態になると、以下のようになります。

1. 作業場所節に記述された内部データが以下の方法で初期化されます。データ項目の記述にVALUE句が書かれていた場合、データ項目は、定義された値で初期化されます。データ項目にVALUE句が書かれていない場合、データ項目の初期値は規定されません。指標の初期値も規定されません。
2. メソッドの内部ファイル結合子は、ファイルが開かれていない状態に設定されることにより初期化されます。

11.8.2.2.2 オブジェクトにだけ関連付けられたデータの状態

- オブジェクトの初期状態は、作成された直後のオブジェクトです。
- オブジェクトの作業場所節で記述された内部データは、メソッドまたはプログラムの作業場所節に記述された内部データと同じ方法で初期化されます。
- オブジェクトに関連付けられた内部ファイル結合子を持つファイルは、開かれていない状態です。

11.8.2.3 制御の明示移行と暗黙移行

次の実行可能な文がなく、制御がソース単位の外に移行しない場合、下記以外の場合の制御の流れは規定されません。

1. プログラムの場合、宣言部分でない場所を実行中であり、プログラムがCALL文の制御下にあるとき、暗黙のEXIT PROGRAM文が実行されます。
2. メソッドの場合、宣言部分でない場所を実行中のとき、他の選択指定のない暗黙のEXIT METHOD文が実行されます。

11.8.2.4 条件式

ここでは、.NETプログラミング機能における、条件式について説明します。

11.8.2.4.1 字類条件

字類条件は、データ項目の内容の字類を検査するために使用します。

【書き方】

一意名-1 IS [NOT] { NUMERIC | ALPHABETIC | ALPHABETIC-LOWER | ALPHABETIC-UPPER | BOOLEAN | 字類名 }

一般規則

1. 字類条件では、字類検査の指定にしたがって一意名-1の内容を検査し、真理値を決定します。字類検査が真になる条件を、以下の表に示します。

表11.3 字類検査が真になる条件

字類検査の種類	字類検査が真になる条件
NUMERIC	一意名-1の内容が、数字0、1、2、…、9だけからなる、または、これらに演算符号を付けたものである場合

字類検査の種類	字類検査が真になる条件
ALPHABETIC	一意名-1の内容が、英大文字A、B、…、Zと空白からなる、または、英小文字a、b、…、zと空白からなる、または、大文字と英小文字と空白の組合せからなる場合
ALPHABETIC-LOWER	一意名-1の内容が、英小文字a、b、…、zと空白からなる場合
ALPHABETIC-UPPER	一意名-1の内容が、英大文字A、B、…、Zと空白からなる場合
BOOLEAN	一意名-1の内容が、ブール文字0、1だけからなる場合
字類名	一意名-1の内容が、特殊名段落のCLASS句で指定した文字の組からなる場合

2. 一意名-1は、以下のいずれかでなければなりません。

   ー 用途が表示用のデータ項目

   ー 英数字関数または日本語関数の関数一意名

   ー 内部10進項目

3. NUMERIC検査の場合、一意名-1に以下のデータ項目を指定することはできません。

   ー 英字項目

   ー 符号付き数字項目を従属する集団項目

   ー 日本語項目

   ー 日本語編集項目

   ー ブール項目

4. ALPHABETIC、ALPHABETIC-LOWER、ALPHABETIC-UPPERおよび字類名検査の場合、一意名-1に以下のデータ項目を指定することはできません。

   ー 数字項目

   ー 日本語項目

   ー 日本語編集項目

   ー ブール項目

5. BOOLEAN検査の場合、一意名-1に以下のデータ項目を指定することはできません。

   ー 数字項目

   ー 英字項目

   ー 日本語項目

   ー 日本語編集項目

6. NUMERIC検査が真になる条件は、以下の通りです。

   a. 一意名-1のデータ項目に演算符号を指定しなかった場合、その内容が数字だけからなり、かつ、演算符号がないときです。また、一意名-1が内部10進項目の場合、その符号部分が16進表現のFのときです。

   b. 一意名-1のデータ項目に演算符号を指定した場合、その内容が数字だけからなり、かつ、有効な演算符号が存在するときです。有効な演算符号が存在するときとは、以下の場合です。

   ー 一意名-1のデータ項目のSIGN句にSEPARATE CHARACTERを指定した場合、その符号部分が標準データ形式の“+”または“-”のとき。

   ー 一意名-1が外部10進項目で、そのSIGN句にSEPARATE CHARACTERを指定しなかった場合、その符号部分が16進表現の4、5または3のとき。

   ー 一意名-1が内部10進項目の場合、その符号部分が16進表現のC、D、またはFのとき。

7. NOTを書いた場合、NOTと必要語で1つの字類条件になります。NOTを書くと、真理値の真と偽が、NOTを書かなかった場合の逆になります。

11.8.2.4.2 型条件

型条件は、クラスの継承、インタフェースの実装を検査するために使用します。

【書き方】

一意名-1 IS [NOT] { クラス名-1 / インタフェース名-1 / デリゲート名-1 / ENUM名-1 }

一般規則

1. 型条件では、型検査の指定にしたがって一意名-1の内容を検査し、真理値を決定します。型検査が真になる条件を、以下の表に示します。

表11.4 型検査が真になる条件

型検査の種類	型検査が真になる条件
クラス名-1	一意名-1が保持するオブジェクト参照が、クラス名-1で示されるクラスまたはそのクラスを継承しているクラスのオブジェクトを参照している場合
インタフェース名-1	一意名-1が保持するオブジェクト参照が、インタフェース名-1で示されるインタフェースを実装している場合
デリゲート名-1	一意名-1が保持するオブジェクト参照が、デリゲート名-1で示されるデリゲートである場合
ENUM名-1	一意名-1が保持するオブジェクトが、ENUM名-1で示されるENUMである場合

2. 一意名-1には、オブジェクト参照項目だけ指定できます。
3. NOTを書いた場合、NOTと必要語で1つの型条件になります。NOTを書くと、真理値の真と偽が、NOTを書かなかった場合の逆になります。

11.8.2.5 比較の規則

ここでは、.NETプログラミング機能における、比較の規則について説明します。

比較条件の右辺と左辺の項目による、比較規則の作用対象の組合せを、以下に示します。

表11.5 比較規則(1)

条件の左辺 または右辺	英字項目 英数字項目 英数字編集項目 数字編集項目	英数字関数	外部10進項目	2進項目 内部10進項目	算術式 数字関数 整数関数
英字項目 英数字項目 [ALL]文字定数 16進文字定数 英数字編集項目 数字編集項目	文字比較	文字比較	文字比較	－	－
英数字関数	文字比較	文字比較	文字比較	－	－
外部10進項目	文字比較	文字比較	数字比較	数字比較	数字比較
2進項目 内部10進項目	－	－	数字比較	数字比較	数字比較

条件の左辺 または右辺	英字項目 英数字項目 英数字編集項目 数字編集項目	英数字関数	外部10進項目	2進項目 内部10進項目	算術式 数字関数 整数関数
数字定数 16進数字定数	文字比較		数字比較	数字比較	数字比較
算術式 数字関数 整数関数	―	―	数字比較	数字比較	数字比較
浮動小数点項目 浮動小数点定数	―	―	数字比較	数字比較	数字比較
日本語項目 日本語関数 日本語編集項目 [ALL]日本語定数 日本語16進定数	―	―	―	―	―
ブール項目 [ALL]ブール定数	―	―	―	―	―
集団項目	文字比較	文字比較	文字比較	文字比較	―
指標データ項目					―
オブジェクト参照項目					―
ZERO	文字比較	文字比較	数字比較	数字比較	数字比較
SPACE	文字比較	文字比較	文字比較	―	―
HIGH-VALUE LOW-VALUE	文字比較	文字比較	文字比較	―	―
QUOTE、記号文字	文字比較	文字比較	文字比較	―	―
定義済み EXCEPTION-OBJECT	―	―	―	―	―
定義済みNULL	―	―	―	―	―
定義済みSELF	―	―	―	―	―
ENUM値 ENUM型関数	―	―	―	―	―

表11.6 比較規則(2)

条件の左辺 または右辺	浮動小数 点項目	日本語項目 日本語関数 日本語編集項目	ブール項目	集団項目 (型なしまたは弱い型付け)	指標データ項目
英字項目 英数字項目 [ALL]文字定数 16進文字定数 英数字編集項目 数字編集項目	―	―	―	文字比較	―
英数字関数	―	―	―	文字比較	―

条件の左辺 または右辺	浮動小数 点項目	日本語項目 日本語関数 日本語編集項目	ブール項目	集団項目 (型なしまたは弱い 型付け)	指標データ項目
外部10進項目	数字比較	―	―	文字比較	―
2進項目 内部10進項目	数字比較	―	―	文字比較	―
数字定数 16進数字定数	数字比較	―	―	文字比較	―
算術式 数字関数 整数関数	数字比較	―	―	―	―
浮動小数点項目 浮動小数点定数	数字比較	―	―	文字比較	―
日本語項目 日本語関数 日本語編集項目 [ALL]日本語定数 日本語16進定数	―	日本語文字比較	―	文字比較	―
ブール項目 [ALL]ブール定数	―	―	ブール比較	―	―
集団項目	文字比較	文字比較	―	文字比較	―
指標データ項目	―	―	―	―	指標比較
オブジェクト参照項目	―	―	比較可能(*3)	―	―
ZERO	数字比較	―	ブール比較	文字比較	―
SPACE	―	日本語文字比較	―	文字比較	―
HIGH-VALUE LOW-VALUE	―	日本語文字比較	―	文字比較	―
QUOTE、記号文字	―	―	―	文字比較	―
定義済み EXCEPTION-OBJECT	―	―	―	―	―
定義済みNULL	―	―	―	―	―
定義済みSELF	―	―	―	―	―
ENUM値 ENUM型関数	―	―	―	―	―

表11.7 比較規則(3)

条件の左辺 または右辺	オブジェクト参照項目	定義済み EXCEPTION-OBJECT	定義済みNULL	定義済みSELF	ENUM値 ENUM型関数
英字項目 英数字項目 [ALL]文字定数 16進文字定数	―	―	―	―	―

条件の左辺 または右辺	オブジェクト参照項目	定義済み EXCEPTION- OBJECT	定義済みNULL	定義済みSELF	ENUM値 ENUM型関数
英数字編集項目 数字編集項目					
英数字関数	―	―	―	―	―
外部10進項目	―	―	―	―	―
2進項目 内部10進項目	―	―	―	―	―
数字定数 16進数字定数	―	―	―	―	―
算術式 数字関数 整数関数	―	―	―	―	―
浮動小数点項目 浮動小数点定数	―	―	―	―	―
日本語項目 日本語関数 日本語編集項目 [ALL]日本語定数 日本語16進定数	―	―	―	―	―
ブール項目 [ALL]ブール定数	比較可能(*3)	―	―	―	―
集団項目	―	―	―	―	―
指標データ項目	―	―	―	―	―
オブジェクト参照項目	比較可能(*1)	比較可能(*1)	比較可能(*1)	比較可能(*1)	比較可能(*2)
ZERO	―	―	―	―	―
SPACE	―	―	―	―	―
HIGH-VALUE LOW-VALUE	―	―	―	―	―
QUOTE、記号文字	―	―	―	―	―
定義済み EXCEPTION- OBJECT	比較可能(*1)	―	比較可能	比較可能	―
定義済みNULL	比較可能(*1)	比較可能	―	比較可能	―
定義済みSELF	比較可能(*1)	比較可能	比較可能	―	―
ENUM値 ENUM型関数	比較可能(*2)	―	―	―	比較可能

*1 ： 以下の組合せを除き値型オブジェクト参照項目は条件の左辺および右辺に記述できません。(COBOLのオブジェクト比較は“参照の比較”であり、“内容(値)の比較”ではないため、比較できません。)

- 同じENUM型同士
- System.Boolean型同士

*2 ： 同じENUM型の組合せでなければなりません。

*3 ： System.Boolean型と1桁のブール項目またはブール定数の組合せでなければなりません。

11.8.2.6 適合

ここでは、.NETプログラミング機能における、代入時の適合および引数の適合について説明します。

11.8.2.6.1 代入時の適合

代入時の適合関係について説明します。適合関係は、以下のSET文の設定方法ごとに異なります。

- オブジェクト参照とオブジェクト参照の設定
- オブジェクト参照からオブジェクト参照以外のデータ型への設定
- オブジェクト参照以外のデータ型からオブジェクト参照への設定

代入時の適合関係を、SET文の設定方法ごとに、以降の表に示します。

表11.8 代入時の適合(オブジェクト参照からオブジェクト参照へ)

送出し側		受取り側				
		クラス名		インタフェース名	デリゲート名	ENUM名
		Reference Type	Value Type			
定義済みオブジェクト一意名 NULL		○	×	○	○	×
定義済みオブジェクト一意名 SELF		○(*4)	×	○(*5)	×	×
クラス名	**Reference Type**	○(*1)	○(*1)	○(*2)	×	×
	Value Type	○(*1)	○(*1)	○(*2)	×	×
インタフェース名		○(*8)	×	○(*3)	×	×
デリゲート名		○(*1)	×	×	○(*6)	×
ENUM名		○(*1)	×	×	×	○(*7)

○ ： 代入可能であることを示します。

× ： 構文的には記述可能ですが、規則として許されないことを示します。

*1 ： 送出し側オブジェクトに指定されたクラス、デリゲートおよびENUMは、受取り側オブジェクトに指定されたクラスを継承していなければなりません。

*2 ： 送出し側オブジェクトに指定されたクラスは、受取り側オブジェクトに指定されたインタフェースを実装していなければなりません。

*3 ： 送出し側オブジェクトに指定されたインタフェースは、受取り側オブジェクトに指定されたインタフェースを継承していなければなりません。

*4 ： その文は、クラス名を継承したクラスのオブジェクトメソッド中に現れなければなりません。

*5 ： その文は、インタフェース名を実装したクラスを継承したクラスのオブジェクトメソッド中に現れなければなりません。

*6 ： 同名のデリゲート名でなければなりません。

*7 ： 同名のENUM名でなければなりません。

*8 ： ただし、受取り側のクラス名が“System.Object”でなければなりません。

表11.9 代入時の適合-オブジェクト参照からオブジェクト参照以外のデータ型へ（1／3）

送出し側	受取り側			
	BINARY-CHAR UNSIGNED	BINARY-SHORT SIGNED	BINARY-LONG SIGNED	BINARY-DOUBLE SIGNED
Byte	○	○	○	○
Sbyte	○(*6)	○	○	○
Int16	○(*6)	○	○	○
UInt16	○(*6)	○(*6)	○	○
Int32	○(*6)	○(*6)	○	○
UInt32	○(*6)	○(*6)	○(*6)	○
Int64	○(*6)	○(*6)	○(*6)	○
UInt64	○(*6)	○(*6)	○(*6)	○(*6)
Single	×	○(*10)	○(*10)	○(*10)
Double	×	○(*10)	○(*10)	○(*10)
Decimal	×	○(*2)	○(*2)	○(*2)
Char	×	×	×	×
Boolean	×	×	×	×
String	×	×	×	×
Byte[]	×	×	×	×
Char[]	×	×	×	×
ENUM	○(*7)	○(*7)	○(*7)	○(*7)

表11.10 代入時の適合-オブジェクト参照からオブジェクト参照以外のデータ型へ（2／3）

送出し側	受取り側				
	数字項目				英字項目 英数字項目
	符号なし		符号つき		
	整数	非整数	整数	非整数	
Byte	○(*1)	○(*1)	○(*1)	○(*1)	×
Sbyte	×	×	○(*1)	○(*1)	×
Int16	×	×	○(*1)	○(*1)	×
UInt16	○(*1)	○(*1)	○(*1)	○(*1)	×
Int32	×	×	○(*1)	○(*1)	×
UInt32	○(*1)	○(*1)	○(*1)	○(*1)	×
Int64	×	×	○(*1)	○(*1)	×
UInt64	○(*1)	○(*1)	○(*1)	○(*1)	×
Single	×	×	○(*10)	○(*10)	×
Double	×	×	○(*10)	○(*10)	×
Decimal	×	×	○(*2)	○(*2)	×
Char	×	×	×	×	×
Boolean	×	×	×	×	×
String	×	×	×	×	○(*4)(*5)
Byte[]	×	×	×	×	○

送出し側	受取り側				
	数字項目				英字項目 英数字項目
	符号なし		符号つき		
	整数	非整数	整数	非整数	
Char[]	×	×	×	×	×
ENUM	×	×	○(*1) (*7)	×	×

表11.11 代入時の適合-オブジェクト参照からオブジェクト参照以外のデータ型へ（3／3）

送出し側	受取り側						
	日本語項目	集団項目(型なし、弱い型)	外部浮動小数点項目	内部浮動小数点		ブール項目	
				単精度	倍精度	内部	外部
Byte	×	×	×	×	×	×	×
Sbyte	×	×	×	×	×	×	×
Int16	×	×	×	×	×	×	×
UInt16	×	×	×	×	×	×	×
Int32	×	×	×	×	×	×	×
UInt32	×	×	×	×	×	×	×
Int64	×	×	×	×	×	×	×
UInt64	×	×	×	×	×	×	×
Single	×	×	○(*8)	○(*8)	○(*8)	×	×
Double	×	×	○(*8)	×	○(*8)	×	×
Decimal	×	×	×	×	×	×	×
Char	○(*3)	×	×	×	×	×	×
Boolean	×	×	×	×	×	○(*3)	○(*3)
String	○(*5)(*9)	×	×	×	×	×	×
Byte[]	×	○	×	×	×	×	×
Char[]	○(*9)	×	×	×	×	×	×
ENUM	×	×	×	×	×	×	×

○ ： 代入可能であることを示します。

× ： 構文的には記述可能ですが、規則として許されないことを示します。

*1 ： 格納可能分の桁数があれば、代入可能です。

*2 ： 指定可能ですが、実行時桁数不足の場合には、CLR例外が発生します。

*3 ： 1桁、または、1文字だけ代入可能です。なお、送出し側がchar型、かつ、実行時データのコード系をシフトJISとする場合、UCS-2からシフトJISコードへの変換になります。

*4 ： UCS-2からUTF-8コードへの変換となります。ただし、実行時データのコード系をシフトJISとする場合、または、実行時データのコード系を、英数字字類はシフトJIS、日本語字類はUnicodeとする場合は、UCS-2からシフトJISコードへの変換になります。

*5 ： 受取り側の長さより送出し側の長さが長い部分には、空白がつめられます。送出し側がNULLの場合は、空白が代入されます。

*6 ： 送出し側の受取り側に重なるバイト位置部分のバイトが、そのまま受取り側にコピーされます。

*7 ： 代入の規則はENUMのbasetypeに依存します。

*8 ： SingleはUSAGE COMP-1、DoubleはUSAGE COMP-2とみなし、同じ転記の関係をMOVE文で書いたときと同じ規則が適用されます。

*9 ： 実行時データのコード系をシフトJISとする場合、UCS-2からシフトJISコードへの変換になります。

*10 ： 送出し側を浮動小数点項目とみなした転記と同じ規則が適用されます。

表11.12 代入時の適合-オブジェクト参照以外のデータ型からオブジェクト参照へ(1)

送出し側			受取り側					
			SByte	Byte	Int16	UInt16	Int32	Uint32
BINARY-CHAR UNSIGNED			○(*12)	○	○	○	○	○
BINARY-SHORT SIGNED			○(*12)	○(*12)	○	○(*11)	○	○(*11)
BINARY-LONG SIGNED			○(*12)	○(*12)	○(*12)	○(*12)	○	○(*11)
BINARY-DOUBLE SIGNED			○(*12)	○(*12)	○(*12)	○(*12)	○(*12)	○(*12)
数字項目	符号なし	整数	○	○	○	○	○	○
		非整数	×	×	×	×	×	×
	符号あり	整数	○	×	○	×	○	×
		非整数	×	×	×	×	×	×
英字項目			×	×	×	×	×	×
英数字項目、英数字関数			×	×	×	×	×	×
外部浮動小数点項目			×	×	×	×	×	×
内部浮動小数点項目		単精度	×	×	×	×	×	×
		倍精度	×	×	×	×	×	×
日本語項目、日本語関数			×	×	×	×	×	×
内部ブール項目			×	×	×	×	×	×
外部ブール項目			×	×	×	×	×	×
数字編集項目			×	×	×	×	×	×
英数字編集項目			×	×	×	×	×	×
日本語編集項目			×	×	×	×	×	×
集団項目(型なし、弱い型)			×	×	×	×	×	×
指標データ項目			×	×	×	×	×	×
ポインタデータ項目			×	×	×	×	×	×
固定小数点定数	符号なし	整数	○(*1)	○(*1)	○(*1)	○(*1)	○(*1)	○(*1)
		非整数	×	×	×	×	×	×
	符号あり	整数	○(*1)	×	○(*1)	×	○(*1)	×
		非整数	×	×	×	×	×	×
16進数字定数			○(*1)	○(*1)	○(*1)	○(*1)	○(*1)	○(*1)
浮動小数点定数			×	×	×	×	×	×
文字定数、16進文字定数			×	×	×	×	×	×
日本語定数、日本語16進定数			×	×	×	×	×	×
ブール定数			×	×	×	×	×	×
ZERO			○	○	○	○	○	○
SPACE			×	×	×	×	×	×

送出し側	受取り側					
	SByte	Byte	Int16	UInt16	Int32	Uint32
HIGH-VALUE、LOW-VALUE	×	×	×	×	×	×
QUOTE、記号文字	×	×	×	×	×	×

表11.13 代入時の適合-オブジェクト参照以外のデータ型からオブジェクト参照へ(2)

送出し側			受取り側					
			Int64	UInt64	Single	Double	Decimal	Char
BINARY-CHAR UNSIGNED			○	○	○(*14)	○(*14)	○	×
BINARY-SHORT SIGNED			○	○(*11)	○(*14)	○(*14)	○	×
BINARY-LONG SIGNED			○	○(*11)	○(*14)	○(*14)	○	×
BINARY-DOUBLE SIGNED			○	○(*11)	○(*14)	○(*14)	○	×
数字項目	**符号なし**	**整数**	○	○	○(*14)	○(*14)	○	×
		非整数	×	×	○(*14)	○(*14)	○	×
	符号あり	**整数**	○	×	○(*14)	○(*14)	○	×
		非整数	×	×	○(*14)	○(*14)	○	×
英字項目			×	×	×	×	×	×
英数字項目、英数字関数			×	×	×	×	×	×
外部浮動小数点項目			×	×	○(*14)	○(*14)	×	×
内部浮動小数点項目		**単精度**	×	×	○(*14)	○(*14)	×	×
		倍精度	×	×	×	○(*14)	×	×
日本語項目、日本語関数			×	×	×	×	×	○(*2)(*15)
内部ブール項目			×	×	×	×	×	×
外部ブール項目			×	×	×	×	×	×
数字編集項目			×	×	×	×	×	×
英数字編集項目			×	×	×	×	×	×
日本語編集項目			×	×	×	×	×	×
集団項目(型なし、弱い型)			×	×	×	×	×	×
指標データ項目			×	×	×	×	×	×
ポインタデータ項目			×	×	×	×	×	×
固定小数点定数	**符号なし**	**整数**	○(*1)	○(*1)	○(*14)	○(*14)	○(*13)	×
		非整数	×	×	○(*14)	○(*14)	○(*13)	×
	符号あり	**整数**	○(*1)	×	○(*14)	○(*14)	○(*13)	×
		非整数	×	×	○(*14)	○(*14)	○(*13)	×
16進数字定数			○(*1)	○(*1)	○(*14)	○(*14)	○(*13)	×
浮動小数点定数			×	×	○(*14)	○(*14)	×	×
文字定数、16進文字定数			×	×	×	×	×	×
日本語定数、日本語16進定数			×	×	×	×	×	○(*2)(*15)
ブール定数			×	×	×	×	×	×
ZERO			○	○	○	○	○	×

送出し側	受取り側					
	Int64	UInt64	Single	Double	Decimal	Char
SPACE	×	×	×	×	×	○
HIGH-VALUE、LOW-VALUE	×	×	×	×	×	×
QUOTE、記号文字	×	×	×	×	×	×

表11.14 代入時の適合-オブジェクト参照以外のデータ型からオブジェクト参照へ(3)

送出し側			受取り側					
			Boolean	String	Byte[]	Char[]	ENUM	Object
BINARY-CHAR UNSIGNED			×	×	×	×	×	○(*5)
BINARY-SHORT SIGNED			×	×	×	×	×	○(*5)
BINARY-LONG SIGNED			×	×	×	×	×	○(*5)
BINARY-DOUBLE SIGNED			×	×	×	×	×	○(*5)
数字項目	**符号なし**	**整数**	×	×	×	×	×	○(*6)
		非整数	×	×	×	×	×	○(*8)
	符号あり	**整数**	×	×	×	×	×	○(*6)
		非整数	×	×	×	×	×	○(*8)
英字項目			×	○(*3)(*4)	○	○(*4)	×	○(*3)(*4)(*9)
英数字項目、英数字関数			×	○(*3)(*4)	○	○(*4)	×	○(*3)(*4)(*9)
外部浮動小数点項目			×	×	×	×	×	○(*8)
内部浮動小数点項目		**単精度**	×	×	×	×	×	○(*7)
		倍精度	×	×	×	×	×	○(*8)
日本語項目、日本語関数			×	○(*3)(*15)	×	○(*15)	×	○(*3)(*9)(*15)
内部ブール項目			○(*2)	×	×	×	×	○(*2)(*10)
外部ブール項目			○(*2)	×	×	×	×	○(*2)(*10)
数字編集項目			×	○(*4)	×	×	×	○(*4)(*9)
英数字編集項目			×	○(*4)	×	×	×	○(*4)(*9)
日本語編集項目			×	○(*15)	×	×	×	○(*9)(*15)
集団項目(型なし、弱い型)			×	×	○	×	×	×
指標データ項目			×	×	×	×	×	×
ポインタデータ項目			×	×	×	×	×	×
固定小数点定数	**符号なし**	**整数**	×	×	×	×	×	○(*6)
		非整数	×	×	×	×	×	○(*8)
	符号あり	**整数**	×	×	×	×	×	○(*6)
		非整数	×	×	×	×	×	○(*8)
16進数字定数			×	×	×	×	×	○(*6)
浮動小数点定数			×	×	×	×	×	○(*8)
文字定数、16進文字定数			×	○(*4)	○	○(*4)	×	○(*4)(*9)

送出し側	受取り側					
	Boolean	String	Byte[]	Char[]	ENUM	Object
日本語定数、日本語16進定数	×	○(*15)	×	○(*15)	×	○(*9)(*15)
ブール定数	○(*2)	×	×	×	×	○(*2)(*10)
ZERO	×	×	×	×	×	×
SPACE	×	×	×	×	×	×
HIGH-VALUE、LOW-VALUE	×	×	×	×	×	×
QUOTE、記号文字	×	×	×	×	×	×

○ ： 代入可能であることを示します。

× ： 構文的には記述可能ですが、規則として許されないことを示します。

*1 ： 格納可能な値であれば、代入可能です。

*2 ： 1桁、または、1文字だけ代入可能です。

*3 ： 後ろの空白は削除され、すべて空白の場合には長さゼロのStringオブジェクトが設定されます。

*4 ： UTF-8からUCS-2コードへの変換となります。ただし、実行時データのコード系をシフトJISとする場合、または、実行時データのコード系を、英数字字類はシフトJIS、日本語字類はUnicodeとする場合は、シフトJISからUCS-2コードへの変換になります。

*5 ： BINARY-CHAR UNSIGNED→System.Byte、BINARY-SHORT SIGNED→System.Int16、BINARY-LONG SIGNED→System.Int32、BINARY-DOUBLE SIGNED→System.Int64として扱われて、代入されます。

*6 :: 桁数により、以下の型として扱われて、代入されます。

桁数4以下	System.Int16
桁数5以上9以下	System.Int32
桁数10以上	System.Int64

*7 ： System.Singleとして扱われて、代入されます。

*8 ： System.Doubleとして扱われて、代入されます。

*9 ： N(1)の場合は、System.Charとして扱われ、それ以外の場合は、System.Stringとして扱われて、代入されます。なお、実行時データのコード系がシフトJISの場合は、シフトJISからUCS-2コードへの変換後に、この規則が適用されます。

*10 ： System.Booleanとして扱われて、代入されます。

*11 ： 受取り側をsignedとみなして転記が実施されます。たとえば受取り側がUint32，送出し側がInt16で値が-1(X“FFFF”)の場合、受け側には4294967295(X“FFFFFFFF”)が格納されます。

*12 ： 送出し側の受取り側に重なるバイト位置部分のバイトが、そのまま受け側にコピーされます。

*13 ： 送出し側の小数点位置を最右端とみなした時の値がInt64の範囲になければなりません。たとえば送出し側が999999.9999999000000の場合、小数点位置を最右端とみなした時の値9999999999999000000はInt64の範囲を超えるため、翻訳エラーとなります。

*14 ： SingleはUSAGE COMP-1、DoubleはUSAGE COMP-2とみなし、同じ転記の関係をMOVE文で書いたときと同じ規則が適用されます。

*15 ： 実行時データのコード系をシフトJISとする場合、シフトJISからUCS-2コードへの変換になります。

11.8.2.6.2 引数の適合

実引数と仮引数の適合関係について説明します。適合関係は、引数の渡し方ごとに異なります。

・ BY REFERENCEで渡される実引数と仮引数の適合

・ BY CONTENTまたはBY VALUEで渡される実引数と仮引数の適合

BY CONTENTまたはBY VALUEで渡されるパラメタの受渡しには、実引数を送り側、仮引数を受け側として以下の文の規則が適用されます。

― 実引数または仮引数がオブジェクト参照の場合、SET文

― 仮引数が、指標、ポインタのいずれかの場合、SET文

― 上記以外、かつ仮引数が数字の場合、ROUNDED指定の無いCOMPUTE文

― その他の場合、MOVE文

ただし、組合せによっては特別な規則が適用されることがあります。詳細については、“表11.21 引数の適合-BY CONTENTとBY VALUE(9)”を参照して下さい。

引数の適合関係を、引数の渡しかたごとに、以降の表に示します。

表11.15 引数の適合-BY REFERENCE(1)

送出し側		受取り側					
		Int8	UInt8	Int16	UInt16	Int32	Uint32
CLRデータ型	**BINARY-CHAR UNSIGNED**	×	○	×	×	×	×
	BINARY-SHORT SIGNED	×	×	○	×	×	×
	BINARY-LONG SIGNED	×	×	×	×	○	×
	BINARY-DOUBLE SIGNED	×	×	×	×	×	×
	S9(4) COMP-5	×	×	○	×	×	×
	S9(9) COMP-5	×	×	×	×	○	×
	S9(18) COMP-5	×	×	×	×	×	×
	COMP-1	×	×	×	×	×	×
	COMP-2	×	×	×	×	×	×
	N(1) [DISPLAY]	×	×	×	×	×	×
	OBJECT REFERENCE	○(*1)	○(*1)	○(*1)	○(*1)	○(*1)	○(*1)

表11.16 引数の適合-BY REFERENCE(2)

送出し側		受取り側					
		Int8	UInt8	Int16	UInt16	Int32	Uint32
CLRデータ型	**BINARY-CHAR UNSIGNED**	×	×	×	×	×	×
	BINARY-SHORT SIGNED	×	×	×	×	×	×
	BINARY-LONG SIGNED	×	×	×	×	×	×
	BINARY-DOUBLE SIGNED	○	×	×	×	×	×
	S9(4) COMP-5	×	×	×	×	×	×
	S9(9) COMP-5	×	×	×	×	×	×
	S9(18) COMP-5	○	×	×	×	×	×

送出し側		受取り側					
		Int8	UInt8	Int16	UInt16	Int32	Uint32
	COMP-1	×	×	○	×	×	×
	COMP-2	×	×	×	○	×	×
	N(1) [DISPLAY]	×	×	×	×	○	×
	OBJECT REFERENCE	○(*1)	○(*1)	○(*1)	○(*1)	○(*1)	○(*1)

表11.17 引数の適合-BY REFERENCE(3)

送出し側		受取り側					
		Int8	UInt8	Int16	UInt16	Int32	Uint32
CLRデータ型	**BINARY-CHAR UNSIGNED**	×	×	×	×	×	×
	BINARY-SHORT SIGNED	×	×	×	×	×	×
	BINARY-LONG SIGNED	×	×	×	×	×	×
	BINARY-DOUBLE SIGNED	×	×	×	×	×	×
	S9(4) COMP-5	×	×	×	×	×	×
	S9(9) COMP-5	×	×	×	×	×	×
	S9(18) COMP-5	×	×	×	×	×	×
	COMP-1	×	×	×	×	×	×
	COMP-2	×	×	×	×	×	×
	N(1) [DISPLAY]	×	×	×	○(*6)	×	×
	OBJECT REFERENCE	○(*2)	○(*2)	○(*1)	○(*1)	○(*2)	×

表11.18 引数の適合-BY REFERENCE(4)

送出し側			受取り側					
			Int8	UInt8	Int16	UInt16	Int32	Uint32
COBOL独自型 (*8)	埋め込みCLR型	**BINARY-CHAR UNSIGNED**	×	○(*3)	×	×	×	×
		BINARY-SHORT SIGNED	×	×	○(*3)	×	×	×
		BINARY-LONG SIGNED	×	×	×	×	○(*3)	×
		BINARY-DOUBLE SIGNED	×	×	×	×	×	×
		S9(4) COMP-5	×	×	○(*3)	×	×	×
		S9(9) COMP-5	×	×	×	×	○(*3)	×
		S9(18) COMP-5	×	×	×	×	×	×
		COMP-1	×	×	×	×	×	×
		COMP-2	×	×	×	×	×	×
		N(1) [DISPLAY]	×	×	×	×	×	×

送出し側					受取り側					
					Int8	UInt8	Int16	UInt16	Int32	Uint32
	数字項目	2進項目	整数	符号なし	×	×	×	×	×	×
				符号あり	×	×	×	×	×	×
			非整数	符号なし	×	×	×	×	×	×
				符号あり	×	×	×	×	×	×
		内部10進項目	整数	符号なし	×	×	×	×	×	×
				符号あり	×	×	×	×	×	×
			非整数	符号なし	×	×	×	×	×	×
				符号あり	×	×	×	×	×	×
		外部10進項目	整数	符号なし	×	×	×	×	×	×
				符号あり	×	×	×	×	×	×
			非整数	符号なし	×	×	×	×	×	×
				符号あり	×	×	×	×	×	×
	英字項目				×	×	×	×	×	×
	英数字項目				×	×	×	×	×	×
	外部浮動小数点項目				×	×	×	×	×	×
	日本語項目				×	×	×	×	×	×
	ブール項目				×	×	×	×	×	×
	数字編集項目				×	×	×	×	×	×
	英数字編集項目				×	×	×	×	×	×
	日本語編集項目				×	×	×	×	×	×
	集団項目(型なし、弱い型)				×	×	×	×	×	×
	指標データ項目				×	×	×	×	×	×
	ポインタデータ項目				×	×	×	×	×	×

表11.19 引数の適合-BY REFERENCE(5)

送出し側			受取り側					
			Int8	UInt8	Int16	UInt16	Int32	Uint32
COBOL独自型(*8)	埋め込みCLR型	BINARY-CHAR UNSIGNED	×	×	×	×	×	×
		BINARY-SHORT SIGNED	×	×	×	×	×	×
		BINARY-LONG SIGNED	×	×	×	×	×	×
		BINARY-DOUBLE SIGNED	○(*3)	×	×	×	×	×
		S9(4) COMP-5	×	×	×	×	×	×
		S9(9) COMP-5	×	×	×	×	×	×
		S9(18) COMP-5	○(*3)	×	×	×	×	×
		COMP-1	×	×	○(*3)	×	×	×
		COMP-2	×	×	×	○(*3)	×	×
		N(1) [DISPLAY]	×	×	×	×	○(*3)	×

送出し側					受取り側					
					Int8	UInt8	Int16	UInt16	Int32	Uint32
	数字項目	2進項目	整数	符号なし	×	×	×	×	×	×
				符号あり	×	×	×	×	×	×
			非整数	符号なし	×	×	×	×	×	×
				符号あり	×	×	×	×	×	×
		内部10進項目	整数	符号なし	×	×	×	×	×	×
				符号あり	×	×	×	×	×	×
			非整数	符号なし	×	×	×	×	×	×
				符号あり	×	×	×	×	×	×
		外部10進項目	整数	符号なし	×	×	×	×	×	×
				符号あり	×	×	×	×	×	×
			非整数	符号なし	×	×	×	×	×	×
				符号あり	×	×	×	×	×	×
	英字項目				×	×	×	×	×	×
	英数字項目				×	×	×	×	×	×
	外部浮動小数点項目				×	×	×	×	×	×
	日本語項目				×	×	×	×	×	×
	ブール項目				×	×	×	×	×	×
	数字編集項目				×	×	×	×	×	×
	英数字編集項目				×	×	×	×	×	×
	日本語編集項目				×	×	×	×	×	×
	集団項目(型なし、弱い型)				×	×	×	×	×	×
	指標データ項目				×	×	×	×	×	×
	ポインタデータ項目				×	×	×	×	×	×

表11.20 引数の適合-BY REFERENCE(6)

送出し側			受取り側					
			Int8	UInt8	Int16	UInt16	Int32	Uint32
COBOL独自型(*8)	埋め込みCLR型	BINARY-CHAR UNSIGNED	×	×	×	×	×	×
		BINARY-SHORT SIGNED	×	×	×	×	×	×
		BINARY-LONG SIGNED	×	×	×	×	×	×
		BINARY-DOUBLE SIGNED	×	×	×	×	×	×
		S9(4) COMP-5	×	×	×	×	×	×
		S9(9) COMP-5	×	×	×	×	×	×
		S9(18) COMP-5	×	×	×	×	×	×
		COMP-1	×	×	×	×	×	×
		COMP-2	×	×	×	×	×	×
		N(1) [DISPLAY]	×	×	×	×	×	×

送出し側					受取り側					
					Int8	UInt8	Int16	UInt16	Int32	Uint32
	数字項目	2進項目	整数	符号なし	×	×	×	×	×	○(*4)
				符号あり	×	×	×	×	×	○(*4)
			非整数	符号なし	×	×	×	×	×	○(*4)
				符号あり	×	×	×	×	×	○(*4)
		内部10進項目	整数	符号なし	×	×	×	×	×	○(*4)
				符号あり	×	×	×	×	×	○(*4)
			非整数	符号なし	×	×	×	×	×	○(*4)
				符号あり	×	×	×	×	×	○(*4)
		外部10進項目	整数	符号なし	×	×	×	×	×	○(*4)
				符号あり	×	×	×	×	×	○(*4)
			非整数	符号なし	×	×	×	×	×	○(*4)
				符号あり	×	×	×	×	×	○(*4)
	英字項目				×	×	×	○(*6)	×	○(*4)
	英数字項目				×	×	×	○(*6)	×	○(*4)
	外部浮動小数点項目				×	×	×	×	×	○(*4)
	日本語項目				×	×	×	○(*6)	×	○(*4)
	ブール項目				×	×	×	×	×	○(*4)
	数字編集項目				×	×	×	×	×	○(*4)
	英数字編集項目				×	×	×	×	×	○(*4)
	日本語編集項目				×	×	×	×	×	○(*4)
	集団項目(型なし、弱い型)				×	×	×	×	×	○(*5)
	指標データ項目				×	×	×	×	×	○(*7)
	ポインタデータ項目				×	×	×	×	×	○(*9)

○ ： 代入可能であることを示します。

× ： 構文的には記述可能ですが、規則として許されないことを示します。

*1 ： それぞれの型のaliased type(System.xxxx)のインスタンスでなければなりません。

*2 ： 参照する型(クラス・インタフェース・デリゲート・ENUM)そのもののインスタンスでなければなりません。

*3 ： スタティックデータおよびオブジェクトデータはWエラーとなります。暗黙の型変換として埋め込みCLR型変換が適用されます。

*4 ： カスタムアトリビュートの情報が引数の属性と完全に一致しなければなりません。

*5 ： 項目長が同じでなければなりません。

*6 ： スタティックデータおよびオブジェクトデータはWエラーとなります。暗黙の型変換として参照変換が適用されます。参照変換時の内部処理の概要は、以下のようになります。

・ 実行時データのコード系をUnicodeとする場合

英字／英数字項目パラメタの場合	末尾の空白を除去し、UCS-2へ変換した後、呼出し先へ引き渡します。復帰後、UTF-8へ変換し、空白パディングを実施します。
日本語項目パラメタの場合	末尾の空白を除去し、呼出し先へ引き渡します。復帰後、空白パディングを実施します。

・ 実行時データのコード系を、英数字字類はシフトJIS、日本語字類はUnicodeとする場合

英字／英数字項目パラメタの場合	末尾の空白を除去し、UCS-2へ変換した後、呼出し先へ引き渡します。復帰後、シフトJISへ変換し、空白パディングを実施します。
日本語項目パラメタの場合	末尾の空白を除去し、呼出し先へ引き渡します。復帰後、空白パディングを実施します。

・ 実行時データのコード系をシフトJISとする場合

英字／英数字項目パラメタの場合	末尾の空白を除去し、UCS-2へ変換した後、呼出し先へ引き渡します。復帰後、シフトJISへ変換し、空白パディングを実施します。
日本語項目パラメタの場合	末尾の空白を除去し、UCS-2へ変換した後、呼出し先へ引き渡します。復帰後、シフトJISへ変換し、空白パディングを実施します。

参考

実行時データのコード系は、翻訳オプションRCSで指定します。

*7 ： 仮引数も指標データ項目(USAGE INDEX)であった場合です。

*8 ： 英字項目、英数字項目または日本語項目を実引数として渡す場合、適合する候補としてCOBOL独自型の仮引数とString型の仮引数があるならば、COBOL独自型の仮引数をより適合しているとみなします。

*9 ： 仮引数もポインタデータ項目(USAGE POINTER)であった場合です。

表11.21 引数の適合-BY CONTENTとBY VALUE(9)

送出し側		受取り側					
		Int8	UInt8	Int16	UInt16	Int32	Uint32
CLRデータ型	**BINARY-CHAR UNSIGNED**	×	○	○	○	○	○
	BINARY-SHORT SIGNED	×	×	○	×	○	×
	BINARY-LONG SIGNED	×	×	×	×	○	×
	BINARY-DOUBLE SIGNED	×	×	×	×	×	×
	S9(4) COMP-5	×	×	○	×	○	×
	S9(9) COMP-5	×	×	×	×	○	×
	S9(18) COMP-5	×	×	×	×	×	×
	COMP-1	×	×	×	×	×	×
	COMP-2	×	×	×	×	×	×
	N(1) [DISPLAY]	×	×	×	×	×	×
	OBJECT REFERENCE	○(*1)	○(*1)	○(*1)	○(*1)	○(*1)	○(*1)

表11.22 引数の適合-BY CONTENTとBY VALUE(2)

送出し側		受取り側					
		Int64	UInt64	Float32	Float64	Char	Boolean
CLR	**BINARY-CHAR UNSIGNED**	○	○	○	○	×	×

送出し側		受取り側					
		Int64	UInt64	Float32	Float64	Char	Boolean
データ型	**BINARY-SHORT SIGNED**	○	×	○	○	×	×
	BINARY-LONG SIGNED	○	×	○	○	×	×
	BINARY-DOUBLE SIGNED	○	×	○	○	×	×
	S9(4) COMP-5	○	×	○	○	×	×
	S9(9) COMP-5	○	×	○	○	×	×
	S9(18) COMP-5	○	×	○	○	×	×
	COMP-1	×	×	○	○	×	×
	COMP-2	×	×	×	○	×	×
	N(1) [DISPLAY]	×	×	×	×	○	×
	OBJECT REFERENCE	○(*1)	○(*1)	○(*1)	○(*1)	○(*1)	○(*1)

表11.23 引数の適合-BY CONTENTとBY VALUE(3)

送出し側		受取り側					
		Class	Value(含ENUM型)	Object	String	配列型	COBOL独自型
CLRデータ型	**BINARY-CHAR UNSIGNED**	×	×	○	×	○(*4)	×
	BINARY-SHORT SIGNED	×	×	○	×	○(*4)	×
	BINARY-LONG SIGNED	×	×	○	×	○(*4)	×
	BINARY-DOUBLE SIGNED	×	×	○	×	○(*4)	×
	S9(4) COMP-5	×	×	○	×	○(*4)	×
	S9(9) COMP-5	×	×	○	×	○(*4)	×
	S9(18) COMP-5	×	×	○	×	○(*4)	×
	COMP-1	×	×	○	×	○(*4)	×
	COMP-2	×	×	○	×	○(*4)	×
	N(1) [DISPLAY]	×	×	○	○(*8)	○(*4)(*22)	×
	OBJECT REFERENCE	○(*2)	○(*3)	○	○(*1)	○(*3)(*4)	○(*5)

表11.24 引数の適合-BY CONTENTとBY VALUE(4)

送出し側			受取り側					
			Int8	UInt8	Int16	UInt16	Int32	Uint32
COBOL	埋め込みC	**BINARY-CHAR UNSIGNED**	×	○	○	○	○	○
		BINARY-SHORT SIGNED	×	×	○	×	○	×

送出し側					受取り側					
					Int8	UInt8	Int16	UInt16	Int32	Uint32
独自型(*21)	LR型	BINARY-LONG SIGNED			×	×	×	×	○	×
		BINARY-DOUBLE SIGNED			×	×	×	×	×	×
		S9(4) COMP-5			×	×	○	×	○	×
		S9(9) COMP-5			×	×	×	×	○	×
		S9(18) COMP-5			×	×	×	×	×	×
		COMP-1			×	×	×	×	×	×
		COMP-2			×	×	×	×	×	×
		N(1) [DISPLAY]			×	×	×	×	×	×
	数字項目	2進項目	整数	符号なし	○(*6)	○(*6)	○(*6)	○(*6)	○(*6)	○(*6)
				符号あり	○(*6)	×	○(*6)	×	○(*6)	×
			非整数	符号なし	×	×	×	×	×	×
				符号あり	×	×	×	×	×	×
		内部10進項目	整数	符号なし	○(*6)	○(*6)	○(*6)	○(*6)	○(*6)	○(*6)
				符号あり	○(*6)	×	○(*6)	×	○(*6)	×
			非整数	符号なし	×	×	×	×	×	×
				符号あり	×	×	×	×	×	×
		外部10進項目	整数	符号なし	○(*6)	○(*6)	○(*6)	○(*6)	○(*6)	○(*6)
				符号あり	○(*6)	×	○(*6)	×	○(*6)	×
			非整数	符号なし	×	×	×	×	×	×
				符号あり	×	×	×	×	×	×
	英字項目				×	×	×	×	×	×
	英数字項目				×	×	×	×	×	×
	外部浮動小数点項目				×	×	×	×	×	×
	日本語項目				×	×	×	×	×	×
	ブール項目				×	×	×	×	×	×
	数字編集項目				×	×	×	×	×	×
	英数字編集項目				×	×	×	×	×	×
	日本語編集項目				×	×	×	×	×	×
	集団項目(型なし、弱い型)				×	×	×	×	×	×
	指標データ項目				×	×	×	×	×	×
	ポインタデータ項目				×	×	×	×	×	×

表11.25 引数の適合-BY CONTENTとBY VALUE(5)

送出し側					受取り側						
					Int64	UInt64	Float32	Float64	Char	Boolean	Decimal
COBOL独自型 (*21)	埋め込みCLR型	BINARY-CHAR UNSIGNED			○	○	○	○	×	×	×
		BINARY-SHORT SIGNED			○	×	○	○	×	×	×
		BINARY-LONG SIGNED			○	×	○	○	×	×	×
		BINARY-DOUBLE SIGNED			○	×	○	○	×	×	×
		S9(4) COMP-5			○	×	○	○	×	×	×
		S9(9) COMP-5			○	×	○	○	×	×	×
		S9(18) COMP-5			○	×	○	○	×	×	×
		COMP-1			×	×	○	○	×	×	×
		COMP-2			×	×	×	○	×	×	×
		N(1) [DISPLAY]			×	×	×	×	○	×	×
	数字項目	2進項目	整数	符号なし	○(*6)	○(*6)	○(*6)	○(*6)	×	×	○
				符号あり	○(*6)	×	○(*6)	○(*6)	×	×	○
			非整数	符号なし	×	×	○(*6)	○(*6)	×	×	○
				符号あり	×	×	○(*6)	○(*6)	×	×	○
		内部10進項目	整数	符号なし	○(*6)	○(*6)	○(*6)	○(*6)	×	×	○
				符号あり	○(*6)	×	○(*6)	○(*6)	×	×	○
			非整数	符号なし	×	×	○(*6)	○(*6)	×	×	○
				符号あり	×	×	○(*6)	○(*6)	×	×	○
		外部10進項目	整数	符号なし	○(*6)	○(*6)	○(*6)	○(*6)	×	×	○
				符号あり	○(*6)	×	○(*6)	○(*6)	×	×	○
			非整数	符号なし	×	×	○(*6)	○(*6)	×	×	○
				符号あり	×	×	○(*6)	○(*6)	×	×	○
	英字項目				×	×	×	×	×	×	×
	英数字項目				×	×	×	×	×	×	×
	外部浮動小数点項目				×	×	○(*6)	○(*6)	×	×	×
	日本語項目				×	×	×	×	×	×	×
	ブール項目				×	×	×	×	×	○(*11)	×
	数字編集項目				×	×	×	×	×	×	×
	英数字編集項目				×	×	×	×	×	×	×
	日本語編集項目				×	×	×	×	×	×	×
	集団項目(型なし、弱い型)				×	×	×	×	×	×	×
	指標データ項目				×	×	×	×	×	×	×

送出し側		受取り側						
		Int64	UInt64	Float32	Float64	Char	Boolean	Decimal
	ポインタデータ項目	×	×	×	×	×	×	×

表11.26 引数の適合-BY CONTENTとBY VALUE(6)

送出し側					受取り側					
					Class	Value(含ENUM型)	Object	String	配列型	COBOL独自型
COBOL独自型(*21)	埋め込みCLR型	BINARY-CHAR UNSIGNED			×	×	○	×	○(*4)	×
		BINARY-SHORT SIGNED			×	×	○	×	○(*4)	×
		BINARY-LONG SIGNED			×	×	○	×	○(*4)	×
		BINARY-DOUBLE SIGNED			×	×	○	×	○(*4)	×
		S9(4) COMP-5			×	×	○	×	○(*4)	×
		S9(9) COMP-5			×	×	○	×	○(*4)	×
		S9(18) COMP-5			×	×	○	×	○(*4)	×
		COMP-1			×	×	○	×	○(*4)	×
		COMP-2			×	×	○	×	○(*4)	×
		N(1) [DISPLAY]			×	×	○	×	○(*4)(*22)	×
	数字項目	2進項目	整数	符号なし	×	×	○	×	×	○(*7)
				符号あり	×	×	○	×	×	○(*7)
			非整数	符号なし	×	×	○	×	×	○(*7)
				符号あり	×	×	○	×	×	○(*7)
		内部10進項目	整数	符号なし	×	×	○	×	×	○(*7)
				符号あり	×	×	○	×	×	○(*7)
			非整数	符号なし	×	×	○	×	×	○(*7)
				符号あり	×	×	○	×	×	○(*7)
		外部10進項目	整数	符号なし	×	×	○	×	×	○(*7)
				符号あり	×	×	○	×	×	○(*7)
			非整数	符号なし	×	×	○	×	×	○(*7)
				符号あり	×	×	○	×	×	○(*7)
		英字項目			×	×	○	○(*8)	○(*9)	○(*7)
		英数字項目			×	×	○	○(*8)	○(*9)	○(*7)
		外部浮動小数点項目			×	×	○	×	×	○(*7)
		日本語項目			×	×	○	○(*8)	○(*10)	○(*7)
		ブール項目			×	×	○(*11)	×	×	○(*7)

送出し側		受取り側					
		Class	Value(含ENUM型)	Object	String	配列型	COBOL独自型
	数字編集項目	×	×	○	×	×	○(*7)
	英数字編集項目	×	×	○	○(*8)	×	○(*7)
	日本語編集項目	×	×	○	○(*8)	×	○(*7)
	集団項目(型なし、弱い型)	×	×	×	×	○(*9)	○(*7)
	指標データ項目	×	×	×	×	×	○(*7)
	ポインタデータ項目	×	×	×	×	×	○(*7)

表11.27 引数の適合-BY CONTENTとBY VALUE(7)

送出し側		受取り側					
		Int8	UInt8	Int16	UInt16	Int32	Uint32
定数(*21)	固定小数点定数(正整数)	×	×	○(*12)	×	○(*12)	×
	固定小数点定数(負整数)	×	×	○(*12)	×	○(*12)	×
	固定小数点定数(小数部有)	×	×	×	×	×	×
	浮動小数点定数(単精度)	×	×	×	×	×	×
	浮動小数点定数(倍精度)	×	×	×	×	×	×
	16進数字定数	×	×	○(*12)	×	○(*12)	×
	文字定数	×	×	×	×	×	×
	16進文字定数	×	×	×	×	×	×
	日本語文字定数	×	×	×	×	×	×
	日本語16進文字定数	×	×	×	×	×	×
	ブール定数	×	×	×	×	×	×
表意定数	ZERO	○	○	○	○	○	○
	SPACE	×	×	×	×	×	×
	NULL	×	×	×	×	×	×
	HIGH-VALUE	×	×	×	×	×	×
	LOW-VALUE	×	×	×	×	×	×
	QUOTE	×	×	×	×	×	×
	ALL定数	×	×	×	×	×	×
	記号文字	×	×	×	×	×	×
ENUMフィールド参照(*18)		○(*19)	○(*19)	○(*19)	○(*19)	○(*19)	○(*19)
TYPE OF特殊レジスタ(*18)		×	×	×	×	×	×

表11.28 引数の適合-BY CONTENTとBY VALUE(8)

送出し側		受取り側						
		Int64	UInt64	Float32	Float64	Char	Boolean	Decimal
定数(*21)	固定小数点定数(正整数)	○(*12)	×	○(*12)	○(*12)	×	×	×
	固定小数点定数(負整数)	○(*12)	×	○(*12)	○(*12)	×	×	×

送出し側		受取り側						
		Int64	UInt64	Float32	Float64	Char	Boolean	Decimal
	固定小数点定数(小数部有)	×	×	○(*13)	○(*13)	×	×	○(*13)
	浮動小数点定数(単精度)	×	×	○(*14)	○(*14)	×	×	×
	浮動小数点定数(倍精度)	×	×	○(*14)	○(*14)	×	×	×
	16進数字定数	○(*12)	×	○(*12)	○(*12)	×	×	×
	文字定数	×	×	×	×	×	×	×
	16進文字定数	×	×	×	×	×	×	×
	日本語文字定数	×	×	×	×	○(*15)	×	×
	日本語16進文字定数	×	×	×	×	○(*15)	×	×
	ブール定数	×	×	×	×	×	○(*11)	×
表意定数	ZERO	○	○	○	○	×	×	○
	SPACE	×	×	×	×	○	×	×
	NULL	×	×	×	×	×	×	×
	HIGH-VALUE	×	×	×	×	×	×	×
	LOW-VALUE	×	×	×	×	×	×	×
	QUOTE	×	×	×	×	×	×	×
	ALL定数	×	×	×	×	×	×	×
	記号文字	×	×	×	×	×	×	×
ENUMフィールド参照(*18)		○(*19)	○(*19)	○(*19)	○(*19)	×	×	×
TYPE OF特殊レジスタ(*18)		×	×	×	×	×	×	×

表11.29 引数の適合-BY CONTENTとBY VALUE(9)

送出し側		受取り側					
		Int8	UInt8	Int16	UInt16	Int32	Uint32
定数(*21)	固定小数点定数(正整数)	×	×	○	×	×	○(*7)
	固定小数点定数(負整数)	×	×	○	×	×	○(*7)
	固定小数点定数(小数部有)	×	×	○	×	×	○(*7)
	浮動小数点定数(単精度)	×	×	○	×	×	○(*7)
	浮動小数点定数(倍精度)	×	×	○	×	×	○(*7)
	16進数字定数	×	×	○	×	×	○(*7)
	文字定数	×	×	○	○(*8)	○(*9)	○(*7)
	16進文字定数	×	×	○	○(*8)	○(*9)	○(*7)
	日本語文字定数	×	×	○	○(*8)	○(*9)	○(*7)
	日本語16進文字定数	×	×	○	○(*8)	○(*10)	○(*7)
	ブール定数	×	×	○(*11)	×	×	○(*7)
表意定数	ZERO	×	×	×	×	×	○(*17)
	SPACE	×	×	×	×	×	○(*17)
	NULL	○	×	○	○	○(*16)	○(*17)
	HIGH-VALUE	×	×	×	×	×	○(*17)

送出し側		受取り側					
		Int8	UInt8	Int16	UInt16	Int32	Uint32
	LOW-VALUE	×	×	×	×	×	○(*17)
	QUOTE	×	×	×	×	×	○(*17)
	ALL定数	×	×	×	×	×	×
	記号文字	×	×	×	×	×	×
ENUMフィールド参照(*18)		×	○(*3)	×	×	×	×
TYPE OF特殊レジスタ(*18)		○(*20)	×	×	×	×	×

○ ： 代入可能であることを示します。

× ： 構文的には記述可能ですが、規則として許されないことを示します。

*1 ： それぞれの型のaliased type(System.xxxx)のインスタンスでなければなりません。

*2 ： オブジェクトの参照する型、または、それから派生する型のインスタンスでなければなりません。

*3 ： オブジェクトの参照する型のインスタンスでなければなりません。

*4 ： 仮引数側がparams指定を持ち、実引数の型のaliased type(System.xxxx)の1次元配列である場合だけです。ただし、仮引数がSystem. Arrayである場合は、多次元の実引数も可能です。

*5 ： String型から、英字、英数字または日本語の場合だけ可能です。受渡しは、実行時データのコード系によって以下のように行なわれます。

・ 実行時データのコード系をUnicodeとする場合

受取り側が英字／英数字項目の場合	UTF-8へ変換し、空白パディングを実施します。
受取り側が日本語項目の場合	空白パディングを実施します。

・ 実行時データのコード系を、英数字字類はシフトJIS、日本語字類はUnicodeとする場合

受取り側が英字／英数字項目の場合	シフトJISへ変換し、空白パディングを実施します。
受取り側が日本語項目パラメタの場合	空白パディングを実施します。

・ 実行時データのコード系をシフトJISとする場合

受取り側が英字／英数字項目の場合	シフトJISへ変換し、空白パディングを実施します。
受取り側が日本語項目の場合	シフトJISへ変換し、空白パディングを実施します。

参考

実行時データのコード系は、翻訳オプションRCSで指定します。

*6 ： 受取り側にあわせ、暗黙の型変換として整数変換を適用します。桁あふれが生じる場合、切り捨てられます。

*7 ： 実引数を送出し側、仮引数を受取り側としたとき、以下の表で転記が可能な組合せ(○)でなければなりません。

実引数	仮引数										
	集団項目	ブール	符号付整数	符号無整数	非整数	数字編集	英数字	英字	日本語	指標	ポインタ
集団項目	○	×	○	○	○	○	○	○	○	×	×
ブール	○	○	×	×	×	×	○	×	×	×	×
符号付整数	○	×	○	○	○	○	○	×	×	×	×
符号無整数	○	×	○	○	○	○	○	×	×	×	×

実引数	仮引数										
	集団項目	ブール	符号付整数	符号無整数	非整数	数字編集	英数字	英字	日本語	指標	ポインタ
非整数	×	×	○	○	○	○	×	×	×	×	×
数字編集	○	×	○	○	○	○	○	×	×	×	×
英数字	○	×	○	○	○	○	○	○	×	×	×
英字	○	×	×	×	×	×	○	○	×	×	×
日本語	×	×	×	×	×	×	×	×	○	×	×
指標	×	×	×	×	×	×	×	×	×	○	×
ポインタ	×	×	×	×	×	×	×	×	×	×	○

*8 ： 引数は、以下のように渡されます。

・ 実行時データのコード系をUnicodeとする場合、または、実行時データのコード系を、英数字字類はシフトJIS、日本語字類はUnicodeとする場合

英字項目または英数字項目からString	末尾の空白を除去し、UCS-2へ変換した後、呼出し先へ引き渡します。
日本語項目からString	末尾の空白を除去し、呼出し先へ引き渡します。
文字定数または16進文字定数からString	UCS-2へ変換し、呼出し先へ引き渡します。
日本語定数または日本語16進定数からString	そのまま呼出し先へ引き渡します。

・ 実行時データのコード系をシフトJISとする場合

英字項目または英数字項目からString	末尾の空白を除去し、UCS-2へ変換した後、呼出し先へ引き渡します。
日本語項目からString	末尾の空白を除去し、UCS-2へ変換した後、呼出し先へ引き渡します。
文字定数または16進文字定数からString	UCS-2へ変換し、呼出し先へ引き渡します。
日本語定数または日本語16進定数からString	UCS-2へ変換し、呼出し先へ引き渡します。

*9 ： 仮引数がChar[]またはByte[]でなければなりません。引数は、以下のように渡されます。

英字項目、英数字項目、または集団項目からByte[]	末尾の空白は考慮しないで、領域全体をByte配列に変換し、呼出し先へ引き渡します。
文字定数または16進文字定数からByte[]	末尾の空白は考慮しないで、内容全体をByte配列に変換し、呼出し先へ引き渡します。
英字項目または英数字項目からChar[]	末尾の空白は考慮しないで、領域全体をUCS-2変換したものをChar配列に変換し、呼出し先へ引き渡します。
文字定数または16進文字定数からChar[]	末尾の空白は考慮しないで、内容全体をUCS-2変換したものをChar配列に変換し、呼出し先へ引き渡します。

*10 ： 仮引数がChar[]でなければなりません。引数は、以下のように渡されます。

・ 実行時データのコード系をUnicodeとする場合、または、実行時データのコード系を、英数字字類はシフトJIS、日本語字類はUnicodeとする場合

日本語項目からChar[]	末尾の空白は考慮しないで、領域全体をChar配列に変換し、呼出し先へ引き渡します。
日本語定数または日本語16進定数からChar[]	末尾の空白は考慮しないで、内容全体をChar配列に変換し、呼出し先へ引き渡します。

・ 実行時データのコード系をシフトJISとする場合

日本語項目からChar[]	末尾の空白は考慮しないで、領域全体をUCS-2変換したものをChar配列に変換し、呼出し先へ引き渡します。
日本語定数または日本語16進定数からChar[]	末尾の空白は考慮しないで、内容全体をUCS-2変換したものをChar配列に変換し、呼出し先へ引き渡します。

*11 ： 長さが1の場合だけ可能です。

*12 ： 整数は符号の有無に関係なく符号付きとみなして、値の範囲により、次のCLR型にマッピングします。ただし、仮引数がCOBOL独自型の場合は、符号付き整数項目として扱います。

値の範囲	CLR型
-32,768 ～ 32,767	Int16
-2,147,483,648 ～ 2,147,483,647	Int32
上記以外	Int64

*13 ： 非整数定数は、すべてCOBOL独自型とみなします。

*14 ： 浮動小数点定数は、すべてFloat32とみなします。

*15 ： 1文字の場合だけ可能です。

*16 ： 仮引数の次元数が多次元であっても可能です。

*17 ： 実引数を送出し側、仮引数を受取り側としたとき、以下の表で転記が可能な組合せ(○)でなければなりません。

実引数	仮引数										
	集団項目	ブール	符号付整数	符号無整数	非整数	数字編集	英数字	英字	日本語	指標	ポインタ
ZERO	○	○	○	○	○	○	○	○	×	×	×
SPACE	○	×	×	×	×	×	○	○	○	×	×
NULL	×	×	×	×	×	×	×	×	×	×	○
HIGH-VALUE	○	×	×	×	×	×	○	○	○	×	×
LOW-VALUE	○	×	×	×	×	×	○	○	○	×	×
QUOTE	○	×	×	×	×	×	○	○	×	×	×

*18 ： 環境部の特殊名段落の、CUSTOM-ATTRIBUTE句のUSINGおよびPROPERTY指定だけ可能です。INVOKE文およびメソッドの行内呼出しには指定できません。

*19 ： ENUMのbasetypeによって適合する型は異なります。

*20 ： System.Type型でなければなりません。

*21 ： COBOL数字型、COBOL文字型、数字定数または文字定数を実引数として渡す場合、適合する候補としてCOBOL独自型の仮引数と転記可能なCLR型の仮引数があるならば、COBOL独自型の仮引数をより適合しているとみなします。

*22 ： 実行時データのコード系をシフトJISとする場合、UCS-2に変換し、呼出し先へ引き渡します。

11.8.2.6.3 引数のより適切な適合

BY CONTENTまたはBY VALUEで渡される実引数と仮引数の適合では、1つの実引数の型に対して、複数の仮引数の型が適合します。このとき、複数の仮引数の型の間でより適切な適合が定義できる場合があります。

実引数の型T1に対して、適合する引数の型T2とT3があるとき、次のいずれかの条件を満たすなら、より適切な適合が決定できます。

1. T1とT2が一致するなら、T1はT3よりT2により適切に適合する。
2. T1とT3が一致するなら、T1はT2よりT3により適切に適合する。

3. T2はT3に適合するが、T3はT2に適合しない場合、T1はT3よりT2により適切に適合する。
4. T3はT2に適合するが、T2はT3に適合しない場合、T1はT2よりT3により適切に適合する。

これ以外の場合、実引数の型T1と、それに適合する仮引数の型T2およびT3のどちらが適切に適合するか決定することはできません。

11.8.2.7 例外条件

ここでは、.NETプログラミング機能における、例外条件について説明します。

例外条件は、例外オブジェクトです。例外オブジェクトは、Exceptionクラスを継承するクラスのオブジェクトインスタンスでなければなりません。

例外オブジェクト

TRY文の中以外で例外オブジェクトが発生した場合、以下のどちらかの事象が引き起こされます。なお、TRY文の中で例外オブジェクトが発生した場合については、“11.8.3.18 TRY文”を参照してください。

1. プログラム中のUSE文に、例外オブジェクトに適合するクラスが指定されている場合、関連付けられた宣言部分が実行されます。宣言部分の実行が正常に成功した場合(EXIT PROGRAM、STOP RUN、実行単位を終了させるプログラム、メソッドを参照するCALL、または、メソッドを参照するINVOKEがない)、暗黙のEXIT PROGRAM、または、EXIT METHODが実行されます。
2. ソース単位内に適用できるUSE文がない場合で、例外オブジェクトがSystem.Exceptionを継承するクラスのオブジェクトのとき、その例外オブジェクトをRAISING指定に持つ、暗黙のEXIT PROGRAM、または、EXIT METHODが実行されます。

11.8.3 文

ここでは、.NETプログラミング機能における、以下の文について説明します。その他の文については、“6.4 文”を参照してください。

- ALTER文
- CALL文
- ENTRY文
- EVALUATE文
- EXIT文
- GO TO文
- INITIALIZE文
- INVOKE文
- MERGE文
- OPEN文
- PERFORM文
- RAISE文
- RESUME文
- SET文
- SORT文
- STOP文
- TRY文
- USE文

・ WRITE文(順ファイル)

11.8.3.1 条件文と無条件文

以下に示した.NETプログラミング固有の文は、すべて無条件文となります。なお、条件文と無条件文については、“6.1 手続き部の構成（PROCEDURE DIVISION）”を参照してください。

・ TRY文
・ INVOKE文
・ RAISE文
・ RESUME文
・ USE文
・ EXIT TRY文
・ EXIT METHOD文
・ EXIT PROGRAM文

11.8.3.2 ALTER文

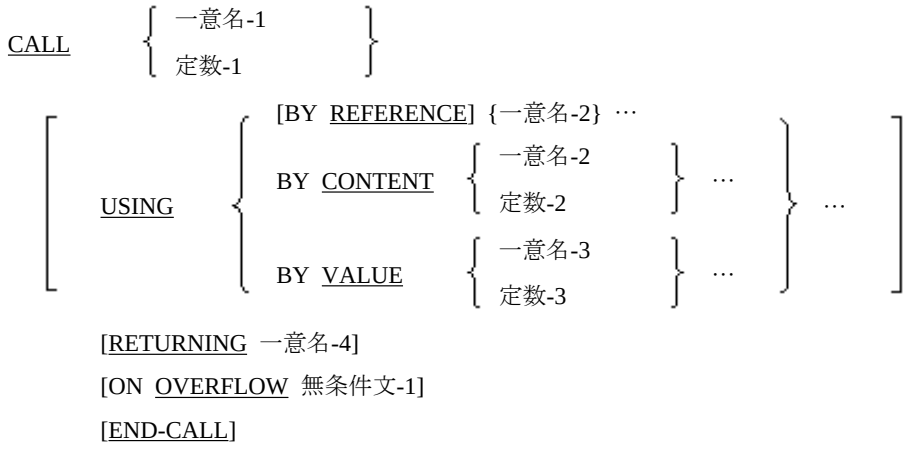
.NETプログラミング機能では、ALTER文を使用することはできません。

11.8.3.3 CALL文

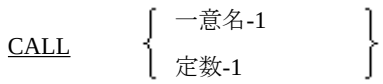
ここでは、.NETプログラミング機能における、CALL文について説明します。

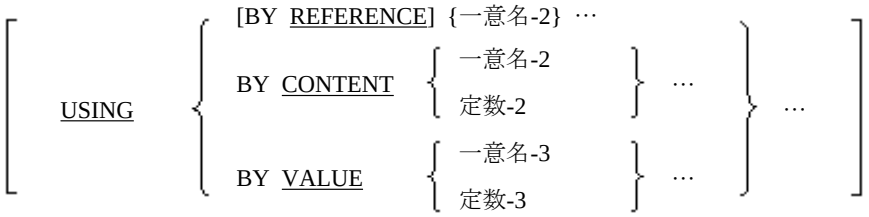
CALL文は、実行単位中のほかのプログラムに制御を移します。

【書き方1】(ON OVERFLOW)



【書き方2】(ON EXCEPTION)





[RETURNING 一意名-4]

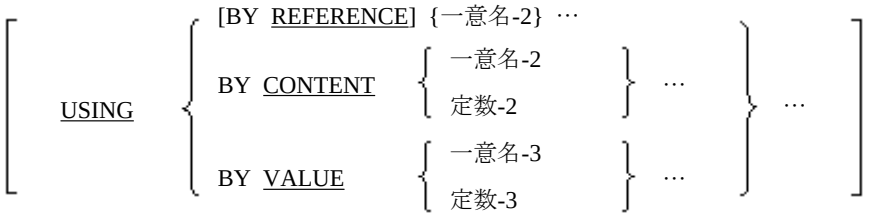
[ON EXCEPTION 無条件文-1]

[NOT ON EXCEPTION 無条件文-2]

[END-CALL]

【書き方3】(プログラム原型)

CALL {プログラム原型名-1}



[RETURNING 一意名-4]

[ON EXCEPTION 無条件文-1]

[NOT ON EXCEPTION 無条件文-2]

[END-CALL]

構文規則

すべての書き方に共通する規則

一意名-4は、ファイル節、作業場所節または連絡節で定義されたデータ項目でなければなりません。

書き方1および書き方2に共通する規則

1. 一意名-2および一意名-3は、基底場所節、ファイル節、作業場所節、定数節または連絡節で定義されたデータ項目でなければなりません。
2. 一意名-3は、用途がCOMPUTATIONAL-5の9桁以下の整数項目、BINARY-SHORT SIGNEDまたはBINARY-LONG SIGNEDでなければなりません。
3. 定数-3は、9桁以下の整数でなければなりません。
4. その他の構文規則については、“6.4.6 CALL文(プログラム間連絡)”を参照してください。

書き方3の規則

1. 一意名-2は、基底場所節、ファイル節、作業場所節または連絡節で定義されたデータ項目でなければなりません。
2. 一意名-3は、基底場所節、ファイル節、作業場所節、定数節または連絡節で定義されたデータ項目でなければなりません。
3. USING指定は、プログラム原型のUSING指定の書き方に従います。
4. 引数にBY CONTENTまたはBY REFERENCEが指定された場合、手続き部の見出しの対応する仮引数にBY REFERENCE指定が書かれていなければなりません。

5. 定数-2は、BY CONTENTで渡される基本項目の適合に規定されているように、仮引数に適合していなければなりません。
6. 定数-2は、数字定数、文字定数、日本語文字定数またはブール定数でなければなりません。

一般規則

すべての書き方に共通する規則

CALL文の実行の始めに、一意名-1、一意名-2、一意名-3、一意名-4が評価されます。

書き方1および書き方2に共通する規則

1. RETURNING指定が書かれた場合、一意名-1または定数-1で識別されるプログラムの結果が一意名-4に格納されます。ただし、呼び出されたプログラムからの制御の復帰が、例外の通知とともに行われた場合には、プログラムの結果は一意名-4には反映されません。
2. その他の一般規則については“6.4.6 CALL文（プログラム間連絡）”を参照してください。

書き方3の規則

1. BY VALUEを指定した場合、引数の値が直接呼び出されるプログラムに渡されます。呼び出されるプログラムでBY VALUE指定によって渡された値を変更しても、呼び出すプログラムのデータ項目の値は変更されません。
2. CALL文のそれぞれの引数に、BY CONTENT指定、BY REFERENCE指定、BY VALUE指定のいずれも指定されなかった場合、その引数の渡しかたは対応する仮引数の指定により決まります。ただし、定数-2か、一意名-3としてEXCEPTION-OBJECT、NULL、SELFまたは定数節で定義したデータ項目が書かれた場合は、BY CONTENTが指定されたとみなします。
3. プログラム原型名-1は、呼び出されるプログラムを決めるために使われます。
4. 制御は、プログラムに対する呼出し規則に矛盾のない方法で、呼び出されたプログラムに移行します。
5. RETURNING指定が書かれた場合、プログラム原型名-1で識別されるプログラムの結果が一意名-4に格納されます。ただし、呼び出されたプログラムからの制御の復帰が、例外の通知とともに行われた場合には、プログラムの結果は一意名-4には反映されません。
6. 呼び出すプログラムがプログラムの結果を返さない場合は、RETURNINGを指定してはなりません。

11.8.3.4 ENTRY文

.NETプログラミング機能では、ENTRY文を使用することはできません。

11.8.3.5 EVALUATE文

.NETプログラミング機能以外の規則については、“6.4.17 EVALUATE文（中核）”を参照してください。

構文規則

THROUGH指定の前後の作用対象は、字類がオブジェクトであってはなりません。

11.8.3.6 EXIT文

ここでは、.NETプログラミング機能における、EXIT文について説明します。

EXIT PROGRAM文は、呼び出されたプログラムの論理的な終わりを示します。EXIT METHOD文は、呼び出されたメソッドの論理的な終わりを示します。EXIT TRY文は、TRYブロックおよびCATCHブロックの論理的な終わりを示します。

【書き方1】（プログラム）

EXIT PROGRAM [RAISING 一意名-1]

【書き方2】（メソッド）

EXIT METHOD [RAISING 一意名-1]

【書き方3】(TRYブロック/CATCHブロック)

EXIT TRY

構文規則

書き方1および書き方2に共通する規則

一意名-1は、このEXIT文を含むソース要素の手続き部の、見出しのRAISING指定に書かれたクラスのオブジェクトを参照するオブジェクト参照一意名でなければなりません。かつ、そのクラスは、System.Exceptionクラスを継承するクラスでなければなりません。

書き方1の規則

EXIT PROGRAM文は、メソッド定義に書いてはなりません。

書き方2の規則

EXIT METHOD文は、メソッドの手続き部にだけ指定できます。

書き方3の規則

EXIT TRY文は、TRY文の中にだけ指定できます。ただし、EXIT TRY文が含まれる最も内側のTRY文のFINALLYブロックには指定できません。

一般規則

書き方1および書き方2に共通する規則

RAISING指定が書かれた場合、一意名-1により参照されているオブジェクトが、例外オブジェクトとして呼出し元に通知されます。

書き方1の規則

呼び出したエンティティの制御下にないプログラム中でEXIT PROGRAM文を実行した場合、EXIT PROGRAM文はCONTINUE文のように扱われます。たとえば、RAISING指定が書かれていても、例外条件は発生しません。

書き方2の規則

EXIT METHOD文の実行は、メソッドの終了を引き起こし、制御は呼び出した文に戻ります。この文を含むメソッド定義にRETURNING指定がある場合、RETURNING指定で参照されるデータ項目の値はメソッド呼出しの結果になります。

書き方3の規則

EXIT TRY文は、それが含まれる最も内側のTRY文と対応付けられます。
EXIT TRY文を実行すると、対応付けられたTRY文が終了します。このとき次の順に動作します。

1. 対応付けられたTRY文にFINALLY指定があれば、そのFINALLYブロックが実行されます。
2. 対応付けられたTRY文のEND-TRYの直後に制御が移されます。

11.8.3.7 GO TO文

ここでは、.NETプログラミング機能における、GO TO文について説明します。

その他の規則については、“6.4.22 GO TO文(中核)”を参照してください。

【書き方1】(特定の部分へ制御を移す)

GO TO [手続き名-1]

【書き方2】(データ項目の値にしたがって制御を移す)

GO TO {手続き名-1} … DEPENDING ON 一意名-1

構文規則

GO TO文の手続き名-1を省略することはできません。

11.8.3.8 INITIALIZE文

.NETプログラミング機能以外の規則については、"6.4.24 INITIALIZE文(中核)"を参照してください。

一般規則

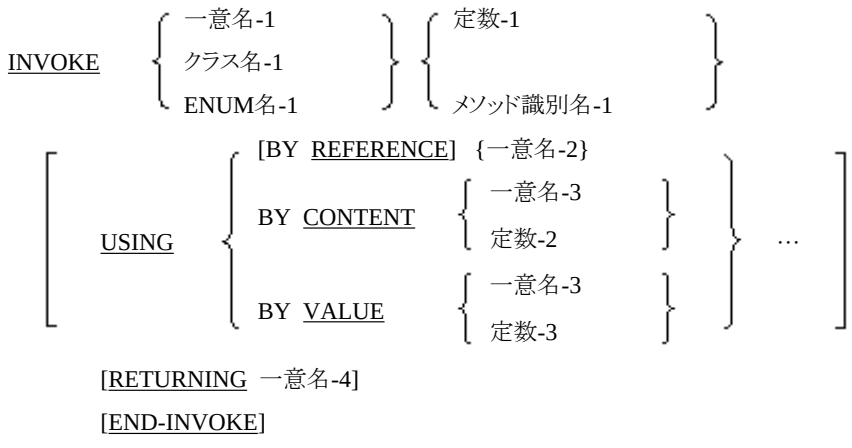
それぞれの暗黙のMOVEの受取り側作用対象を決める処理で、オブジェクト参照データ項目は、受取り側項目から除外されます。

11.8.3.9 INVOKE文

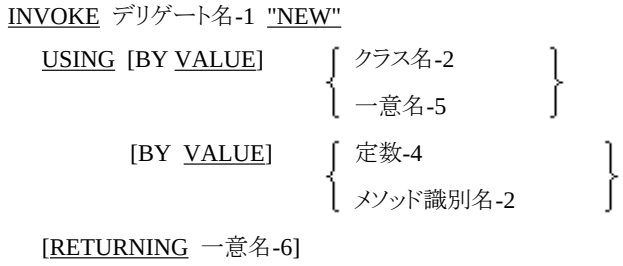
ここでは、.NETプログラミング機能における、INVOKE文について説明します。

INVOKE文は、メソッドを呼び出します。

【書き方1】(通常のメソッド呼出し)



【書き方2】(デリゲートオブジェクトのコンストラクタ呼出し)



構文規則

書き方1および書き方2に共通する規則

メソッド名NEWは、コンストラクタを示す名前としてコンパイラに予約されています。

書き方1の規則

1. クラス名-1、ENUM名-1およびメソッド識別名-1は、それぞれ、リポジトリ段落のクラス指定子、ENUM指定子、メソッド指定子で指定された名前でなければなりません。
2. 一意名-1は、オブジェクト一意名でなければなりません。ただし、NULLであってはなりません。
3. 定数-1は、文字定数または日本語文字定数でなければなりません。ただし、16進文字定数や日本語16進文字定数であってはなりません。
4. 定数-1で“NEW”と指定した場合は、一意名-1としてSUPERを指定するか、クラス名-1を指定しなければなりません。ただし、一意名-1にSUPERを指定した場合、INVOKE文はNEWメソッドの手続き部に記述されなければなりません。
5. 一意名-2は、基底場所節、ファイル節、作業場所節または連絡節で定義されたデータ名でなければなりません。
6. 一意名-3は、SELF、NULL、EXCEPTION-OBJECT、基底場所節、ファイル節、作業場所節、定数節または連絡節で定義されたデータ名でなければなりません。
7. 一意名-4は、ファイル節、作業場所節または連絡節で定義されたデータ名でなければなりません。
8. 呼び出されるメソッドの手続き部の見出しがUSING指定を持つ場合、USINGを指定しなければなりません。また、呼び出されるメソッドの手続き部の見出しがUSING指定を持たない場合、USINGを指定してはなりません。
9. 呼び出されるメソッドの手続き部の見出しがRETURNING指定を持つ場合に、RETURNINGを指定しなくてもかまいません。呼び出されるメソッドの手続き部の見出しがRETURNING指定を持たない場合は、RETURNINGを指定してはなりません。ただし、コンストラクタ(NEW)の呼出しの場合は、この限りでありません。
10. 一意名-2、一意名-3、一意名-4、定数-2および定数-3は、引数の適合で規定したように、適合の規則にしたがっていなければなりません。
11. 一意名-2に明または暗にBY REFERENCEを指定した場合、部分参照してはなりません。
12. 一意名-2に明または暗にBY REFERENCEを指定し、かつ一意名-2として内部ブール項目を指定する場合、内部ブール項目はバイト境界で始まるように定義しなければなりません。
13. 一意名-4は部分参照してはなりません。
14. 一意名-4として内部ブール項目を指定する場合、内部ブール項目はバイト境界で始まるように定義しなければなりません。
15. 引数にBY VALUE指定が書かれた場合、手続き部の見出しの対応する仮引数にBY VALUE指定が書かれていなければなりません。
16. クラス名-1、ENUM-1名または一意名-1、メソッド名およびUSINGに指定した実引数の組合せから、呼び出されるメソッドが一意に決定できなければなりません。呼出し可能なメソッドが複数ある場合、その1つが次の条件を満たすなら、呼び出されるメソッドを一意に決定することができます。
    a. 実引数に対して、より適切な適合をする仮引数が1つ以上存在する。かつ、
    b. 上記以外の引数について、このメソッドより適切な適合を持つメソッドが他に存在しない。

書き方2の規則

1. デリゲート名-1は、このソース要素を含むクラスのリポジトリ段落で宣言されたデリゲート名でなければなりません。
2. クラス名-2は、このソース要素を含むクラスのリポジトリ段落で宣言されたクラス名でなければなりません。
3. 一意名-5は、オブジェクト参照項目のデータ名か、または定義済みオブジェクト一意名SELFでなければなりません。

4. 定数-4は文字定数でなければなりません。ただし、16進文字定数定数であってはなりません。また、その内容はNEWであってはなりません。

5. メソッド識別名-2は、このソース要素を含むクラスのリポジトリ段落で宣言されたメソッド識別名でなければなりません。

6. 一意名-6は、デリゲート名-1のオブジェクト参照項目が代入できるオブジェクト参照項目でなければなりません。

一般規則

書き方1の規則

1. 定数-1は呼び出すメソッドの名前を識別します。呼び出されるメソッドがCOBOLのメソッドの場合、定数-1の内容は、呼び出されるメソッドの外部名とみなされます。

2. メソッド識別名-1は、型パラメタ付きのメソッドの名前と、それに渡される型引数の型の組合せを識別します。

3. 一意名-1は、オブジェクトインスタンスを識別します。このとき、定数-1またはメソッド識別名-1の指定により、次のようにメソッドの識別が行われます。

    a. 定数-1が指定された場合、定数-1で指定される名前を持つオブジェクトのメソッドが、型パラメタの有無にかかわらず識別されます。

    b. メソッド識別名-1が指定された場合、メソッド識別名-1で指定した名前と同じ名前のオブジェクトメソッドの中で、メソッド識別名-1を定義するメソッド指定子のUSING指定に記述した型引数と同じ個数の型パラメタを持つメソッドだけが識別されます。

4. クラス名-1は、指定したクラスそのものを識別します。このとき、定数-1またはメソッド識別名-1の指定により、次のようにメソッドの識別が行われます。

    a. 定数-1により指定されるメソッドの名前がNEWである場合、クラス名-1で指定されるクラスで定義されているオブジェクトインスタンスのコンストラクタが識別されます。

    b. 定数-1により指定されるメソッドの名前がNEW以外である場合、クラス名-1で指定されるクラスまたはその親クラスで定義されている名前が一致するスタティックメソッドが、型パラメタの有無にかかわらず識別されます。

    c. メソッド識別名-1が指定された場合、クラス名-1で指定されたクラスまたはその親クラスで定義された名前と同じスタティックメソッドの中で、メソッド識別名-1を定義するメソッド指定子のUSING指定に記述した型引数の個数と同じ個数の型パラメタを持つメソッドだけが識別されます。

5. ENUM名-1は、指定したENUMそのものを識別します。このとき、定数-1の指定により、ENUM名-1で指定されるENUM名またはその親クラスで定義されているスタティックメソッドが識別されます。

6. 一意名-3、一意名-4、定数-2および定数-3は、このメソッド呼出しの実パラメタです。実パラメタはUSING指定に指定した順序で、手続き部の見出しのUSING指定で指定した仮パラメタに対応付けられ、以下の規則でメソッドが呼び出されます。

    a. 呼び出すメソッドが型パラメタを持たない場合、パラメタの適合で規定する適合の規則に従い、実パラメタと適合する仮パラメタを持つメソッドが呼び出されます。

    b. 呼び出すメソッドが型パラメタを持つ場合、型の推測で規定する規則に従い、実パラメタによって型引数の型の推測が行われます。その後、パラメタの適合で規定する適合の規則に従い、実パラメタと適合する仮パラメタを持つメソッドが呼び出されます。
    ただし、推測された型引数の型が、対応する型パラメタの制約を満たさない型である場合、メソッドを呼び出すことはできません。

7. RETURNING指定が書かれた場合、一意名-1と定数-1で識別されるメソッドの返却項目の内容が一意名-4に格納されます。ただし、呼び出されたメソッドからの制御の復帰が、例外の通知とともに行われた場合には、メソッドの結果は一意名-4には反映されません。

8. INVOKE文が実行されると、指定されたメソッドが実行のために有効になり、制御は呼び出されたメソッドに移されます。メソッドの実行が完了すると、制御はINVOKE文の最後に移ります。

9. INVOKE文が実行されたときに、INVOKE文で指定されたメソッドが実行できないとわかった場合、その実行単位は例外処理を引き起こします。

10. INVOKE文の各々の引数に対して、引数にBY CONTENT、BY REFERENCEおよびBY VALUEが指定されなかった場合、呼び出されるメソッドの手続き部の見出しに対応する仮引数の指定に合わせます。ただし、定数-2が書かれるか、一意名-3としてEXCEPTION-OBJECT、NULL、SELF、クラス名、または定数節で定義されたデータ名が指定された場合は、BY CONTENTとみなされます。
11. INVOKE文のUSING指定で参照された引数の値は、INVOKE文が実行されたときに、メソッドの呼出しおよび引数の適合にしたがって、呼び出されたメソッドに対して有効になります。
12. 呼び出されたメソッドは、INVOKE文を含んでいてもかまいません。呼び出されたメソッドは、呼び出したメソッドを直接または間接的に呼び出すINVOKE文を実行してもかまいません。
13. END-INVOKE指定は、INVOKE文の範囲を定めます。

書き方2の規則

1. この書き方はデリゲートのコンストラクタ呼出しです。
2. 一意名-5は、オブジェクトインスタンスを識別します。オブジェクト参照項目で識別されるオブジェクトインスタンス上で動作するメソッドが、生成されるデリゲートオブジェクトに対応付けられます。このとき、定数-4またはメソッド識別名-2の指定で、次のようにメソッドの識別が行われます。
    a. 定数-4が指定された場合、定数-4で指定される名前を持つオブジェクトのメソッドが、型パラメタの有無にかかわらず識別されます。
    b. メソッド識別名-2が指定された場合、メソッド識別名-2で指定した名前と同じ名前のオブジェクトメソッドの中で、メソッド識別名-1を定義するメソッド指定子のUSING指定に記述した型引数の個数と同じ個数の型パラメタを持つメソッドだけが識別されます。
3. クラス名-2は，指定したクラスそのものを識別します。このクラス名で識別されるクラスまたはその親クラスのスタティックメソッドが、生成されるデリゲートオブジェクトに対応付けられます。このとき、定数-4またはメソッド識別名-2の指定で、以下のようにメソッドの識別が行われます。
    a. 定数-4が指定された場合、クラス名-2で指定されたクラスまたはその親クラスで定義された名前と同じスタティックメソッドが、型パラメタの有無にかかわらず識別されます。
    b. メソッド識別名-2が指定された場合、クラス名-2で指定されたクラスまたはその親クラスで定義された名前と同じスタティックメソッドの中で、メソッド識別名-2を定義するメソッド指定子のUSING指定に記述した型引数の個数と同じ個数の型パラメタを持つメソッドだけが識別されます。
4. 1. の規則にしたがって識別されるメソッドは、デリゲート名-1によって識別されるデリゲートに含まれるInvokeメソッドと引数が適合するものでなければなりません。
5. この文の実行により、一意名-6に返されるデリゲートオブジェクトと対応付けたメソッドは、一意名-6に返されるデリゲートオブジェクトからInvokeメソッドを呼び出すことによって、間接的に呼び出すことができます。

11.8.3.10 MERGE文

.NETプログラミング機能以外の規則については、“6.4.27 MERGE文（整列併合）”を参照してください。

一般規則

OUTPUT PROCEDURE指定の規則

出力手続きの範囲で、MERGE文と同じソース要素に書かれたEXIT METHOD文を実行することはできません。

11.8.3.11 OPEN文

.NETプログラミング機能以外の規則については、“6.4.30 OPEN文（順ファイル・相対ファイル・索引ファイル）”を参照してください。

構文規則

OPEN文のREVERSED指定は、プログラム定義の手続き部にだけ書くことができます。

11.8.3.12 PERFORM文

【書き方1】、【書き方2】、【書き方3】、【書き方4】、【書き方5】

“6.4.33 PERFORM文（中核）”を参照してください。

【書き方6】

データ集合(コレクション)内の全要素に対して、指定された文の組を繰り返し実行する

PERFORM VARYING 一意名-1
{ THRU | THROUGH } 一意名-2
無条件文-1
END-PERFORM

構文規則

書き方1、書き方2、書き方3、書き方4、書き方5の規則

“6.4.33 PERFORM文（中核）”を参照してください。

書き方6の規則

1. THRUとTHROUGHは同義語であるため、どちらを書いてもかまいません。
2. 一意名-1は、データ項目でなければなりません。
3. 一意名-2は、データ集合を示すオブジェクト一意名でなければなりません。ただし、定義済みオブジェクト一意名であってはいけません。
4. 一意名-1が示す型は、一意名-2が示すデータ集合内の要素を代入できる型でなければなりません。
5. 一意名-2のオブジェクト一意名が示す型は、次のどれかでなければなりません。
    a. System.Collections.IEnumerableインタフェースを実装する型
    b. System.Collections.Generic.IEnumerable`1インタフェースを実装する型
    c. 次の条件をすべて満たす型
        - GetEnumeratorという名前の、アクセス可能なインスタンスメソッドを持つ型。
          なお、このGetEnumeratorメソッドはUSING引数を持たず、かつ任意の型Tを返す。
        - 上記GetEnumeratorメソッドが返却する型Tには、MoveNextという名前の、アクセス可能なインスタンスメソッドが含まれる。
          なお、このMoveNextメソッドはUSING引数を持たず、かつBoolean型を返す。
        - 上記GetEnumeratorメソッドが返却する型Tには、Currentという名前のアクセス可能なインスタンスプロパティが含まれる。
          なお、このCurrentプロパティはgetアクセサを持つ。

一般規則

書き方1、書き方2、書き方3、書き方4、書き方5の規則

“6.4.33 PERFORM文（中核）”を参照してください。

書き方6の規則

1. END-PERFORMは、うちPERFORM文の文の範囲を区切ります。

2. PERFORM文は、一意名-2が示すデータ集合内の全要素を取り出し終わるまで、指定された文の組を繰り返し実行します。
3. PERFORM文を実行すると、指定された文の組の最初の文が実行される前に、一意名-2が示すデータ集合内の最初の要素が一意名-1に格納されます。その後、指定された文の組が実行されます。
   指定された文の組の最後の文の実行が終わると、一意名-2が示すデータ集合内に次の要素があるかどうか検査されます。
   次の要素があれば、それを一意名-1に格納した後で、再び指定された文の組を最初から実行します。次の要素がなければ、PERFORM文の終わりに制御が移ります。
4. 一意名-2が示すデータ集合は、空であってはいけません。
5. 繰り返し処理中に、一意名-1の値を変更してはいけません。繰り返し処理中に一意名-1の値を変更した場合の動作は保証されません。
6. TEST指定付きのEXIT文が、無条件文-1として指定された場合は、一意名-2が示すデータ集合内に、取り出すべき次の要素があるかどうかを検査します。

11.8.3.13 RAISE文

ここでは、.NETプログラミング機能における、RAISE文について説明します。

【書き方】

RAISE 一意名-1

構文規則

1. 一意名-1は、オブジェクト参照一意名でなければなりません。かつ、そのクラスはSystem.Exceptionクラスを継承するクラスでなければなりません。
2. 一意名-1を省略したRAISE文は、TRY文の中にだけ記述できます。ただし、RAISE文が含まれる最も内側のTRY文のCATCHブロックに記述しなければなりません。

一般規則

1. 一意名-1が省略されたRAISE文は、それが含まれる最も内側のTRY文に対応付けられます。
2. 一意名-1が指定されたRAISE文を実行すると、一意名-1で指定された例外オブジェクトが発生します。一意名-1が指定されていないRAISE文を実行すると、次に示す順に手続きが行われます。
   1. 対応付けられたTRY文にFINALLY指定があれば、そのFINALLYブロックが実行されます。
   2. RAISE文が含まれる、対応付けられたTRY文のCATCHブロックに制御が移ったときの例外オブジェクトが、対応付けられたTRY文の外側に対して通知されます。
3. 以下の条件が成り立つ場合、TRY文中に記述された場合を除くと、RAISE文の効果は、CONTINUE文と同じです。
   a. 一意名-1が指定されたが、適用可能な宣言手続きがない。
4. TRY文中に記述され、かつ一意名-1が指定されたRAISE文を実行すると、一意名-1がNULLオブジェクトを保持していた場合、System.NullReference例外が発生します。

11.8.3.14 RESUME文

ここでは、.NETプログラミング機能における、RESUME文について説明します。

RESUME文は、RESUME文が書かれたUSE手続きの例外処理を完了させた後、宣言節外の別の部分に制御を移します。

【書き方】

RESUME AT 手続き名-1

構文規則

1. RESUME文は、例外オブジェクトのUSE手続きの中にだけ書くことができます。
2. 手続き名-1は、メソッドまたはプログラムの宣言節以外の手続き名でなければなりません。

一般規則

RESUME文が実行されると、「GO TO 手続き名-1」が実行された場合と同じように制御は手続き名-1に移ります。また、RESUME文が書かれたUSE手続きが、例外処理として実行されている場合、その例外処理を完了します。

11.8.3.15 SET文

ここでは、.NETプログラミング機能における、SET文について説明します。

SET文は、オブジェクト参照の代入手段を提供します。

【書き方1】(オブジェクト参照からオブジェクト参照へ)

SET {一意名-1} … TO 一意名-2

【書き方2】(オブジェクト参照からオブジェクト参照以外のデータ型へ)

SET {一意名-3} … TO 一意名-4

【書き方3】(オブジェクト参照以外のデータ型からオブジェクト参照へ)

SET {一意名-5} … TO { 一意名-6 / 定数-1 }

構文規則

書き方1の規則

一意名-1および一意名-2は、オブジェクト参照一意名でなければなりません。

書き方2および書き方3に共通する規則

1. 一意名-3は、数字項目、英数字項目、日本語項目、浮動小数点項目またはブール項目でなければなりません。
2. 一意名-6は、数字項目、英数字項目、日本語項目、浮動小数点項目、ブール項目、数字編集項目、英数字編集項目または日本語編集項目でなければなりません。
3. 一意名-4および一意名-5は、以下のオブジェクトのオブジェクト参照一意名でなければなりません。
   - System.Byte
   - System.SByte

- System.Int16
- System.UInt16
- System.Int32
- System.UInt32
- System.Int64
- System.UInt64
- System.Single
- System.Double
- System.Decimal
- System.Char
- System.Boolean
- System.String
- System.Byte[]
- System.Char[]
- System.Object
- ENUM

4. SET文で許される一意名-3と一意名-4の組合せ、および一意名-5と一意名-6、定数-1の組合せは、“11.8.2.6.1 代入時の適合”を参照してください。

一般規則

書き方1の規則

SET文は、一意名-2で識別されるオブジェクトのオブジェクト参照を取出し、それぞれの一意名-1に割り当てられた記憶領域に指定された順序で格納します。

11.8.3.16 SORT文

.NETプログラミング機能以外の規則については、“6.4.41 SORT文（整列併合）”を参照してください。

一般規則

INPUT PROCEDURE指定の規則

入力手続きの範囲で、SORT文と同じソース要素に書かれたEXIT METHOD文を実行することはできません。

OUTPUT PROCEDURE指定の規則

出力手続きの範囲で、SORT文と同じソース要素に書かれたEXIT METHOD文を実行することはできません。

11.8.3.17 STOP文

ここでは、.NETプログラミング機能における、STOP文について説明します。

その他の規則については、“6.4.44 STOP文（中核）”を参照してください。

【書き方】

```
STOP { RUN    }
     { 定数-1 }
```

構文規則

STOP文の定数-1を指定してはなりません。

11.8.3.18 TRY文

ここでは、.NETプログラミング機能における、TRY文について説明します。

【書き方】

TRY 無条件文-1

[{CATCH [データ名-1] 無条件文-2} …]

[FINALLY 無条件文-3]

END-TRY

構文規則

1. データ名-1は、System.Exceptionクラス、またはSystem.Exceptionクラスを継承するクラスを参照する、オブジェクト参照データ項目でなければなりません。
2. CATCH指定およびFINALLY指定は、両方またはどちらか一方を記述しなければなりません。
3. 無条件文-1、無条件文-2および無条件文-3に、次の文を記述してはいけません。
    a. そとPERFORM文
    b. INPUT PROCEDURE指定またはOUTPUT PROCEDURE指定を持つ、SORT文およびMERGE文
    c. メソッド定義のファイル節に定義された、LINAGE句指定のファイルを操作する入出力文
4. 無条件文-3に、次の文を記述してはいけません。
    a. GO TO文
    b. EXIT PROGRAM文またはEXIT METHOD文
    c. EXIT TRY文
    d. STOP RUN文
    e. EXIT PERFORM文

    ただし、EXIT PERFORM文は、対応付けられる、うちPERFORM文も無条件文-3に含まれている場合は記述できます。
    また、EXIT TRY文は、無条件文-3としてTRY文が記述された場合、その内側の無条件文-1および無条件文-2としては記述できます。他の文には、無条件文-3の範囲内ならば、無条件に記述できません。
5. TRY文が記述された手続きに宣言節が存在し、かつそこにINPUT、OUTPUT、I-OまたはEXTEND指定の入出力誤り処理手続きのUSE文がある場合、無条件文-1、無条件文-2および無条件文-3に入出力文を記述してはいけません。
6. TRY文が記述された手続きに宣言節が存在し、かつそこにファイル名指定の入出力誤り処理手続きのUSE文がある場合、無条件文-1、無条件文-2および無条件文-3に、そのUSE文に指定されたファイルを対象とした入出力文を記述してはいけません。
7. 手続きに宣言節が存在し、かつそこに例外オブジェクトのUSE文がある場合、TRY文を記述してはいけません。

一般規則

1. TRY文は、手続きを特定の範囲でブロック化し、その範囲に限定した例外処理と終了処理を実現します。無条件文-1の範囲をTRYブロック、無条件文-2の範囲をCATCHブロック、無条件文-3の範囲をFINALLYブロックと呼びます。
2. TRYブロックの実行が正常に終了した場合、以下に示す手続きが上から順に動作します。
    1. FINALLY指定が存在する場合、無条件文-3が実行されます。
    2. END-TRYの直後に制御が移されます。
3. TRYブロックの実行途中で例外が発生した場合、そこでTRYブロックの手続きは終了し、以降は次に示す手続きが上から順に動作します。

    ー 発生した例外に、該当するCATCH指定が存在する場合

    1. 該当するCATCH指定の無条件文-2が実行されます。
    2. FINALLY指定が存在するならば、無条件文-3が実行されます。
    3. END-TRYの直後に制御が移されます。

    ー 発生した例外に、該当するCATCH指定が存在しない場合

    1. FINALLY指定が存在するならば、無条件文-3が実行されます。
    2. TRY文の外側に，発生した例外が通知されます。

    発生した例外に該当するCATCH指定とは、以下のものです。

    a. データ名-1が参照するクラスが、発生した例外のクラスまたは発生した例外の親クラスである

    b. データ名-1が省略されている

    該当するCATCH指定が複数ある場合、最も上位に記述されたCATCH指定が選択されます。
4. 例外の発生で、制御が特定のCATCHブロックに移行したとき、対応するCATCH指定にデータ名-1が指定されていると、発生した例外のオブジェクト参照がデータ名-1に設定されます。
5. データ名-1が省略されているCATCH指定は、System.Exceptionクラスを継承している例外のすべてが該当します。このCATCHブロックに制御が移行した場合、例外オブジェクトを参照できません。
6. CATCHブロックの実行途中で例外が発生した場合、そこでCATCHブロックの手続きは終了し、以降は次に示す手続きが上から順に動作します。
    1. FINALLYブロックが存在する場合、無条件文-3が実行されます。
    2. TRY文の外側に、発生した例外が通知されます。
7. FINALLYブロックの実行途中で例外が発生した場合、そこでFINALLYブロックの手続きは終了し、TRY文の外側に発生した例外が通知されます。
8. 無条件文-3として埋め込みSQL文が記述された場合、その埋め込みSQL文に対しては埋込み例外宣言が有効になりません。

11.8.3.19 USE文

ここでは、.NETプログラミング機能における、USE文について説明します。

その他の規則については、“6.4.51 USE文（順ファイル・相対ファイル・索引ファイル・表示ファイル・報告書作成）”を参照してください。

【書き方】(例外オブジェクト)

USE AFTER EXCEPTION クラス名-1

構文規則

クラス名-1は、リポジトリ段落で指定されたクラスの名前でなければなりません。かつ、そのクラスは、System.Exceptionクラスを継承するクラスでなければなりません。

一般規則

1. プログラムに指定された順番にUSE文を解析し、実行する宣言が選択されます。
   クラス名-1が指定され、発生した例外がクラス名-1を継承している場合、関連する宣言が実行されます。
2. 関連した宣言の実行中、定義済みオブジェクト一意名EXCEPTION-OBJECTは例外オブジェクトを参照します。

11.8.3.20 WRITE文（順ファイル）

.NETプログラミング機能以外の規則については、“6.4.54 WRITE文（順ファイル）”を参照してください。

構文規則

クラス定義に含まれないメソッド定義の手続き部で、スタティック定義またはオブジェクト定義で宣言したファイルに対するWRITE文にADVANCING指定を書く場合、以下のどれかを満たしていなければなりません。

- そのファイルに対し、ASSIGN句でPRINTERまたはPRINTER-n(nは1～9までの整数)が指定されている。
- そのファイルを定義したソース単位に含まれるソース要素に、そのファイルに対するADVANCING指定付きのWRITE文が指定されている。
- そのファイルがFORMAT句付き印刷ファイルである。

11.9 データベース（SQL）

ここでは、クラス定義でデータベース(SQL)機能を使用する際の規則について説明します。

“第8章 データベース（SQL）”の対応する部分を置き換えてお読みください。

11.9.1 埋込みSQLの正書法

埋込みSQLは、スタティック定義、オブジェクト定義またはクラスに含まれるメソッド定義に書くことができます。

11.9.2 データ部

11.9.2.1 埋込みSQL宣言節

一般規則

埋込みSQL宣言節は、スタティック定義、オブジェクト定義のデータ部の作業場所節、クラスに含まれるメソッド定義のデータ部の作業場所節および連絡節に記述することができます。

11.9.3 埋込み例外宣言

一般規則

1. 埋込み例外宣言は、原始プログラム上の次の文以降、同一の条件を指定した埋込み例外宣言が現れるか、またはメソッドの終わりに達するまで、その間にあるすべてのSQL文に対して有効です。
2. 埋込み例外宣言の有効範囲は、メソッド単位です。

11.9.4 カーソル系データ操作文

11.9.4.1 カーソル宣言

構文規則

カーソル名は、スタティック定義またはオブジェクト定義内で一意でなければなりません。

一般規則

1. スタティックメソッド内で宣言したカーソルは、同じクラスのすべてのスタティックメソッドで有効です。実行単位の終了時に、スタティックメソッド内で宣言したカーソルが閉じられていない場合、これらのカーソルはシステムによってすべて閉じられます。
2. オブジェクトメソッド内で宣言したカーソルは、同じクラスのすべてのオブジェクトメソッドで有効です。同一のクラスに2つ以上のオブジェクトが存在する場合、それぞれのオブジェクトが自分自身のカーソルを持ちます。オブジェクトが破棄されたときに、オブジェクト内で宣言されたカーソルが閉じられていない場合、これらのカーソルはシステムによってすべて閉じられます。

11.9.5 動的SQL

11.9.5.1 EXECUTE文

一般規則

1. スタティックメソッドで使用するSQL文識別子に対応する被準備文は、同一スタティック定義内の同名のSQL文識別子に対するPREPARE文で、実行の準備を完了していなければなりません。
2. スタティックメソッド内で宣言され、準備された状態にあるSQL文識別子は、そのクラスを使用する実行単位すべてで有効です。実行単位の終了時に、これらスタティックメソッド内で宣言されたSQL文識別子が準備された状態になっている場合、これらのSQL文識別子はシステムによってすべて準備されていない状態になります。
3. オブジェクトメソッドで使用するSQL文識別子に対応する被準備文は、同一オブジェクト定義内の同名のSQL文識別子に対するPREPARE文で、実行の準備を完了していなければなりません。
4. オブジェクトメソッド内で宣言され、準備された状態にあるSQL文識別子は、そのメソッドを含むオブジェクト内すべてで有効です。
   同一のクラスに2つ以上のオブジェクトがある場合、それぞれのオブジェクトが自分自身のSQL文識別子を持ちます。オブジェクトが破棄されたときに、オブジェクト内で宣言されたSQL文識別子が準備された状態になっている場合、これらのSQL文識別子はシステムによってすべて準備されていない状態になります。

11.9.5.2 動的カーソル宣言

構文規則

1. カーソル名は、スタティック定義またはオブジェクト定義内で一意でなければなりません。
2. スタティックメソッド内の動的カーソル宣言で指定するSQL文識別子は、同じクラスのスタティックメソッド内のPREPARE文で指定しなければなりません。
3. オブジェクトメソッド内の動的カーソル宣言で指定するSQL文識別子は、同じクラスのオブジェクトメソッド内のPREPARE文で指定しなければなりません。

一般規則

1. スタティックメソッド内で動的に宣言したカーソルとそれに使用した文識別子は、そのクラスのスタティックメソッドを使用する実行単位すべてで有効です。実行単位の終了時、スタティックメソッド内で宣言し、開かれたカーソルが閉じられていない場合、これらのカーソルはシステムによってすべて閉じられます。
   また、スタティックメソッド内で宣言されたSQL文識別子が準備された状態になっている場合、これらのSQL文識別子はシステムによってすべて準備されていない状態になります。

2. オブジェクトメソッド内で動的に宣言したカーソルとそれに使用した文識別子は、そのメソッドを含むオブジェクト内すべてで有効です。
   同一のクラスに2つ以上のオブジェクトが存在する場合、それぞれのオブジェクトが自分自身のカーソルおよびSQL文識別子を持ちます。オブジェクトが破棄されたときに、オブジェクト内で宣言されたカーソルが閉じられていない場合、これらのカーソルはシステムによってすべて閉じられます。
   また、オブジェクト内で宣言されたSQL文識別子が準備された状態になっている場合、これらのSQL文識別子は、システムによってすべて準備されていない状態になります。

11.10 組込み関数

ここでは、.NETプログラミング機能における組込み関数について説明します。

その他の組込み関数の規則については、“6.5 関数の全般規則”および“6.6 関数”を参照してください。

11.10.1 組込み関数の全般規則

ここでは、.NETプログラミング機能における、組込み関数の全般規則について説明します。

組込み関数機能は、プログラムの実行時に引用されると自動的に値が導かれるデータ項目を参照する機能を提供します。

11.10.1.1 概念

本項では、.NETプログラミング機能における、組込み関数の概念について説明します。

関数名

組込み関数機能において、関数は、一時的なデータ項目です。関数が文の実行中に引用されたとき、関数の値は、関数の定義で示される機構を実行することによって決定されます。関数名は、関数の値を決定するこの機構の名前です。関数名は、COBOLプログラムの中で使用できるCOBOLの語の1つです。

関数値

関数が返す値は、データ値とみなされます。実行時に関数を引用すると、その関数にデータ値が割り当てられます。関数の値を決定するために、関数の評価機構は、プログラムからのデータ値を必要とすることがあります。関数を引用するときに、引数によってこれらのデータ値を与えます。これらの引数には、関数によって範囲などの制限を設けることがあります。関数を引用したときに、その引数の値が制限にしたがっていないならば、関数値は規定されません。

関数一意名

関数一意名は、COBOLプログラムの手続きの中で関数を引用するのに用います。

11.10.1.2 関数の定義

ここでは、.NETプログラミング機能における、組込み関数の定義について説明します。

関数の定義では、次が規定されます。

1. 関数値が規定されます。
2. 英数字関数では、関数値の長さが規定されます。
3. 数字関数や整数関数では、関数値の符号および関数値が整数であるかどうかが規定されます。
4. ENUM関数では、関数値のオブジェクトの型が規定されます。

日付変換関数

日付変換関数では、西暦を用います。1601年1月1日(月曜日)を通日の第1日とします。

11.10.1.3 引数の型

ここでは、.NETプログラミング機能における、組込み関数の引数の型について説明します。

引数は、関数を評価するときに用いる値を指定します。引数は、関数一意名の中で指定します。これらの引数には、一意名、算術式または定数を指定できます。引数を指定できる個数は、関数の定義にしたがって、0個または1個以上です。いくつかの関数では、指定できる引数の個数は可変です。関数一意名の中で指定する引数の順序位置は、関数値を導出するときの、その引数の値の解釈を定めます。

引数は、ある字類であること、またはある字類の一部であることを要求されることがあります。

引数の型と指定できる値は、次のとおりです。

1. 引数の型が数字の場合、引数として算術式を指定しなければなりません。関数値を定めるのにこの算術式の値(符号を含む)を用います。
2. 引数の型が英字の場合、字類が英字の基本データ項目または英字だけを含む文字定数を指定しなければなりません。関数値を定めるのにこの引数の大きさを用いることがあります。
3. 引数の型が英数字の場合、字類が英字、英数字の基本データ項目または文字定数を指定しなければなりません。関数値を定めるのに、この引数の大きさを用いることがあります。
4. 引数の型が日本語の場合、字類が日本語の基本データ項目または日本語定数を指定しなければなりません。関数値を定めるのに、この引数の大きさを用いることがあります。
5. 引数の型が整数の場合、その値が常に整数値となる算術式を指定しなければなりません。関数値を定めるのに、この算術式の値（符号を含む）を用います。
6. 引数の型がオブジェクト参照の場合、オブジェクト参照一意名またはENUMフィールド参照を指定しなければなりません。関数値を定めるのに、この引数の型を用います。

関数の規則は、意味のある関数値を導くために、引数の値にある制限を設けることがあります。関数を引用したときに、その引数の値が許される範囲を越えていたならば、その関数値は規定されません。

関数の定義が、ある引数を不定の回数繰り返すことを許す場合、そこに表を指定することができます。表を指定するには、その表を一意に識別するデータ名と修飾語および添字付けを書き、添字のうちの1つ以上は、語ALLを書きます。

添字に語ALLを書くと、その添字に関連する表要素をすべて指定したかのように扱われます。表要素の暗黙的な指定は、表要素をその表の中の出現順にしたがって左から右へ並べたものと等価とします。語ALLの添字がOCCURS DEPENDING ON句に関連している場合の添字の上限は、OCCURS DEPENDING ON句の目的語の値に従います。語ALLのどの添字を評価したときにも、1個以上の引数を指定することにならなければなりません。それ以外の場合、関数値は、規定されません。

11.10.1.4 関数の型

ここでは、.NETプログラミング機能における、組込み関数の型について説明します。

関数のデータ項目は、基本データ項目とし、英数字、数字、整数またはオブジェクトの値を返します。関数のデータ項目は、受取り側には指定できません。

関数のデータ項目の型とその項類は、次のとおりです。

1. 英数字関数の字類および項類は、英数字とします。このデータ項目の文字位置の個数は、その関数の定義で規定します。英数字関数の用途は、暗黙的にDISPLAYとします。
2. 日本語関数の字類および項類は、日本語とします。このデータ項目の文字位置の個数は、その関数の定義で規定します。日本語関数の用途は、暗黙的にDISPLAYとします。
3. 数字関数の字類および項類は、数字とします。数字関数は、常に符号付きです。この関数値の属性は、その関数の定義で規定されます。数字関数は、算術式の中にだけ書くことができます。
   整数の作用対象が要求されている箇所で、数字関数を引用してはなりません。たとえば、そのときの関数引用が整数値をとる場合であっても、引用してはなりません。
4. 整数関数の字類および項類は、数字とします。整数関数は、常に符号付きです。この関数値の属性は、その関数の定義で規定されます。
   - 整数関数は、算術式の中にだけ書くことができます。
   - 整数関数は、整数の作用対象が要求され、かつ符号付きの作用対象が許される箇所で引用することができます。

5. オブジェクト関数の場合、字類はオブジェクト、項類はオブジェクト参照とします。この関数値の属性は、その関数の定義で規定されます。
   オブジェクト関数は、以下の箇所に書くことができます。

   ー 環境部のCUSTOM-ATTRIBUTE句のUSING指定またはPROPERTY指定

   ー SET文の送出し側

   ー 等価比較または不等価比較の左右の作用対象

   ー オブジェクト関数の引数

11.10.2 組込み関数

ここでは、.NETプログラミング機能において組み込まれた、それぞれの関数について説明します。

11.10.2.1 ACP-OF関数

ACP-OF関数は、引数1の値をACP(Ansi Code Page)の文字列に変換したものを返します。

【書き方】

FUNCTION ACP-OF (引数-1)

引数

引数-1は、英数字または日本語でなければなりません。

関数値

1. 関数値は、引数-1の含まれる文字列の値をACP(Ansi Code Page)で表現される文字列に変換した結果を返します。なお、関数値のエンコーディング形式は引数1の形式に従います。
2. 引数-1に含まれる文字列の値がACPで表現できない場合は、実行時例外が発生します。
3. 関数値の長さは、変換後の文字列の大きさに従います。

関数値の型

関数値の型は引数-1の型に従います。

引数の型	関数の型
英数字	英数字
日本語	日本語

11.10.2.2 DISPLAY-OF関数

DISPLAY-OF関数は、引数-1に含まれる日本語文字を対応する英数字文字に置き換えます。

【書き方】

FUNCTION DISPLAY-OF (引数-1 [引数-2])

引数

1. 引数-1の字類は、日本語でなければなりません。
2. 引数-2の字類は、英字または英数字で、長さは1文字でなければなりません。
3. 引数-2は、対応する英数字がなかった場合の、英数字文字の代用文字を指定します。

関数値

1. 関数値は、引数-1に含まれる日本語文字を対応する英数字文字に置き換えた文字列です。
2. 引数-2を指定した場合、対応する英数字文字を持たない引数-1中の日本語文字を、代用文字に置き換えた文字列が返却されます。
3. 引数-2を省略し、かつ、引数-1中の日本語文字に対応する英数字文字がない場合は、実行時例外が発生します。
4. 関数値の大きさは、変換後の文字列を表示用の用途で保持するために必要な大きさです。

関数値の型

関数値の型は英数字です。

DISPLAY-OF関数固有の規則

この関数は、実行時データのコード系がシフトJISの場合、使用できません。

11.10.2.3 ENUM-AND関数

ENUM-AND関数は、引数-1の並びのブール積を返します。

【書き方】

FUNCTION ENUM-AND ({引数-1} …)

引数

1. 引数-1は、ENUMのフィールド参照またはENUMのオブジェクト参照でなければなりません。
2. 引数-1は、すべて同じENUM型でなければなりません。

関数値

関数値は、引数-1と同じENUM型のオブジェクト参照です。

11.10.2.4 ENUM-NOT関数

ENUM-NOT関数は、引数-1の値のブール否定を返します。

【書き方】

FUNCTION ENUM-NOT (引数-1)

引数

引数-1は、ENUMのフィールド参照またはENUMのオブジェクト参照でなければなりません。

関数値

関数値は、引数-1と同じENUM型のオブジェクト参照です。

11.10.2.5 ENUM-OR関数

ENUM-OR関数は、引数-1の並びのブール和を返します。

【書き方】

FUNCTION ENUM-OF ({引数-1} …)

引数

1. 引数-1は、ENUMのフィールド参照またはENUMのオブジェクト参照でなければなりません。
2. 引数-1は、すべて同じENUM型でなければなりません。

関数値

関数値は、引数-1と同じENUM型のオブジェクト参照です。

11.10.2.6 NATIONAL-OF関数

NATIONAL-OF関数は、引数-1に含まれる英数字文字を対応する日本語文字に置き換えます。

【書き方】

FUNCTION NATIONAL-OF (引数-1 [引数-2])

引数

1. 引数-1の字類は、英字または英数字でなければなりません。
2. 引数-2の字類は、日本語で、長さは1文字でなければなりません。
3. 引数-2は、対応する日本語文字がなかった場合の、日本語文字の代用文字を指定します。

関数値

1. 関数値は、引数-1に含まれる英数字文字を対応する日本語文字に置き換えた文字列です。
2. 引数-2を指定した場合、対応する日本語文字を持たない引数-1中の英数字文字を、代用文字に置き換えた文字列が返却されます。
3. 引数-2を省略し、かつ、引数-1中の英数字文字に対応する日本語文字がない場合は、実行時例外が発生します。
4. 関数値の大きさは、変換後の日本語文字列を保持するために必要な大きさです。

関数値の型

関数値の型は日本語です。

NATIONAL-OF関数固有の規則

この関数は、実行時データのコード系がシフトJISの場合、使用できません。

11.10.2.7 UNICODE-OF関数

UNICODE-OF関数は、引数-1の値をUNICODEの文字列に変換したものを返します。

【書き方】

FUNCTION UNICODE-OF (引数-1)

引数

引数-1は、英数字または日本語でなければなりません。

関数値

1. 関数値は、引数-1の含まれる文字列の値をUNICODEで表現される文字列に変換した結果を返します。なお、関数値のエンコーディング形式は引数-1の形式に従います。
2. 引数-1に含まれる文字列の値がUNICODEで表現できない場合は、実行時例外が発生します。
3. 関数値の長さは、変換後の文字列の大きさに従います。

関数値の型

関数値の型は引数-1の型に従います。

引数の型	関数の型
英数字	英数字
日本語	日本語

付録A　予約語一覧

ここでは、COBOLの予約語を示します。

下表の記号の意味は、以下のとおりです。

A欄 ： NetCOBOL

B欄 ： CODASYL 1988(参考)

C欄 ： 85 ANSI(92 JISも同じ)

D欄 ： 74 ANSI

備考欄の＊印 ： オブジェクト指向プログラミング機能で追加された予約語であることを示します。

A欄～D欄の○印 ： 各仕様の予約語であることを示します。

語	A		B	C	D	備考
	[Linux]	その他				
ACCEPT	○	○	○	○	○	
ACCESS	○	○	○	○	○	
ACTUAL	○	○				
ADD	○	○	○	○	○	
ADDRESS	○	○				
ADVANCING	○	○	○	○	○	
AFTER	○	○	○	○	○	
ALL	○	○	○	○	○	
ALPHABET	○	○	○	○		
ALPHABETIC	○	○	○	○	○	
ALPHABETIC-LOWER	○	○	○	○		
ALPHABETIC-UPPER	○	○	○	○		
ALPHANUMERIC	○	○	○	○		
ALPHANUMERIC-EDITED	○	○	○	○		
ALSO	○	○	○	○	○	
ALTER	○	○		○	○	
ALTERNATE	○	○	○	○	○	
AND	○	○	○	○	○	
ANY	○	○	○	○		
APPLY	○	○				
ARE	○	○	○	○	○	
AREA	○	○	○	○	○	
AREAS	○	○	○	○	○	
ARITHMETIC	○	○	○			
AS	○	○				
ASCENDING	○	○	○	○	○	
ASSIGN	○	○	○	○	○	
AT	○	○	○	○	○	

語	A		B	C	D	備考
	[Linux]	その他				
AUTHOR	○	○		○	○	
AUTO	○	○				
AUTOMATIC	○	○				
B-AND	○	○	○			
B-EXOR	○	○	○			
B-LESS	○	○	○			
B-NOT	○	○	○			
B-OR	○	○	○			
BACKGROUND-COLOR	○	○				
BASED	○	○				
BASED-STORAGE	○	○				
BEFORE	○	○	○	○	○	
BEGINNING	○	○				
BELL	○	○				
BINARY	○	○	○	○		
BINARY-CHAR	○	○				
BINARY-DOUBLE	○	○				
BINARY-LONG	○	○				
BINARY-SHORT	○	○				
BIT	○	○	○			
BITS	○	○	○			
BLANK	○	○	○	○	○	
BLINK	○	○				
BLOCK	○	○	○	○	○	
BMP		○				
BOOLEAN	○	○	○			
BOTTOM	○	○	○	○	○	
BY	○	○	○	○	○	
CALL	○	○	○	○	○	
CANCEL	○	○	○	○	○	
CATCH		○				
CBL	○	○				
CD	○	○	○	○	○	
CF	○	○	○	○	○	
CH	○	○	○	○	○	
CHANGED	○	○				
CHARACTER	○	○	○	○	○	
CHARACTERS	○	○	○	○	○	
CLASS	○	○	○	○		

語	A		B	C	D	備考
	[Linux]	その他				
CLASS-ID	○	○				＊
CLOCK-UNITS	○	○		○	○	
CLOSE	○	○	○	○	○	
COBOL	○	○		○	○	
CODE	○	○	○	○	○	
CODE-SET	○	○	○	○	○	
COLLATING	○	○	○	○	○	
COLUMN	○	○	○	○	○	
COM-REG	○	○				
COMMA	○	○	○	○	○	
COMMAND	○	○				
COMMIT	○	○	○			
COMMON	○	○	○	○		
COMMUNICATION	○	○	○	○	○	
COMP	○	○	○	○	○	
COMP-X	○	○				
COMP-1	○	○				
COMP-2	○	○				
COMP-3	○	○				
COMP-4	○	○				
COMP-5	○	○				
COMPLEX	○	○				
COMPUTATIONAL	○	○	○	○	○	
COMPUTATIONAL-X	○	○				
COMPUTATIONAL-1	○	○				
COMPUTATIONAL-2	○	○				
COMPUTATIONAL-3	○	○				
COMPUTATIONAL-4	○	○				
COMPUTATIONAL-5	○	○				
COMPUTE	○	○	○	○	○	
CONFIGURATION	○	○	○	○	○	
CONNECT	○	○	○			
CONSTANT	○	○				
CONTAINED	○	○	○			
CONTAINS	○	○	○	○	○	
CONTENT	○	○	○	○		
CONTINUE	○	○	○	○		
CONTROL	○	○	○	○	○	
CONTROL-CHARACTER	○	○				

語	A		B	C	D	備考
	[Linux]	その他				
CONTROLS	○	○	○	○	○	
CONVERTING	○	○	○	○		
COPY	○	○	○	○	○	
CORE-INDEX	○	○				
CORR	○	○		○	○	
CORRESPONDING	○	○		○	○	
COUNT	○	○	○	○	○	
CRP	○	○				
CRT	○	○				
CRT-UNDER	○	○				
CURRENCY	○	○	○	○	○	
CURRENT	○	○	○			
CURSOR	○	○				
CUSTOM-ATTRIBUTE	○	○				
DATA	○	○	○	○	○	
DATE	○	○	○	○	○	
DATE-COMPILED	○	○		○	○	
DATE-WRITTEN	○	○		○	○	
DAY	○	○	○	○	○	
DAY-OF-WEEK	○	○	○	○		
DB	○	○	○			
DB-ACCESS-CONTROL-KEY	○	○	○			
DB-DATA-NAME	○	○	○			
DB-EXCEPTION	○	○	○			
DB-RECORD-NAME	○	○	○			
DB-SET-NAME	○	○	○			
DB-STATUS	○	○	○			
DBCS	○	○				
DE	○	○	○	○	○	
DEAD-LOCK	○	○				
DEBUG-CONTENTS	○	○		○	○	
DEBUG-ITEM	○	○		○	○	
DEBUG-LINE	○	○		○	○	
DEBUG-NAME	○	○		○	○	
DEBUG-SUB-1	○	○		○	○	
DEBUG-SUB-2	○	○		○	○	
DEBUG-SUB-3	○	○		○	○	
DEBUGGING	○	○	○	○	○	
DECIMAL-POINT	○	○	○	○	○	

語	A		B	C	D	備考
	[Linux]	その他				
DECLARATIVES	○	○	○	○	○	
DEFAULT	○	○	○			
DELEGATE	○	○				
DELEGATE-ID	○	○				
DELETE	○	○	○	○	○	
DELIMITED	○	○	○	○	○	
DELIMITER	○	○	○	○	○	
DEPENDING	○	○	○	○	○	
DESCENDING	○	○	○	○	○	
DESTINATION	○	○	○	○	○	
DESTINATION-1	○	○				
DESTINATION-2	○	○				
DESTINATION-3	○	○				
DETAIL	○	○	○	○	○	
DEVICE	○	○				
DIRECT	○	○				
DISABLE	○	○	○	○	○	
DISCONNECT	○	○	○			
DISJOINING	○	○				
DISPLAY	○	○	○	○	○	
DISPLAY-EXIT	○	○				
DISPLAY-1	○	○				
DIVIDE	○	○	○	○	○	
DIVISION	○	○	○	○	○	
DOWN	○	○	○	○	○	
DUPLICATE	○	○	○			
DUPLICATES	○	○	○	○	○	
DYNAMIC	○	○	○	○	○	
EDIT-COLOR	○	○				
EDIT-CURSOR	○	○				
EDIT-MODE	○	○				
EDIT-OPTION	○	○				
EDIT-OPTION2	○	○				
EDIT-OPTION3	○	○				
EDIT-STATUS	○	○				
EGCS	○	○				
EGI	○	○	○	○	○	
EJECT	○	○				
ELSE	○	○	○	○	○	

語	A		B	C	D	備考
	[Linux]	その他				
EMI	○	○	○	○	○	
EMPTY	○	○	○			
ENABLE	○	○	○	○	○	
END	○	○	○	○	○	
END-ACCEPT	○	○				
END-ADD	○	○	○	○		
END-CALL	○	○	○	○		
END-COMPUTE	○	○	○	○		
END-DELETE	○	○	○	○		
END-DISPLAY	○	○				
END-DIVIDE	○	○	○	○		
END-EVALUATE	○	○	○	○		
END-EXEC	○	○				
END-IF	○	○	○	○		
END-INVOKE	○	○				＊
END-MULTIPLY	○	○	○	○		
END-OF-PAGE	○	○	○	○	○	
END-PERFORM	○	○	○	○		
END-READ	○	○	○	○		
END-RECEIVE	○	○	○	○		
END-RETURN	○	○	○	○		
END-REWRITE	○	○	○	○		
END-SEARCH	○	○	○	○		
END-START	○	○	○	○		
END-STRING	○	○	○	○		
END-SUBTRACT	○	○	○	○		
END-TRY		○				
END-UNSTRING	○	○	○	○		
END-WRITE	○	○	○	○		
ENDCOBOL	○	○				
ENDING	○	○				
ENTER	○	○		○	○	
ENTRY	○	○				
ENUM	○	○				
ENUM-ID	○	○				
ENVIRONMENT	○	○	○	○	○	
EOL	○	○				
EOP	○	○	○	○	○	
EOS	○	○				

語	A		B	C	D	備考
	[Linux]	その他				
EQUAL	○	○	○	○	○	
EQUALS	○	○	○			
ERASE	○	○	○			
ERROR	○	○	○	○	○	
ESI	○	○	○	○	○	
EVALUATE	○	○	○	○		
EVERY	○	○		○	○	
EXAMINE	○	○				
EXCEEDS	○	○	○			
EXCEPTION	○	○	○	○	○	
EXCEPTION-OBJECT	○	○				＊
EXCLUSIVE	○	○	○			
EXEC	○	○				
EXIT	○	○	○	○	○	
EXOR	○	○				
EXTEND	○	○	○	○	○	
EXTERNAL	○	○	○	○		
FACTORY	○	○				＊
FALSE	○	○	○	○		
FD	○	○	○	○	○	
FETCH	○	○	○			
FILE	○	○	○	○	○	
FILE-CONTROL	○	○	○	○	○	
FILE-LIMIT	○	○				
FILE-LIMITS	○	○				
FILES	○	○				
FILLER	○	○	○	○	○	
FINAL	○	○	○	○	○	
FINALLY		○				
FIND	○	○	○			
FINISH	○	○	○			
FIRST	○	○	○	○	○	
FLADD	○	○				
FLOAT-EXTENDED	○	○				
FLOAT-LONG	○	○				
FLOAT-SHORT	○	○				
FOOTING	○	○	○	○	○	
FOR	○	○	○	○	○	
FOREGROUND-COLOR	○	○				

語	A		B	C	D	備考
	[Linux]	その他				
FORMAT	○	○	○			
FORMATTED	○	○				
FREE	○	○	○			
FROM	○	○	○	○	○	
FULL	○	○				
FUNCTION	○	○	○			
GENERATE	○	○	○	○	○	
GET	○	○	○			
GIVING	○	○	○	○	○	
GLOBAL	○	○	○	○		
GO	○	○	○	○	○	
GOBACK	○	○				
GREATER	○	○	○	○	○	
GRID	○	○				
GROUP	○	○	○	○	○	
HEADING	○	○	○	○	○	
HIGH-VALUE	○	○	○	○	○	
HIGH-VALUES	○	○	○	○	○	
HIGHLIGHT	○	○				
I-O	○	○	○	○	○	
I-O-CONTROL	○	○	○	○	○	
ID	○	○				
IDENTIFICATION	○	○	○	○	○	
IF	○	○	○	○	○	
IN	○	○	○	○	○	
INCLUDE	○	○				
INDEX	○	○	○	○	○	
INDEXED	○	○	○	○	○	
INDICATE	○	○	○	○	○	
INHERITS	○	○				＊
INITIAL	○	○	○	○	○	
INITIALIZE	○	○	○	○		
INITIATE	○	○	○	○	○	
INPUT	○	○	○	○	○	
INPUT-OUTPUT	○	○	○	○	○	
INSPECT	○	○	○	○	○	
INSTALLATION	○	○		○	○	
INTERFACE	○	○				
INTERFACE-ID	○	○				

語	A		B	C	D	備考
	[Linux]	その他				
INTERNAL	○	○				
INTO	○	○	○	○	○	
INVALID	○	○	○	○	○	
INVARIANT	○	○				
INVOKE	○	○				＊
IS	○	○	○	○	○	
JAPANESE	○	○				
JOB	○	○				
JOINING	○	○				
JUST	○	○	○	○	○	
JUSTIFIED	○	○	○	○	○	
KANJI	○	○				
KEEP	○	○	○			
KEY	○	○	○	○	○	
LABEL	○	○		○	○	
LAST	○	○	○	○	○	
LD	○	○	○			
LEADING	○	○	○	○	○	
LEFT	○	○		○	○	
LEFT-JUSTIFY	○	○				
LEFTLINE	○	○				
LENGTH	○	○	○	○	○	
LESS	○	○	○	○	○	
LIMIT	○	○	○	○	○	
LIMITED	○	○				
LIMITS	○	○	○	○	○	
LINAGE	○	○	○	○	○	
LINAGE-COUNTER	○	○	○	○	○	
LINE	○	○	○	○	○	
LINE-COUNTER	○	○	○	○	○	
LINES	○	○	○	○	○	
LINKAGE	○	○	○	○	○	
LOCALLY	○	○	○			
LOCK	○	○	○	○	○	
LOW-VALUE	○	○	○	○	○	
LOW-VALUES	○	○	○	○	○	
LOWLIGHT	○	○				
MANUAL	○	○				
MEMBER	○	○	○			

語	A		B	C	D	備考
	[Linux]	その他				
MEMORY	○	○		○	○	
MERGE	○	○	○	○	○	
MESSAGE	○	○	○	○	○	
METHOD	○	○				＊
METHOD-ID	○	○				＊
MODE	○	○	○	○	○	
MODE-1	○	○				
MODE-2	○	○				
MODE-3	○	○				
MODIFY	○	○	○			
MODULES	○	○		○	○	
MORE-LABELS	○	○				
MOVE	○	○	○	○	○	
MULTICON	○	○				
MULTICONVERSATION-MODE	○	○				
MULTIPLE	○	○	○	○	○	
MULTIPLY	○	○	○	○	○	
NAMED	○	○				
NATIONAL	○	○				
NATIONAL-EDITED	○	○				
NATIVE	○	○	○	○	○	
NEGATIVE	○	○	○	○	○	
NEXT	○	○	○	○	○	
NO	○	○	○	○	○	
NOMINAL	○	○				
NONE	○	○				
NOT	○	○	○	○	○	
NOTE	○	○				
NULL	○	○	○			
NULLS	○	○				
NUMBER	○	○	○	○	○	
NUMERIC	○	○	○	○	○	
NUMERIC-EDITED	○	○	○	○		
OBJECT	○	○				＊
OBJECT-COMPUTER	○	○	○	○	○	
OCCURS	○	○	○	○	○	
OF	○	○	○	○	○	
OFF	○	○		○	○	
OMITTED	○	○		○	○	

語	A		B	C	D	備考
	[Linux]	その他				
ON	○	○	○	○	○	
ONLY	○	○	○			
OPEN	○	○	○	○	○	
OPTIONAL	○	○	○	○	○	
OR	○	○	○	○	○	
ORDER	○	○	○	○		
ORGANIZATION	○	○	○	○	○	
OTHER	○	○	○	○		
OTHERWISE	○	○				
OUTPUT	○	○	○	○	○	
OVERFLOW	○	○	○	○	○	
OVERLINE	○	○				
OVERRIDE	○	○				＊
OWNER	○	○	○			
PACKED-DECIMAL	○	○	○	○		
PADDING	○	○	○	○		
PAGE	○	○	○	○	○	
PAGE-COUNTER	○	○	○	○	○	
PARTIAL		○				
PASSWORD	○	○				
PERFORM	○	○	○	○	○	
PF	○	○	○	○	○	
PH	○	○	○	○	○	
PIC	○	○	○	○	○	
PICTURE	○	○	○	○	○	
PLUS	○	○	○	○	○	
POINTER	○	○	○	○	○	
POSITION	○	○	○	○	○	
POSITIONING	○	○				
POSITIVE	○	○	○	○	○	
PREFIX	○	○				
PRESENT	○	○	○			
PREVIOUS	○	○				
PRINTING	○	○	○	○	○	
PRIOR	○	○	○			
PRIVATE	○	○				
PROCEDURE	○	○	○	○	○	
PROCEDURES	○	○		○	○	
PROCEED	○	○		○	○	

語	A		B	C	D	備考
	[Linux]	その他				
PROCESSING	○	○				
PROGRAM	○	○	○	○	○	
PROGRAM-ID	○	○	○	○	○	
PROGRAM-STATUS	○	○				
PROMPT	○	○				
PROPERTY	○	○				＊
PROTECTED	○	○	○			
PROTOTYPE	○	○				＊
PUBLIC	○	○				
PURGE	○	○	○	○		
QUEUE	○	○	○	○	○	
QUOTE	○	○	○	○	○	
QUOTES	○	○	○	○	○	
RAISE	○	○				＊
RAISING	○	○				＊
RANDOM	○	○	○	○	○	
RANGE	○	○				
RD	○	○	○	○	○	
READ	○	○	○	○	○	
READY	○	○	○			
REALM	○	○	○			
RECEIVE	○	○	○	○	○	
RECONNECT	○	○	○			
RECORD	○	○	○	○	○	
RECORD-NAME	○	○	○			
RECORD-OVERFLOW	○	○				
RECORDING	○	○				
RECORDS	○	○	○	○	○	
REDEFINES	○	○	○	○	○	
REEL	○	○	○	○	○	
REFERENCE	○	○	○	○		
REFERENCES	○	○		○	○	
RELATION	○	○	○			
RELATIVE	○	○	○	○	○	
RELEASE	○	○	○	○	○	
RELOAD	○	○				
REMAINDER	○	○	○	○	○	
REMARKS	○	○				
REMOVAL	○	○	○	○	○	

語	A		B	C	D	備考
	[Linux]	その他				
RENAMES	○	○	○	○	○	
REORG-CRITERIA	○	○				
REPEATED	○	○	○			
REPLACE	○	○	○	○		
REPLACING	○	○	○	○	○	
REPORT	○	○	○	○	○	
REPORTING	○	○	○	○	○	
REPORTS	○	○	○	○	○	
REPOSITORY	○	○				＊
REQUIRED	○	○				
RERUN	○	○		○	○	
RESERVE	○	○	○	○	○	
RESET	○	○	○	○	○	
RESUME		○				
RETAINING	○	○	○			
RETRIEVAL	○	○	○			
RETURN	○	○	○	○	○	
RETURNING	○	○				
RETURN-CODE	○	○				
REVERSE-VIDEO	○	○				
REVERSED	○	○		○	○	
REWIND	○	○	○	○	○	
REWRITE	○	○	○	○	○	
RF	○	○	○	○	○	
RH	○	○	○	○	○	
RIGHT	○	○	○	○	○	
RIGHT-JUSTIFY	○	○				
ROLL-OUT	○	○				
ROLLBACK	○	○	○			
ROUNDED	○	○	○	○	○	
RUN	○	○	○	○	○	
SA	○	○				
SAME	○	○	○	○	○	
SAVED-AREA	○	○				
SCREEN	○	○				
SD	○	○	○	○	○	
SEARCH	○	○	○	○	○	
SECTION	○	○	○	○	○	
SECURE	○	○				

語	A		B	C	D	備考
	[Linux]	その他				
SECURITY	○	○		○	○	
SEEK	○	○				
SEGMENT	○	○	○	○	○	
SEGMENT-LIMIT	○	○		○	○	
SELECT	○	○	○	○	○	
SELECTED	○	○				
SELECTIVE	○	○				
SELF	○	○				＊
SEND	○	○	○	○	○	
SENTENCE	○	○	○	○	○	
SEPARATE	○	○	○	○	○	
SEQUENCE	○	○	○	○	○	
SEQUENTIAL	○	○	○	○	○	
SERVICE	○	○				
SESSION	○	○	○			
SET	○	○	○	○	○	
SHARED	○	○	○			
SHIFT-IN	○	○				
SHIFT-OUT	○	○				
SIGN	○	○	○	○	○	
SIMPLE	○	○				
SINGLE	○	○				
SIZE	○	○	○	○	○	
SKIP1	○	○				
SKIP2	○	○				
SKIP3	○	○				
SORT	○	○	○	○	○	
SORT-CONTROL	○	○				
SORT-CORE-SIZE	○	○				
SORT-FILE-SIZE	○	○				
SORT-MERGE	○	○	○	○	○	
SORT-MESSAGE	○	○				
SORT-MODE-SIZE	○	○				
SORT-RETURN	○	○				
SORT-STATUS	○	○				
SOURCE	○	○	○	○	○	
SOURCE-COMPUTER	○	○	○	○	○	
SPACE	○	○	○	○	○	
SPACE-FILL	○	○				

語	A		B	C	D	備考
	[Linux]	その他				
SPACES	○	○	○	○	○	
SPECIAL-NAMES	○	○	○	○	○	
STANDARD	○	○	○	○	○	
STANDARD-1	○	○	○	○	○	
STANDARD-2	○	○	○	○		
START	○	○	○	○	○	
STATIC	○	○				
STATION	○	○				
STATIONS	○	○				
STATUS	○	○	○	○	○	
STOP	○	○	○	○	○	
STORE	○	○	○			
STRING	○	○	○	○	○	
SUB-QUEUE-1	○	○	○	○	○	
SUB-QUEUE-2	○	○	○	○	○	
SUB-QUEUE-3	○	○	○	○	○	
SUB-SCHEMA	○	○	○			
SUBRANGE	○	○				
SUBSCHEMA-NAME	○	○				
SUBTRACT	○	○	○	○	○	
SUCCESSIVE	○	○				
SUFFIX	○	○				
SUM	○	○	○	○	○	
SUPER	○	○				＊
SUPPRESS	○	○	○	○	○	
SYMBOLIC	○	○	○	○	○	
SYNC	○	○		○	○	
SYNCHRONIZED	○	○		○	○	
SYSTEM-OBJECT	○	○				
TABLE	○	○	○	○	○	
TALLY	○	○				
TALLYING	○	○	○	○	○	
TAPE	○	○	○	○	○	
TENANT	○	○				
TERMINAL	○	○	○	○	○	
TERMINATE	○	○	○	○	○	
TEST	○	○	○	○		
TEXT	○	○	○	○	○	
THAN	○	○	○	○	○	

語	A		B	C	D	備考
	[Linux]	その他				
THEN	○	○	○	○		
THROUGH	○	○	○	○	○	
THRU	○	○	○	○	○	
TIME	○	○	○	○	○	
TIMES	○	○	○	○	○	
TITLE	○	○				
TO	○	○	○	○	○	
TOP	○	○	○	○	○	
TRACE	○	○				
TRACK	○	○				
TRACK-AREA	○	○				
TRACK-LIMIT	○	○				
TRACK-OVERFLOW	○	○				
TRACKS	○	○				
TRAILING	○	○	○	○	○	
TRAILING-SIGN	○	○				
TRANSACTION	○	○				
TRUE	○	○	○	○		
TRY		○				
TYPE	○	○	○	○	○	
TYPEDEF	○	○				
UNDERLINE	○	○				
UNEQUAL	○	○	○			
UNICODE1		○				
UNIT	○	○	○	○	○	
UNIVERSAL	○	○				＊
UNLOCK	○	○				
UNSTRING	○	○	○	○	○	
UNTIL	○	○	○	○	○	
UP	○	○	○	○	○	
UPDATE	○	○	○			
UPON	○	○	○	○	○	
USAGE	○	○	○	○	○	
USAGE-MODE	○	○	○			
USE	○	○	○	○	○	
USING	○	○	○	○	○	
VALID	○	○	○			
VALIDATE	○	○	○			
VALUE	○	○	○	○	○	

語	A		B	C	D	備考
	[Linux]	その他				
VALUES	○	○	○	○	○	
VARYING	○	○	○	○	○	
WAIT	○	○	○			
WHEN	○	○	○	○	○	
WHEN-COMPILED	○	○				
WITH	○	○	○	○	○	
WITHIN	○	○	○			
WORDS	○	○		○	○	
WORKING-STORAGE	○	○	○	○	○	
WRITE	○	○	○	○	○	
WRITE-ONLY	○	○				
ZERO	○	○	○	○	○	
ZERO-FILL	○	○				
ZEROES	○	○	○	○	○	
ZEROS	○	○	○	○	○	
+	○	○		○	○	
-	○	○		○	○	
*	○	○		○	○	
/	○	○		○	○	
**	○	○		○	○	
>	○	○		○	○	
<	○	○		○	○	
=	○	○		○	○	
>=	○	○		○		
<=	○	○		○		
&	○	○				
->	○	○				
*>	○	○				
::	○	○				*

付録B　システムの定量制限

ここでは、COBOL処理系の定量制限を示します。なお、ここで示す制限は、論理的な値であり、プログラムが動作するための制限は、オペレーティングシステムおよび入出力管理システムに依存します。

B.1　正書法

項目	値
可変形式の最大長（バイト数）	251
自由形式の最大長（バイト数）	251

B.2　中核のデータ部

項目	値	
	[Win16]	その他
データ項目の最大長（バイト数）	64770	2147483647
英字項目、英数字項目の最大長（文字数）	64770	2147483647
英数字編集項目の最大長（文字数）	64770	2147483647
ブール項目の最大長（文字数）	64770	2147483647
数字編集項目の最大長（文字数）	160	
日本語項目の最大長（文字数）	32385	1073741823
日本語編集項目の最大長（文字数）	32385	1073741823
添字の最大値	64770	2147483647
添字の次元数の最大値	7	
OCCURS句の繰返し回数の最大値	2147483647	
OCCURS句のKEY IS指定で指定するキー項目の長さの最大（バイト数）	256	
1つの表要素に関するキーの総数の最大	2147483647	
OCCURS句のINDEXED BY指定の指標名の個数の最大	60	
暗黙のポインタ修飾の深さの最大	7	

B.3　中核の手続き部

項目		値
“ACCEPT 一意名-1 FROM 呼び名”の呼び名に機能名CONSOLE を対応付けた場合の一意名-1の大きさの最大（バイト数）		制限なし
1つのINITIALIZE文の一意名の個数の最大		制限なし
1つのINITIALIZE文の一意名で、添字付け、ポインタ付けまたは部分参照されたデータ項目または可変長項目の数		制限なし
“INSPECT ～ CONVERTING 一意名-6 TO 一意名-7”の、一意名-6と一意名-7の大きさの最大（バイト数）	一意名-6と一意名-7が日本語／日本語編集項目以外の場合	256
	一意名-6と一意名-7が日本語／日本語編集項目の場合	制限なし
“PERFORM ～ 一意名-1／整数-1 TIMES ～ ”の一意名-1／整数-1の最大値		制限なし

項目	値
“DISPLAY 一意名-1 UPON 呼び名”の呼び名に機能名 ENVIRONMENT-NAMEを対応付けた場合の一意名-1の大きさの最大(バイト数)	512 (*1)
“DISPLAY 一意名-1 UPON 呼び名”の呼び名に機能名 ENVIRONMENT-VALUE を対応付けた場合の一意名-1の大きさの最大(バイト数)	2048 (*1)
“ACCEPT 一意名-1 FROM 呼び名”の呼び名に機能名 ENVIRONMENT-VALUE を対応付けた場合の一意名-1の大きさの最大(バイト数)	2048 (*1)

*1 ： [Win32]固有

B.4 順ファイル

項目	値
レコードの最大長(バイト数)	32760
LINAGE句で指定する論理ページの最大行数	制限なし

B.5 相対ファイル

項目	値
レコードの最大長(バイト数)	32760

B.6 索引ファイル

項目	値
レコードの最大長(バイト数)	32760
RECORD KEY句のデータ項目の個数の最大	254 (*1)
RECORD KEY句のデータ項目の長さの総和の最大値(バイト数)	254 (*2)
ALTERNATE RECORD KEY句のデータ項目の個数の最大	254 (*1)
1つのALTERNATE RECORD KEY句のデータ項目の長さの総和の最大値 (バイト数)	254 (*2)
ALTERNATE RECORD KEY句の個数の最大	125

*1 ： RECORD KEY句とALTERNATE RECORD KEY句のデータ項目の総数は、最大255個

*2 ： RECORD KEY句とALTERNATE RECORD KEY句のデータ項目の総長は、最大255バイト

B.7 プログラム間連絡

項目	値
手続き部の見出しのパラメタの個数の最大	制限なし
CALL文のUSING 指定のパラメタの個数の最大	制限なし
ENTRY 文のUSING 指定のパラメタの個数の最大	制限なし
コマンド行文字の長さ(文字数)	260 (引数は255)

B.8 整列併合

項目	値	
	[Win16][Win32]	その他
レコードの最大長(バイト数)	21484	32760
SORT文およびMERGE 文で指定するキーデータ項目の個数の最大	64	
SORT文およびMERGE 文で指定するキーデータ項目の長さの総和の最大(バイト数)	レコードの最大長	
SORT文およびMERGE 文のキーデータ項目として指定する英字項目、英数字項目の最大長(文字数)	16382	
SORT文およびMERGE 文のキーデータ項目として指定する日本語項目の最大長(文字数)	8191	
SORT文およびMERGE 文のUSING で指定する入力ファイルの個数の最大	16	

B.9 原始文操作

項目	値
COPY文の原文語の長さの最大(文字数)	324
“COPY ～ JOINING 語-4”の語-4の長さの最大(文字数)	28
REPLACE 文の原文語の長さの最大	324

B.10 表示ファイル

項目	値
レコードの最大長(バイト数)	32767

B.11 オブジェクト指向プログラミング

項目	値
手続き部の見出しのパラメタの個数の最大	制限なし
INVOKE文のUSING 指定のパラメタの個数の最大	制限なし

付録C　コード表

ここでは、以下のコード系における文字の内部コードを示します。

- EBCDIC (Extended Binary Coded Decimal Interchange Code)
- ASCII (American National Standard Code for Information Interchange, X3.4-1968)
- JIS8単位コード(ISO646準拠)

EBCDIC、ASCIIおよびJIS8単位コードは、特殊名段落のALPHABET句のEBCDIC、STANDARD-1およびSTANDARD-2にそれぞれ対応します。表の見かたは、以下のとおりです。

- 1つの文字は1バイトの内部コードで表現されます。表の“上位4ビット”および“下位4ビット”の欄の0～Fの文字は、それぞれ第1～第4ビットおよび第5～第8ビットを16進数表示したものです。表の中の文字は、上位4ビットと下位4ビットの組合せで表現されます。
- 表の中の文字の位置は、文字の大小順序を示します。表の右にある文字は左にある文字よりも大きく、下の行にある文字は上の行にある文字よりも大きいことを示します。

C.1　EBCDICにおける文字の内部コード

上位4ビット / 下位4ビット	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0					空白	&	−			ソ			{	}	¥	0
1					。	ェ	／		aア	jタ	～￣		A	J		1
2					「	ォ			bイ	kチ	sヘ		B	K	S	2
3					」	ャ			cウ	lツ	tホ		C	L	T	3
4					、	ュ			dエ	mテ	uマ		D	M	U	4
5					・	ョ			eオ	nト	vミ		E	N	V	5
6					ヲ	ッ			fカ	oナ	wム		F	O	W	6
7					ァ				gキ	pニ	xメ		G	P	X	7
8					ィ	−			hク	qヌ	yモ		H	Q	Y	8
9					ゥ			ヽ	iケ	rネ	zヤ		I	R	Z	9
A					£	!	¦	:	コ	ノ	ユ	レ				
B					.	¥$	,	#				ロ				
C					＜	＊	％	@	サ		ヨ	ワ				
D					(	)	_	‘	シ	ハ	ラ	ン				
E					＋	;	＞	＝	ス	ヒ	リ	゛				
F					｜	¬	？	″	セ	フ	ル	°				

参考

1つの欄の中に2つの文字が記入されているところは、2種類の文字が同じ内部コードで表現されることを示します。左側は英小文字用コード系の文字、右側はカナ文字用コード系の文字です。

C.2 ASCIIにおける文字の内部コード

上位4ビット / 下位4ビット	0	1	2	3	4	5	6	7
0			空白	0	@	P	`	p
1			!	1	A	Q	a	q
2			"	2	B	R	b	r
3			#	3	C	S	c	s
4			$	4	D	T	d	t
5			%	5	E	U	e	u
6			&	6	F	V	f	v
7			'	7	G	W	g	w
8			(	8	H	X	h	x
9			)	9	I	Y	i	y
A			*	:	J	Z	j	z
B			+	;	K	[	k	{
C			,	<	L	*1	l	|
D			-	=	M	]	m	}
E			.	>	N	^	n	~*2
F			/	?	O	_	o	

*1 ： システムや装置によっては、この内部コードが“¥”を表すことがあります。

*2 ： “～”は、上付きの波線です。

C.3 JIS8単位コードにおける文字の内部コード

上位4ビット / 下位4ビット	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0			SP	0	@	P	`	p				ー	タ	ミ		
1			!	1	A	Q	a	q			。	ア	チ	ム		
2			"	2	B	R	b	r			「	イ	ツ	メ		
3			#	3	C	S	c	s			」	ウ	テ	モ		
4			$	4	D	T	d	t			、	エ	ト	ヤ		
5			%	5	E	U	e	u			・	オ	ナ	ユ		
6			&	6	F	V	f	v			ヲ	カ	ニ	ヨ		
7			'	7	G	W	g	w			ァ	キ	ヌ	ラ		
8			(	8	H	X	h	x			ィ	ク	ネ	リ		
9			)	9	I	Y	i	y			ゥ	ケ	ノ	ル		
A			*	:	J	Z	j	z			ェ	コ	ハ	レ		
B			+	;	K	[	k	{			ォ	サ	ヒ	ロ		
C			,	<	L	¥	l	|			ャ	シ	フ	ワ		
D			-	=	M	]	m	}			ュ	ス	ヘ	ン		
E			.	>	N	^	n	～*1			ョ	セ	ホ	゛		
F			/	?	O	_	o				ッ	ソ	マ	゜		

*1 ：“～”は、上付きの波線です。

付録D　中間結果

算術文、算術式および関数の値を求めるときに発生する一時的な演算結果を、「中間結果」といいます。中間結果は、算術文および算術式の中の1つの二項演算の結果、および1つの関数の結果のことです。中間結果の属性および精度は、ここで説明する規則に従って決定されます。ここで説明する規則は、データ項目と中間結果、関数値と中間結果および中間結果どうしの二項演算にも適用されます。

実行時、中間結果の値がその精度を超えた場合、桁落としされた値が中間結果の値になります。したがって、算術文の作用対象、算術式の作用対象または関数の引数に、桁数の大きいデータ項目を指定した場合、その結果を求める途中で、予期しない桁落ちが発生することがあるので注意が必要です。

D.1　中間結果の属性と精度

中間結果の属性には、固定小数点属性と浮動小数点属性の2種類があります。

固定小数点属性の中間結果は、2進項目、内部10進項目または外部10進項目として扱われます。固定小数点属性の精度は、全桁数と小数部桁数で表します。

浮動小数点属性の中間結果は、内部浮動小数点項目として扱われます。浮動小数点属性の精度には、倍精度と単精度の2種類があります。倍精度の中間結果は、倍精度浮動小数点項目として扱われます。単精度の中間結果は、単精度浮動小数点項目として扱われます。

中間結果の属性と精度の決め方は、算術演算および関数の種類によって異なります。

D.2　四則演算の中間結果

四則演算(加算、減算、乗算および除算)の中間結果の属性は、以下のとおりです。

- 四則演算の両方の作用対象が数字項目、数字定数、または固定小数点属性の中間結果の場合、中間結果の属性は固定小数点属性です。
- 四則演算の少なくとも一方の作用対象が、浮動小数点項目、浮動小数点定数、または浮動小数点属性の中間結果の場合、中間結果の属性は浮動小数点属性です。

D.2.1　固定小数点属性の四則演算の中間結果の精度

固定小数点属性の四則演算の中間結果の精度は、全桁数と小数部桁数で表します。中間結果の全桁数にPの桁数は含みません。

固定小数点属性の四則演算の中間結果の精度は、以下の手順で決定されます。

1. 四則演算を含む算術文または算術式の作用対象の属性から、小数部基準桁数を求めます。
2. 四則演算の作用対象と小数部基準桁数から、算術演算暫定桁数を求めます。
3. 算術演算暫定桁数と小数部基準桁数から、四則演算の中間結果の精度を決定します。

小数部基準桁数を求める

最初に、小数部基準桁数を求めます。「小数部基準桁数」とは、中間結果として保証すべき最大の小数部桁数です。

以下に小数部基準桁数の求め方を示します。

算術文または算術式の場合

算術文または算術式のすべての作用対象(受取り側項目も含む)を選考の対象とします。そして、それらの小数部桁数のうちの最大のものを小数部基準桁数とします。ただし、算術文または算術式中で、浮動小数点項目、べき乗の指数や除算の除数となる作用対象および関数の引数は除きます。また、受取り側項目にROUNDEDを指定している場合、受取り側項目の小数部桁数に1を加えたものを受取り側項目の小数部桁数として扱います。小数部基準桁数を決定する作用対象がない場合は、小数部基準桁数は0とします。

比較条件またはEVALUATE文の場合

1つの比較条件またはEVALUATE文の1つの選択主体および選択対象の組に現れるすべての作用対象を選考の対象とします。そして、それらの小数部桁数のうちの最大のものを小数部基準桁数とします。ただし、比較条件またはEVALUATE文中で、べき乗の指数および除算の除数となる作用対象は除きます。小数部基準桁数を決定する作用対象がない場合は、小数部基準桁数は0とします。

算術演算暫定桁数を求める

小数部基準桁数を求めた後、算術演算暫定桁数を求めます。「算術演算暫定桁数」とは、その四則演算の取ることができる最大値の桁数であり、最終的な中間結果の桁数を得るための桁数になります。

算術演算暫定桁数は、以下の値を使って実際に演算することによって求めます。減算の場合の算術演算暫定桁数は、加算に準じて求めます。

作用対象がデータ項目の場合

除数でない場合、そのデータ項目のPICTURE句の文字列が取ることができる最大値を使います。例えば、“PIC S9(5)V99”の場合、99999.99を使います。

除数の場合、そのデータ項目のPICTURE句の文字列が取ることができる正数の最小値を使います。例えば、“PIC S9(5)V99”の場合、0.01を使います。

作用対象のデータ項目がint型2進整数データ項目の場合は、全けた数と小数部けた数を下記の表に示す値とみなし、上記規則を適用します。

USAGE句種別	全桁数	小数部桁数
BINARY-CHAR UNSIGNED	3	0
BINARY-SHORT SIGNED	5	0
BINARY-LONG SIGNED	10	0
BINARY-DOUBLE SIGNED	19	0

作用対象が定数の場合

定数の値の絶対値を使います。

作用対象が中間結果の場合

中間結果の精度を決定したときに求めた、以下の演算結果を使います。

- 算術演算暫定桁数が29桁以下の場合、算術演算暫定桁数を求めたときの演算結果を使います。
- 算術演算暫定桁数が29桁を超える場合、中間結果の精度を持つ数の最大値(30桁の99….…99)を使います。

算術演算暫定桁数の小数部桁数

算術演算暫定桁数の小数部桁数を、下表に示します。

四則演算	小数部桁数
加算(OP1 + OP2) 減算(OP1 - OP2)	MAX(OP1d,OP2d)
乗算(OP1 * OP2)	OP1d+OP2d
除算(OP1 / OP2)	MAX(Bd,OP1d-OP2d)

OP1dおよびOP2dは、それぞれOP1およびOP2の小数部桁数を示します。

Bdは、小数部基準桁数を示します。

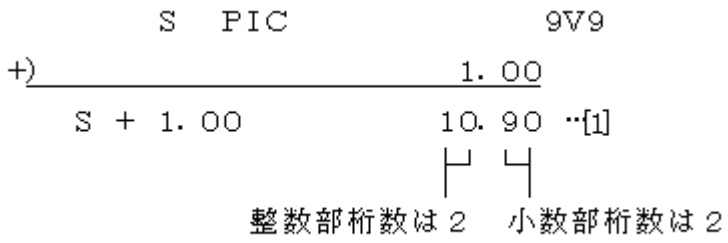
例

・ 算術演算暫定桁数を求める例(例1)

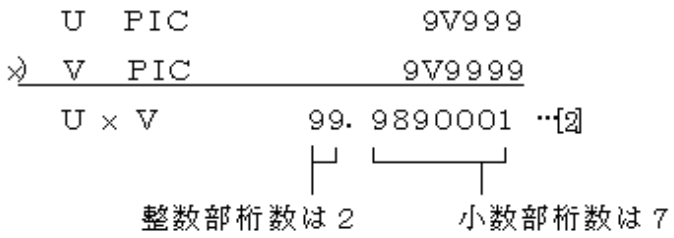
```
S PIC S9V9.
T PIC S9V99.
U PIC S9V999.
V PIC S9V9999.
X PIC S9999V9999

COMPUTE X = S + 1.00 + U * V.
```

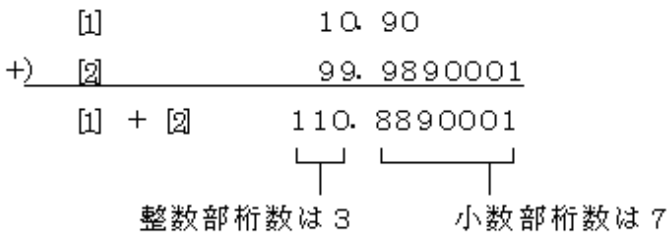
－ S＋1.00の中間結果の精度は、以下のように決定されます。



－ U＊Vの中間結果の精度は、以下のように決定されます。



－ (S＋1.00の中間結果)＋(U＊Vの中間結果)の中間結果の精度は、以下のように決定されます。

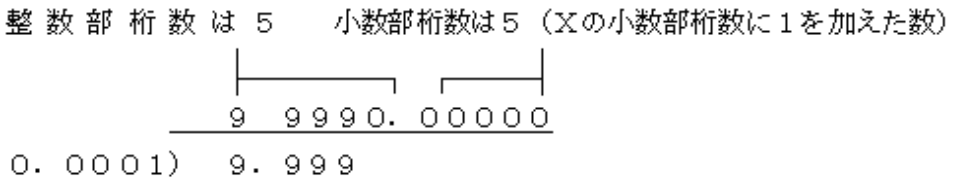


・ 算術演算暫定桁数を求める例(例2)

```
DIVIDE U BY V GIVING X ROUNDED.
```

作用対象の属性は例1と同じとします。

算術演算U／Vの中間結果の精度は、以下のように決定されます。

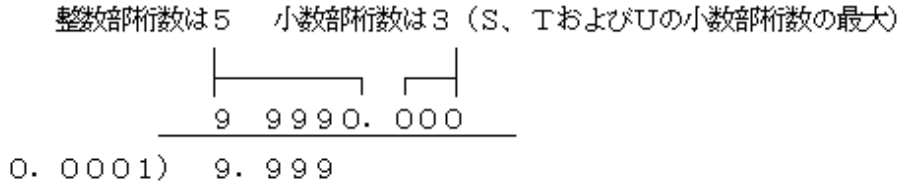


・ 算術演算暫定桁数を求める例(例3)

```
COMPUTE S = S + T + (S * U) / (U / V)

作用対象の属性は例1と同じとします。
```

この算術文中の算術演算U／Vの中間結果の精度は、以下のように決定されます。



四則演算の中間結果の精度を決定する

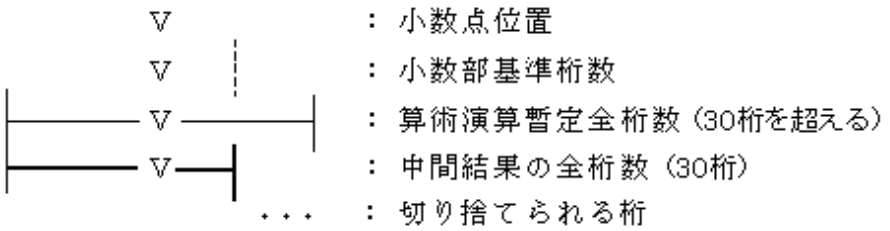
算術演算暫定桁数を求めた後、四則演算の中間結果の精度を決定します。

・ 算術演算暫定桁数が30桁以下の場合、算術演算暫定桁数をそのまま中間結果の精度にします。

・ 算術演算暫定桁数が30桁を超える場合、全体で30桁になるように桁落としたものを中間結果の精度にします。

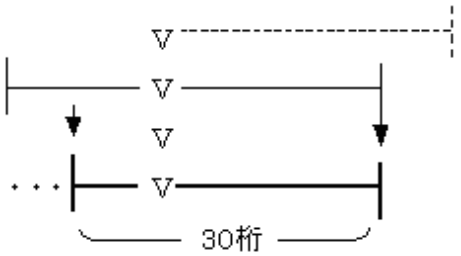
以下に、算術演算暫定桁数が30桁を超える場合の桁落としの規則を説明します。

なお、説明の中で使う記号の意味は、以下のとおりです。



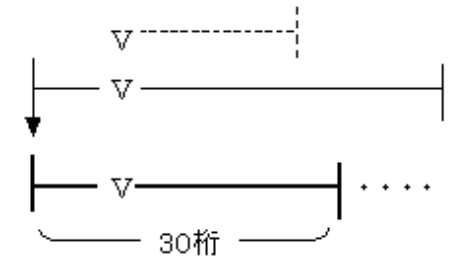
小数部桁数が小数部基準桁数以下の場合

小数部桁数は上記で求めた値とし、整数部の上位を桁落とします。



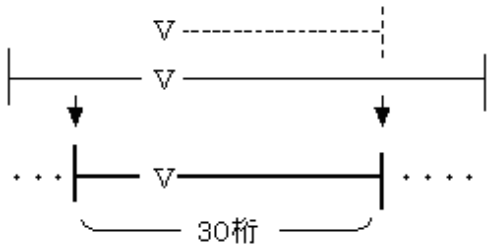
小数部桁数が小数部基準桁数より大きく、かつ、整数部桁数と小数部基準桁数の和が30桁以下の場合

整数部桁数は上記で求めた値とし、小数部の下位を桁落としします。



小数部桁数が小数部基準桁数より大きく、かつ、整数部桁数と小数部基準桁数の和が30桁を超える場合

小数部桁数は小数部基準桁数とし、小数部の下位および整数部の上位を桁落としします。



D.2.2 浮動小数点属性の四則演算の中間結果の精度

浮動小数点属性の四則演算の中間結果の精度は、以下のとおりです。

- 四則演算の少なくとも一方の作用対象の精度が倍精度の場合、中間結果の精度は倍精度です。
- 四則演算の両方の作用対象の精度が単精度の場合、中間結果の精度は単精度です。

D.3 べき乗の中間結果

べき乗の中間結果の属性は、以下のとおりです。

- 指数が整数の場合、べき乗の中間結果の属性は固定小数点属性です。
- 指数が整数でない場合、べき乗の中間結果の属性は浮動小数点属性です。

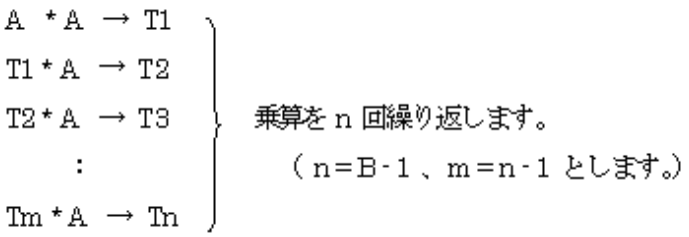
D.3.1 固定小数点属性のべき乗の中間結果の精度

固定小数点属性のべき乗の中間結果の精度は、指数が定数の場合と変数の場合で異なります。

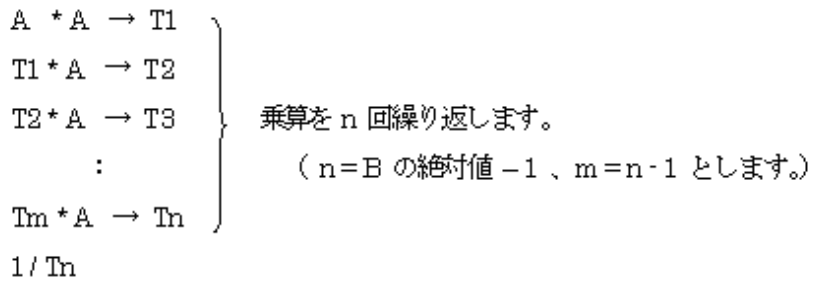
指数が定数の場合

指数が定数(整数)の場合、べき乗(A＊＊B)は下記の乗算に展開されます。このとき、それぞれの乗算の中間結果(T1、T2、…、Tn)およびBが負数のときの除算の中間結果は、乗算および除算の中間結果の精度に従います。ただし、定数（整数）の絶対値が30を超える場合は、“指数が変数の場合”の規則が適用されます。

・ Bが正数の場合



・ Bが負数の場合



指数が変数の場合

指数が変数(整数)の場合、べき乗の中間結果の全桁数は30桁、小数部桁数は小数部基準桁数です。小数部基準桁数は、“D.2.1 固定小数点属性の四則演算の中間結果の精度”で説明した値です。

D.3.2 浮動小数点属性のべき乗の中間結果の精度

浮動小数点属性のべき乗の中間結果の精度は、倍精度です。

D.4 関数値の属性と精度

数字関数および整数関数の関数値の属性と精度を、下表に示します。

関数名	関数の型	関数値の属性と精度
ACOS	数字	倍精度浮動小数点
ANNUITY	数字	関数値を求める式の中間結果に従う
ASIN	数字	倍精度浮動小数点
ATAN	数字	倍精度浮動小数点
COS	数字	倍精度浮動小数点
DATE-OF-INTEGER	整数	8桁の固定小数点
DAY-OF-INTEGER	整数	7桁の固定小数点
FACTORIAL	整数	関数値を求める式の中間結果に従う
INTEGER	整数	引数の整数部桁数に1を加えた桁数の固定小数点
INTEGER-OF-DATE	整数	8桁の固定小数点
INTEGER-OF-DAY	整数	7桁の固定小数点
INTEGER-PART	整数	引数の整数部桁数と同じ桁数の固定小数点
LENG	整数	10桁の固定小数点
LENGTH	整数	引数の長さが収まるだけの桁数の固定小数点
LOG	数字	倍精度浮動小数点

関数名	関数の型	関数値の属性と精度
LOG10	数字	倍精度浮動小数点
MAX	引数の型に依存	関数値を求める式の中間結果に従う
MEAN	数字	関数値を求める式の中間結果に従う
MEDIAN	数字	関数値を求める式の中間結果に従う
MIDRANGE	数字	関数値を求める式の中間結果に従う
MIN	引数の型に依存	関数値を求める式の中間結果に従う
MOD	整数	関数値を求める式の中間結果に従う
NUMVAL	数字	引数の桁数の固定小数点
NUMVAL-C	数字	引数の桁数の固定小数点
ORD	整数	3桁の固定小数点
ORD-MAX	整数	9桁の固定小数点
ORD-MIN	整数	9桁の固定小数点
PRESENT-VALUE	数字	倍精度浮動小数点
RANDOM	数字	関数値を求める式の中間結果に従う
RANGE	引数の型に依存	関数値を求める式の中間結果に従う
REM	数字	関数値を求める式の中間結果に従う
SIN	数字	倍精度浮動小数点
SQRT	数字	倍精度浮動小数点
STANDARD-DEVIATION	数字	倍精度浮動小数点
STORED-CHAR-LENGTH	整数	引数の長さが収まるだけの桁数の固定小数点 [Win32][WinIPF][Winx64][Solaris][Linux][LinuxIPF][Linux64][.NET]固有
SUM	引数の型に依存	関数値を求める式の中間結果に従う
TAN	数字	倍精度浮動小数点
VARIANCE	数字	関数値を求める式の中間結果に従う

付録E　機能差

ここでは、各システムの機能差を示します。

E.1　自由形式の正書法

自由形式の正書法は、下表で○がついているシステムの固有の機能です。

DS	HP	Solaris	Win16	Win32	WinIPF	Winx64	Linux	LinuxIPF	Linux64	.NET
		○		○	○	○	○	○	○	○

E.2　順ファイル

ASSIGN句へPRINTER-nを指定できるのは、下表で○がついているシステムの固有の機能です。

詳細は、“4.3.1.3 ASSIGN句（順ファイル・相対ファイル・索引ファイル）”を参照してください。

DS	HP	Solaris	Win16	Win32	WinIPF	Winx64	Linux	LinuxIPF	Linux64	.NET
		○	○	○	○	○	○	○	○	○

E.3　データ項目定義

数値の扱い

COMP-5の扱いは、下表で○がついているシステムとその他のシステムで異なります。

DS	HP	Solaris	Win16	Win32	WinIPF	Winx64	Linux	LinuxIPF	Linux64	.NET
			○	○	○	○	○	○	○	○

BINARY-CHAR、BINARY-SHORT、BINARY-LONG、BINARY-DOUBLEは、下表で○がついているシステムの固有の機能です。

詳細は、“5.4.15 USAGE句”を参照してください。

DS	HP	Solaris	Win16	Win32	WinIPF	Winx64	Linux	LinuxIPF	Linux64	.NET
		○		○	○	○	○	○	○	○

型

型は、下表で○がついているシステムの固有の機能です。

DS	HP	Solaris	Win16	Win32	WinIPF	Winx64	Linux	LinuxIPF	Linux64	.NET
		○		○	○	○	○	○	○	○

E.4　定数

数字定数の桁数は、下表で○がついているシステムとその他のシステムで異なります。

詳細は、“1.2.3.1 数字定数”を参照してください。

DS	HP	Solaris	Win16	Win32	WinIPF	Winx64	Linux	LinuxIPF	Linux64	.NET
		○		○	○	○	○	○	○	○

E.5　表示ファイル

あて先

指定可能なあて先は、システムごとに異なります。

詳細は、“4.3.1.32 SYMBOLIC DESTINATION句（表示ファイル）”を参照してください。

あて先と指定可能な句の組合せ

表示ファイルでは、あて先と指定可能な句の組合せが、システムごとに異なります。

詳細は、“4.3.1 ファイル管理段落（FILE-CONTROL）”の“表示ファイル機能のファイル管理記述項”の表を参照してください。

USE FOR DEAD-LOCK文

USE FOR DEAD-LOCK文は、[DS]で指定できます。

特殊レジスタ

特殊レジスタEDIT-OPTION2、EDIT-OPTION3は、下表で○がついているシステムの固有の機能です。

DS	HP	Solaris	Win16	Win32	WinIPF	Winx64	Linux	LinuxIPF	Linux64	.NET
		○		○	○	○	○	○	○	○

E.6　プログラム間連絡

WITH指定

手続き部の見出しのPROCEDURE DIVISION、CALL文およびENTRY文にWITH指定をすることでリンケージ種別を指定できるのは、下表で○がついているシステムの固有の機能です。

それぞれ“6.2 手続き部の見出し”、“6.4.6 CALL文（プログラム間連絡）”、“6.4.16 ENTRY文（プログラム間連絡）”を参照してください。

DS	HP	Solaris	Win16	Win32	WinIPF	Winx64	Linux	LinuxIPF	Linux64	.NET
			○	○						

日本語のプログラム名

CALL文およびCANCEL文に日本語項目または日本語定数を指定することができるのは、下表で○がついているシステムの固有の機能です。

詳細は、“6.4.6 CALL文（プログラム間連絡）”、“6.4.7 CANCEL文（プログラム間連絡）”を参照してください。

DS	HP	Solaris	Win16	Win32	WinIPF	Winx64	Linux	LinuxIPF	Linux64	.NET
		○		○	○	○	○	○	○	○

RETURNING指定

手続き部の見出しのPROCEDURE DIVISIONおよびCALL文にRETURNING指定することができるのは、下表で○がついているシステムの固有の機能です。

詳細は、“6.2 手続き部の見出し”、“6.4.6 CALL文（プログラム間連絡）”を参照してください。

DS	HP	Solaris	Win16	Win32	WinIPF	Winx64	Linux	LinuxIPF	Linux64	.NET
		○		○	○	○	○	○	○	○

外部名

外部名を定義することができるのは、下表で○がついているシステムの固有の機能です。

DS	HP	Solaris	Win16	Win32	WinIPF	Winx64	Linux	LinuxIPF	Linux64	.NET
		○		○	○	○	○	○	○	○

E.7 文

WRITE文(順ファイル)

下表で○がついているシステムでは、印刷ファイル、行順ファイルに対するWRITE文にADVANCING指定を書くことができます。

下表で●がついているシステムでは、印刷ファイルに対するWRITE文にだけADVANCING指定を書くことができます。

詳細は、“6.4.54 WRITE文（順ファイル）”を参照してください。

DS	HP	Solaris	Win16	Win32	WinIPF	Winx64	Linux	LinuxIPF	Linux64	.NET
●	○	○	●	○	○	○	●	○	○	○

USE FOR DEAD-LOCK文

USE FOR DEAD-LOCK文は、下表で○がついているシステムの固有の機能です。

DS	HP	Solaris	Win16	Win32	WinIPF	Winx64	Linux	LinuxIPF	Linux64	.NET
		○		○	○	○	○	○	○	

E.8 関数

下表の関数は、○がついているシステムの固有の機能です。

関数	DS	HP	Solaris	Win16	Win32	WinIPF	Winx64	Linux	LinuxIPF	Linux64	.NET
STORED-CHAR-LENGTH			○		○	○	○	○	○	○	○
UCS2-OF			○		○	○	○	○	○	○	○
UTF8-OF			○		○	○	○	○	○	○	○
DISPLAY-OF			○		○	○	○		○	○	○
NATIONAL-OF			○		○	○	○		○	○	○
ACP-OF											○
ENUM-AND											○
ENUM-NOT											○
ENUM-OR											○
UNICODE-OF											○

E.9 データベース

データベース機能は、下表で○がついているシステムの固有の機能です。

DS	HP	Solaris	Win16	Win32	WinIPF	Winx64	Linux	LinuxIPF	Linux64	.NET

			○	○	○	○	○			○

また、データベース機能のストアドプロシージャは、下表で○がついているシステムの固有の機能です。

DS	HP	Solaris	Win16	Win32	WinIPF	Winx64	Linux	LinuxIPF	Linux64	.NET
				○	○	○	○			○

E.10 通信データベース

通信データベース機能としてのPowerAIMとの連携は、[Win16]の固有の機能です。

E.11 Micro Focus固有機能

Micro Focus固有機能の名前付き定数、16進数字定数およびCALL文は、下表で○がついているシステムの固有の機能です。

DS	HP	Solaris	Win16	Win32	WinIPF	Winx64	Linux	LinuxIPF	Linux64	.NET
		○		○	○	○	○	○	○	○

E.12 オブジェクト指向プログラミング機能

オブジェクト指向プログラミング機能は、下表で○がついているシステムの固有の機能です。

DS	HP	Solaris	Win16	Win32	WinIPF	Winx64	Linux	LinuxIPF	Linux64	.NET
		○		○	○	○	○	○	○	

E.13 .NETプログラミング機能

.NETプログラミング機能は、[.NET]の固有の機能です。

E.14 システムの定量制限

中核のデータ部におけるシステムの定量制限は、[Win16]とその他のシステムで一部異なります。

詳細は、“付録B システムの定量制限”の“B.2 中核のデータ部 ”を参照してください。

付録F　制御レコード

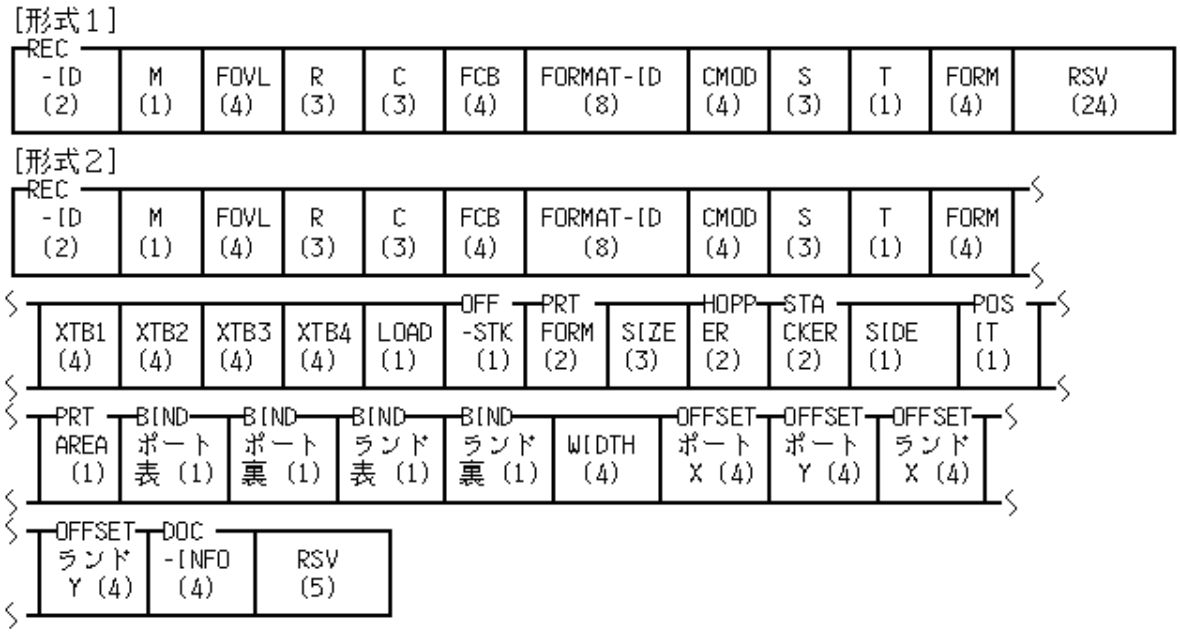
ここでは、I制御レコードおよびS制御レコードについて説明します。

なお、ここで説明した制御レコードの各領域には、システムによって使用できないものもあります。制御レコードの機能範囲については、“NetCOBOL 使用手引書”を参照してください。

F.1　I制御レコード

I制御レコードの形式を“図F.1 I制御レコードの形式”に示します。

図F.1 I制御レコードの形式



参考

各フィールドの括弧内は、文字数(バイト)を示します。

I制御レコードの各フィールドのデータ属性および意味を説明します。

REC-ID ： **制御レコードID**

- PICTURE X(2)
- 制御レコードの識別子であり、I制御レコードの場合“I1”を指定しなければなりません。

M ： **形式**

- PICTURE X
- “0”または“1”を指定します。“0”の場合、形式1のI制御レコードであることを示し、“1”の場合、形式2のI制御レコードであることを示します。

FOVL ： **フォームオーバレイのモジュール名**

- PICTURE X(4)

― 使用するフォームオーバレイのモジュール名を指定します。このフィールドが空白の場合、ファイルオープン時の指定が有効となります。
ファイルオープン時の指定とは、そのファイルがオープンされたときのシステムが持つ印刷環境のことです。

R ： フォームオーバレイの焼付け回数

― PICTURE 9(3)

― そのページに対する複写のうち、最初の何枚に対してFOVLで指定されたフォームオーバレイの焼付けを行うかを指定します。指定可能な値は、0～255です。また、複写数の値を超えてはなりません。FOVLフィールドが空白のときには、このフィールドは意味をもちません。

C ： 複写数

― PICTURE 9(3)

― ページ単位の複写数を指定します。指定可能な値は、0～255です。0を指定するとファイルオープン時の指定が有効となります。

FCB ： FCB名

― PICTURE X(4)

― そのページに対して適用するFCBの名前を指定します。

FORMAT-ID ： フォーマット定義体名

― PICTURE X(8)

― そのページに対して適用するフォーマット定義体の名前を指定します。この指定により固定形式ページとなります。このフィールドが空白の場合、不定形式ページとなります。FCB名と同時にフォーマット定義体名を指定してはいけません。

CMOD ： 複写修正モジュール名

― PICTURE X(4)

― そのページに対して適用する複写修正モジュールの名前を指定します。このフィールドが空白の場合、ファイルオープン時の指定が有効となります。

S ： 複写修正開始番号

― PICTURE 9(3)

― 複写修正モジュールのどのセグメントから使用して、複写データの修正を行うかその開始番号を指定します。指定可能な値は、1～255です。
CMODフィールドが空白のときには、このフィールドは意味をもちません。

T ： 複写修正用文字配列テーブル番号

― PICTURE 9

― 複写修正モジュールのデータを印刷するときの文字配列テーブル番号を指定します。指定可能な値は0～3です。
CMODフィールドが空白の場合、このフィールドは意味がなく、Tの値はファイルオープン時の指定が有効となります。

FORM ： 用紙識別名

― PICTURE X(4)

― 印刷用紙の変更が必要な場合に、利用者が任意に命名した印刷用紙の識別名を指定します。このフィールドが空白の場合はファイルオープン時に使用されていた用紙が指定されたものとみなされます。

XTB1～XTB4 ： 文字配列テーブルまたは追加文字セット

― PICTURE X(4)

― 文字配列テーブルまたは追加文字セットの識別名を指定します。このフィールドが空白の場合、ファイルオープン時の指定が有効となります。
FORMAT句の指定された印刷ファイルに対しては、文字配列テーブルは指定できません。

— LOAD ： ダイナミックロード
PICTURE X

— 印刷実行時に未収容文字を検出したとき、ダイナミックロードを行うことを指定します。“D”を指定するとダイナミックロードが行われます。このフィールドが空白の場合、ファイルオープン時の指定が有効となります。

— OFF-STK ： オフセットスタック
PICTURE X

— オフセットスタックを行うことを指定します。“O”を指定するとオフセットスタックが行われ、空白を指定するとオフセットスタックは行われません。両面印刷時に印刷面を裏面に位置付ける指定をしているときは、オフセットスタックを指定してはなりません。オフセットスタックとは、出力された用紙を任意の単位でスタッカにずらして排出することです。

PRT-FORM ： 印刷形式

— PICTURE X(2)

— 印刷形式を指定します。“P”(ポートレートモード)、“L”(ランドスケープモード)、“PZ”(縮小印刷のポートレートモード)、“LZ”(縮小印刷のランドスケープモード)、“LP”(ラインプリンタモード)のいずれかを指定します。このフィールドが空白のとき、SIZEに空白が指定されていなければならず、この場合、ファイルオープン時の指定が有効となります。

SIZE ： 用紙サイズ

— PICTURE X(3)

— 用紙サイズを指定します。指定可能なサイズは、“A3”、“A4”、“A5”、“B4”、“B5”、“LTR”の文字列です。また、[Win32][WinIPF][Winx64]の場合、任意の3文字以内の文字列も指定可能です。このフィールドが空白のとき、PRT-FORMに空白が指定されていなければならず、この場合、ファイルオープン時の指定が有効となります。また、任意の3文字以内の文字列を指定した場合、実行時に指定した文字列と実際に使用する用紙名を対応付けておかなければなりません。縮小印刷時は、用紙サイズ“A3”を指定してはなりません。ラインプリンタモード指定時は、用紙サイズ“A4”、“B4”、“LTR”を指定しなければなりません。SIZEとPRT-FORMの可能な組合せを“表F.1 用紙サイズと印刷形式の組合せ”に示します。

表F.1 用紙サイズと印刷形式の組合せ

PRT-FORM	SIZE					
	A3	A4	A5	B3	B5	LTR
L	可	可	可	可	可	可
P	可	可	可	可	可	可
LZ	不可	可	可	可	可	可
PZ	不可	可	可	可	可	可
LP	不可	可	不可	可	不可	可

HOPPER ： 用紙供給口

— PICTURE X(2)

— 用紙供給時の用紙供給口を指定します。

[Win32][WinIPF][Winx64]の場合、指定可能な値は、“P1”(主供給口1)、“P2”(主供給口2)、“S”(副供給口)です。“P”が指定された場合、任意の供給口から給紙します。このフィールドが空白の場合、ファイルオープン時の指定が有効となります。このフィールドを指定する場合、PRT-FORMおよびSIZEも指定しなければなりません。

[Solaris][Linux][LinuxIPF][Linux64]の場合、指定可能な値は、“P1”(主供給口1)、“P2”(主供給口2)、“P3”(主供給口3)、“P4”(主供給口4)、“S”(副供給口)です。“P”が指定された場合、任意の供給口から給紙します。このフィールドが空白の場合、ファイルオープン時の指定が有効となります。このフィールドを指定する場合、PRT-FORMおよびSIZEも指定しなければなりません。

STACKER ： 用紙排出口

— PICTURE X(2)

― 用紙出力時の用紙排出口を指定します。指定可能な値は、“P1”(主排出口1)、“P2”(主排出口2)、“S”(副排出口)です。また、“P”が指定された場合、任意の排出口へ出力します。このフィールドが空白の場合、ファイルオープン時の指定が有効となります。

SIDE ： 印刷面指定

― PICTURE X

― 印刷面を指定します。“F”を指定すると片面印刷であり、“B”を指定すると両面印刷となります。このフィールドが空白の場合、ファイルオープン時の指定が有効となります。

POSIT ： 印刷面位置付け

― PICTURE X

― 両面印刷時、印刷面を表面あるいは裏面に位置付けることを指定します。“F”を指定すると表面に位置付けられ、“B”を指定すると裏面に位置付けられます。このフィールドが空白の場合、ファイルオープン時の指定が有効となります。SIDEに空白が指定されている場合、このフィールドは空白でなければなりません。また、片面印刷指定時に印刷位置を裏面に位置付けてはなりません。

PRT-AREA ： 印字禁止領域

― PICTURE X

― 印字禁止領域の設定を指定します。“L”を指定すると印字禁止領域の設定を行い、“N”を指定すると設定を行いません。このフィールドが空白の場合、ファイルオープン時の指定が有効となります。

BIND ： とじしろ方向

― PICTURE X

― 連続して出力される複数ページを製本するときのとじしろ方向を指定します。製本の形式は、印刷形式と印刷面の組合せに対して四つの文字からなり、各フィールドは順にポートレートモード表面、ポートレートモード裏面、ランドスケープモード表面、ランドスケープモード裏面です。指定可能なとじしろ方向とその文字は、“L”(左とじ)、“R”(右とじ)、“U”(上とじ)、“D”(下とじ)です。とじしろ方向は複数ページに対して意味を持つ指定であるため、このフィールドが空白の場合、直前のページでのとじしろ方向がそのまま引き継がれます。以下に指定可能な組合せを示します。

```
ＬＬＵＵ、ＬＬＵＤ、ＬＬＤＵ、ＬＬＤＤ、
ＬＲＵＵ、ＬＲＵＤ、ＬＲＤＵ、ＬＲＤＤ、
ＲＬＵＵ、ＲＬＵＤ、ＲＬＤＵ、ＲＬＤＤ、
ＲＲＵＵ、ＲＲＵＤ、ＲＲＤＵ、ＲＲＤＤ、
ＵＵＬＬ、ＵＵＬＲ、ＵＵＲＬ、ＵＵＲＲ、
ＵＤＬＬ、ＵＤＬＲ、ＵＤＲＬ、ＵＤＲＲ、
ＤＵＬＬ、ＤＵＬＲ、ＤＵＲＬ、ＤＵＲＲ、
ＤＤＬＬ、ＤＤＬＲ、ＤＤＲＬ、ＤＤＲＲ
```

WIDTH ： とじしろ幅

― PICTURE 9(4)

― とじしろ幅を指定します。指定可能な値は0～9999で、単位は1/1440インチです。このフィールドに9999が指定されたとき、OFFSETのすべてのフィールドに、9999が指定されていなければなりません。また、このフィールドに空白が指定されたとき、OFFSETのすべてのフィールドに空白が指定されていなければなりません。これらの場合、ファイルオープン時の指定が有効となります。

OFFSET ： 印刷原点位置

― PICTURE 9(4)

― 印刷原点位置を指定します。指定可能な値は0～9999で、単位は1/1440インチです。印刷形式がポートレートモードだけまたはランドスケープモードだけであるとき、他方の印刷形式のフィールドには空白を指定します。OFFSETのすべてのフィールドに9999が指定されたとき、WIDTHに9999が指定されていなければなりません。また、OFFSETすべてのフィールドに空白が指定されたとき、WIDTHに空白が指定されていなければなりません。これらの場合、ファイルオープン時の指定が有効となります。

DOC-INFO ： 文書名識別情報

- PICTURE X(4)
- 文書名を識別するための文字列を指定します。指定可能な値は、任意の4文字以内の英数字です。このフィールドが空白のときは、ファイルオープン時の指定が有効となります。また、任意の4文字以内の英数字を指定した場合、実行時にこの文字列と表示する文書名を対応付けておかなければなりません。
  なお、DOC-INFOは[Win32][WinIPF][Winx64]固有機能です。

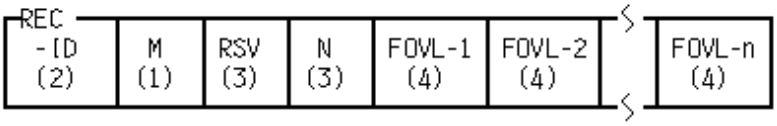
RSV ： システム使用

- [Win32][WinIPF][Winx64] PICTURE X(24) または X(5)
  [Solaris][Linux][LinuxIPF][Linux64] PICTURE X(24) または X(9)
- 空白を指定しなければなりません。

F.2 S制御レコード

S制御レコードの形式を、“図F.2 S制御レコードの形式”に示します。

図F.2 S制御レコードの形式



参考

各フィールドの括弧内は、文字数(バイト)を示します。

S制御レコードの各フィールドのデータ属性および意味を説明します。

REC-ID ： 制御レコードID

- PICTURE X(2)
- S制御レコードを示すため、“S1”を指定します。

M ： システム使用

- PICTURE X
- “0”でなければなりません。

RSV ： システム使用

- PICTURE X(3)
- 空白でなければなりません。

N ： オーバレイシーケンスの個数

- PICTURE 9(3)
- FOVL-nのフォームオーバレイモジュール名の個数を指定します。

FOVL-n ： フォームオーバレイシーケンスグループ内のモジュール名

- PICTURE X(4)
- nで指定された個数だけ、フォームオーバレイのモジュール名を指定します。I制御レコードのCで指定した数の複写に対して、この順序でフォームオーバレイの焼付けが行われます。なお、フォームオーバレイモジュール名お

よびフォームオーバレイ焼付け回数以外のページ属性(複写数、フォーマット定義体名など)は、S制御レコードの書出しによって変更されません。

付録G　型を使用したデータ項目の定義

一般的にCOBOLのデータ項目は、PICTURE句をはじめとする各種の句を指定することで属性を定義しますが、全く同じ属性を持つ複数のデータ項目を定義する場合、特定のデータ記述項をTYPEDEF句で型として宣言し、これを参照することでデータ項目を定義することもできます。

ここでは、型の宣言方法および使用方法について記述します。

なお、型は[Win32][WinIPF][Winx64][Solaris][Linux][LinuxIPF][Linux64][.NET]固有の機能です。

G.1　型

型は、レベル番号01のデータ記述項にTYPEDEF句を指定して宣言します。TYPEDEF句を指定したデータ記述項の項目名は型名とみなされ、型を参照する場合に使用されます。

型の宣言について、代表的な例を以下に示します。

G.1.1　基本項目の型

以下の例では、4桁(2バイト)および9桁(4バイト)の2進項目を、それぞれSHORTおよびLONGという型として宣言しています。在庫数および価格というデータ項目は、これらの型を参照して定義しています(以降、型を参照して定義したデータ項目を型付けされた項目と呼びます)。

例

[COBOLプログラム]

```
DATA           DIVISION.
 WORKING-STORAGE SECTION.
* 型の宣言
  01 SHORT TYPEDEF  PIC S9(4) COMP-5.
  01 LONG  TYPEDEF  PIC S9(9) COMP-5.
      :
* 型を参照してのデータ項目の定義
  01 在庫数          TYPE SHORT.
  01 価格            TYPE LONG.
      :
 PROCEDURE       DIVISION.
      :
     MOVE 0 TO  在庫数
     MOVE 0 TO  価格
```

型付けされた項目は、特に明示的な禁止がない限り、通常のデータ項目と同様にデータ部や手続き部で参照できます。

上の例の在庫数および価格は、以下のとおり定義した場合と同じです。

```
  01 在庫数       PIC S9(4) COMP-5.
  01 価格         PIC S9(9) COMP-5.
```

なお、型名SHORTおよびLONGは、それぞれの型宣言を識別するためのもので、特定の領域を持つものではありません。TYPE句で型を参照することによって、領域が割り当てられます。

型には、以下の句を指定することができます。

- BLANK WHEN ZERO句
- JUSTIFIED句
- OCCURS句
- PICTURE句

・ SIGN句

・ SYNCHRONIZED句

・ USAGE句

これらの句によって決定された属性は、型付けされたデータ項目を定義する際に保護の対象となります。したがって、型付けされた項目を従属する集団項目の属性が型に含まれる属性と矛盾する場合、エラーとなります。

```
 01 NUMDG4 TYPEDEF  PIC 9(4) COMP-5.
 01 NUMDG2 TYPEDEF  PIC 9(2) COMP-5.
     :
 01 年月日 USAGE DISPLAY.
    02 年            TYPE NUMDG4.     ←  エラー
    02 月            TYPE NUMDG2.     ←  エラー
    02 日            TYPE NUMDG2.     ←  エラー
```

G.1.2 集団項目の型

型は基本項目だけでなく、特定のレコード構造を持つ集団項目を定義することもできます。

以下の例は複数の従属項目を含む集団項目である在庫情報を1つの型として宣言し、この型を参照して倉庫A、倉庫Bおよび在庫レコードの3つのデータ項目を定義しています。

例

[COBOLプログラム]

```
 DATA          DIVISION.
 WORKING-STORAGE SECTION.
* 型の宣言
  01 在庫情報 TYPEDEF.
     02 製品コード     PIC 9(9) DISPLAY.
     02 製品名.
        03 綴り長      PIC S9(4) BINARY.
        03 綴り        PIC N(20).
     02 価格           PIC S9(9) BINARY.
     02 在庫数         PIC S9(4) BINARY.
* 型を参照してのデータ項目の定義
  01 倉庫Ａ           TYPE  在庫情報.
  01 倉庫Ｂ           TYPE  在庫情報.
  01 在庫レコード.
     02 倉庫番号       PIC 9(2).
     02 情報レコード   TYPE  在庫情報.
 PROCEDURE      DIVISION.
      :
     IF 在庫数 OF 倉庫Ａ ＞ 在庫数 OF 倉庫Ｂ THEN   … [1]
       MOVE 1       TO 倉庫番号
       MOVE 倉庫Ａ TO 情報レコード
     ELSE
       MOVE 2       TO 倉庫番号
       MOVE 倉庫Ｂ TO 情報レコード
     END-IF
```

型付けされた項目の参照は、参照する型が集団項目であっても基本項目として宣言されている場合と同様に参照することができます。ただし、型の従属項目位置への参照は、型を参照する集団項目名による修飾が必要となります(参照[1])。

集団項目として定義されている型を参照してデータ項目を定義する場合、この項目は型宣言のデータ記述項に従属する項目と同じ名前、記述、階層の従属項目を持つ集団項目となります(これを型付けされた集団項目と呼びます)。この際、型宣言のデータ記述項の従属項目の構造を完全に保証するために、レベル番号の調整とTYPE句を8バイト境界に位置付けるための遊びバイト(参照[2])の挿入が行われます。

この例の倉庫A、倉庫Bおよび在庫レコードは、次のとおり定義した場合と同じです。

```
 01 倉庫Ａ.
    02 製品コード     PIC S9(9) SIGN LEADING SEPARATE.
    02 製品名.
       03 綴り長      PIC S9(4) BINARY.
       03 綴り        PIC N(20).
    02 価格           PIC S9(9) BINARY.
    02 在庫数         PIC S9(4) BINARY.
 01 倉庫Ｂ.
    02 製品コード     PIC S9(9) SIGN LEADING SEPARATE.
    02 製品名.
       03 綴り長      PIC S9(4) BINARY.
       03 綴り        PIC N(20).
    02 価格           PIC S9(9) BINARY.
    02 在庫数         PIC S9(4) BINARY.
 01 在庫レコード.
    02 倉庫番号       PIC 9(2).
    02                PIC X(6).        遊びバイトの挿入　… [2]
    02 情報レコード.
       03 製品コード PIC S9(9) SIGN LEADING SEPARATE.
       03 製品名.
          04 綴り長  PIC S9(4) BINARY.
          04 綴り    PIC N(20).
       03 価格        PIC S9(9) BINARY.
       03 在庫数      PIC S9(4) BINARY.
```

G.2 強い型

COBOLのデータ項目は、項目の字類によって転記や比較の規則が決まります。その字類に許されない操作は翻訳時にエラーとなるため、実行することはできません。たとえば、以下の場合が該当します。

例

[COBOLプログラム]

```
DATA           DIVISION.
 WORKING-STORAGE SECTION.
* データ項目の定義
  01 数字項目         PIC S9(4).
      :
 PROCEDURE      DIVISION.
* 誤った操作
      :
     MOVE  "ABCD" TO  数字項目     … エラー
```

型は同じ構造を持つデータ定義の雛型にすぎません。このため、型付けされた項目であっても、通常のデータ項目と同じように扱うことができます。

型付けされた集団項目の字類は、通常の集団項目と同様に英数字となります。このため、以下の例では、集団項目転記によって従属項目である“綴り長”に不当な値が設定されます。

例

[COBOLプログラム]

```
DATA           DIVISION.
 WORKING-STORAGE SECTION.
* 型の宣言
  01 名前型データ   TYPEDEF.
```

```
     02 綴り長       PIC S9(4) BINARY.
     02 綴り         PIC N(20).
* データ項目の定義
  01 製品名          TYPE 名前型データ.
      :
 PROCEDURE       DIVISION.
      :
     MOVE SPACE  TO 製品名              … [3]
```

しかし、集団項目が強い型を参照して定義されている(強く型付けされた集団項目)場合、このような操作は禁止されます。強く型付けされた集団項目の字類は、英数字ではなく指定された型名となるため、”名前型データ”が強い型であるなら上の例の[3]の操作は翻訳時にエラーとなります。

強い型は、型宣言でTYPEDEF句にSTRONGを指定することで宣言します。ただし、STRONGは集団項目として定義した型にしか指定できません。また、[.NET]では、STRONGは指定できません。

例

[COBOLプログラム]

```
DATA           DIVISION.
 WORKING-STORAGE SECTION.
* 型の宣言
  01 名前型データ   TYPEDEF STRONG.
     02 綴り長       PIC S9(4) BINARY.
     02 綴り         PIC N(20).
```

強く型付けされた項目に固有の操作

強く型付けされた集団項目は、通常の型付けされた集団項目と以下の点で異なります。

- 固有の字類を持つ。
- 従属するすべての基本項目の持つ値の正当性が保証される。
- 参照する型と型に含まれるすべての従属項目の属性を保持する。

このような違いから、強い型の宣言や強く型付けされた集団項目の参照には、いくつかの制限があります。たとえば、強く型付けされた集団項目に集団項目転記で値を設定する場合、転記の送出し側は同等の型を参照する強く型付けされた集団項目でなければなりません。

例

[COBOLプログラム]

```
DATA           DIVISION.
 WORKING-STORAGE SECTION.
* 型の宣言
  01 名前型データ   TYPEDEF STRONG.
     02 綴り長       PIC S9(4) BINARY.
     02 綴り         PIC N(20).
* データ項目の定義
  01 名前Ａ          TYPE 名前型データ.
  01 名前Ｂ          TYPE 名前型データ.
  01 名前Ｃ.
     02 綴り長       PIC S9(4) BINARY.
     02 綴り         PIC N(20).
 PROCEDURE DIVISION.
      :
     MOVE  名前Ａ  TO  名前Ｂ
```

```
    MOVE  名前Ａ  TO  名前Ｃ
    MOVE  名前Ｃ  TO  名前Ａ     …[4]
```

上の例では、通常の集団項目から強く型付けされた集団項目への転記[4]で翻訳エラーとなります。

強く型付けされた集団項目は以下の機能で使用できます。

- 表示
- 転記
- 比較
- メソッドの仮パラメタまたは実パラメタへの指定

一方、強く型付けされた項目でなければできない操作もあります。

通常、オブジェクト参照項目を含む集団項目は、集団項目としての転記、比較などを行うことができません。しかし、オブジェクト参照項目を含む集団項目を強い型として宣言した場合、強く型付けされた集団項目は、参照する型に含まれるすべての従属項目の属性情報を保持するため、これらが可能となります。

例

[COBOLプログラム]

```
      :
 REPOSITORY.
     CLASS  契約.
 DATA           DIVISION.
 WORKING-STORAGE SECTION.
* 強い型の宣言
  01 契約情報  TYPEDEF STRONG.
     02 契約実数          PIC S9(4) BINARY.
     02 契約オブジェクト OBJECT REFERENCE 契約 OCCURS 9999.
      :
* 強く型付けされた項目の定義
  01 当日分契約     TYPE 契約情報.
  01 昨日分契約     TYPE 契約情報.
      :
 PROCEDURE DIVISION.
      :
     MOVE  当日分契約  TO  昨日分契約
```

オブジェクト参照項目を含む集団項目は、強く型付けされた項目として定義された場合だけ、以下の機能で使用できます。

- 転記
- 比較
- メソッドの仮パラメタまたは実パラメタへの指定

また、メソッドの仮パラメタまたは実パラメタに集団項目を指定する場合、通常の集団項目は同じ長さの英数字項目とみなして適合チェックが行われますが、強く型付けされた集団項目の場合、より厳密に適合チェックが行われます。

メソッドの仮パラメタまたは実パラメタに強く型付けされた集団項目を指定する場合、仮パラメタに指定した項目と実パラメタに指定した項目が同等の型を参照するものでなければなりません。

G.3 他の型を参照する型

型は通常のデータ項目の定義にだけでなく、型宣言のためのデータ記述項でも参照することができます。したがって、先に示した在庫情報型は次のように定義することもできます。

例

[COBOLプログラム]

```
 DATA           DIVISION.
 WORKING-STORAGE SECTION.
* 型の宣言
  01 SHORT TYPEDEF   PIC S9(4) COMP-5.
  01 LONG  TYPEDEF   PIC S9(9) COMP-5.
  01 在庫情報 TYPEDEF.
     02 製品コード   PIC 9(9) DISPLAY.
     02 製品名.
        03 綴り長    TYPE SHORT.
        03 綴り      PIC N(20).
     02 価格         TYPE LONG.
     02 在庫数       TYPE SHORT.
      :
* 型を参照してのデータ項目の定義
  01 倉庫Ａ          TYPE  在庫情報.
  01 倉庫Ｂ          TYPE  在庫情報.
  01 在庫レコード.
     02 倉庫番号     PIC 9(2).
     02 情報レコード TYPE  在庫情報.
      :
 PROCEDURE      DIVISION.
      :
     IF 在庫数 OF 倉庫Ａ ＞ 在庫数 OF 倉庫Ｂ THEN
       MOVE 1      TO 倉庫番号
       MOVE 倉庫Ａ TO 情報レコード
     ELSE
       MOVE 2      TO 倉庫番号
       MOVE 倉庫Ｂ TO 情報レコード
     END-IF
```

この例における倉庫A、倉庫Bおよび在庫レコードは、先に“集団項目の型”で示した型を使用しない記述例と同じ結果を与えます。

なお、型宣言のためのデータ記述項で参照する型が強い型である場合は、以下の注意が必要です。

- 強く型付けされた項目を含む型宣言は、強い型を宣言するものでなければなりません。
- 型宣言に含まれる強く型付けされた項目の参照する型の情報も、新しく宣言される型に含まれる従属項目の情報として保存されます。

例

[COBOLプログラム]

```
DATA           DIVISION.
 WORKING-STORAGE SECTION.
* 型の宣言
  01 SHORT TYPEDEF  PIC S9(4) COMP-5.
  01 LONG  TYPEDEF  PIC S9(9) COMP-5.
  01 名前型データ   TYPEDEF STRONG.
     02 綴り長      PIC S9(4) BINARY.
     02 綴り        PIC N(20).
  01 在庫情報       TYPEDEF STRONG.
     02 製品コード  PIC 9(9) DISPLAY.
     02 製品名      TYPE 名前型データ.
     02 価格        TYPE LONG.
     02 在庫数      TYPE SHORT.
      :
```

```
* 型を参照してのデータ項目の定義
  01 倉庫Ａ            TYPE  在庫情報.
  01 入庫品            TYPE  在庫情報.
      :
 PROCEDURE      DIVISION.
      :
     MOVE  製品名 OF 入庫品  TO  製品名 OF 倉庫Ａ     … [5]
```

上の例では、倉庫Aの従属項目である製品名は、在庫情報型の従属項目であると同時に、名前型データ型で強く型付けされた項目でもあります。このため、名前型データ型で強く型付けされたデータ項目である入庫品の内容を設定することが可能です。

G.4 型を使用したデータ定義の有効な使い方

次のようなプログラムを記述する場合、あらかじめ定義しておいた型を参照してデータ項目を定義すると便利です。

- 共通のデータ構造を持つデータ項目を参照するプログラム
- 他言語と連携するプログラム

参照

他言語のデータ型に対応する型を定義する場合は、“NetCOBOL 使用手引書”を参考にしてください。

付録H　サンプル集

いくつかの句や文について、それらの使い方が理解できるようにミニサンプルを用意しました。各サンプルは50ステップ前後に抑えていますので、比較的容易に理解できると思います。また、「この機能は、このように利用すると便利」といったコーディングテクニックの紹介にもなっていますので、有効に活用してください。

H.1　連結式

連結式は、文字定数と文字定数を連結する際に利用します。

16進文字定数や記号定数などと混在して連結できることから、制御文字を付加したい場合などに利用すると便利です。

このサンプルはWindows関数を呼び出しているため、リンク時にCOBOLインストールフォルダ（ディレクトリ）に格納されている”USER32.LIB”をリンクする必要があります。注意してください。

なお、このサンプルは[Win32]で動作します。[Win32]以外でもCと連携する場合などに同様な処理が必要となりますので、参考にしてください。

例

```
000010 @OPTIONS MAIN,ALPHAL(WORD)
000020*---------------------------------------------------------------
000030* 連結式により、文字定数を連結することができます。
000040*---------------------------------------------------------------
000050 IDENTIFICATION   DIVISION.
000060 PROGRAM-ID.      SAMPLE.
000070 DATA             DIVISION.
000080  WORKING-STORAGE SECTION.
000090  01 TITLE-TXT    PIC X(20).
000100  01 MSG-TXT      PIC X(20).
000110  01 RET          PIC S9(9) COMP-5 VALUE 0.
000120 PROCEDURE        DIVISION.
000130*---------------------------------------------------------------
000140* 文字定数を "&" により連結することができます。
000150* 以下の例のように、Cと連携する際のnull文字設定などに便利です。
000160* →C言語に文字列データを渡す場合、文字列の終端にnullが必要です。
000170*   部分参照などで設定することもできますが、連結式を利用すると、
000180*   簡単で、かつ、きれいにコーディングできます。
000190*---------------------------------------------------------------
000200     MOVE "SAMPLE"        & X"00" TO TITLE-TXT.
000210     MOVE "Hello COBOL!!" & X"00" TO MSG-TXT.
000220*---------------------------------------------------------------
000230     CALL "MessageBoxA" WITH STDCALL USING BY VALUE     0
000240                                           BY REFERENCE MSG-TXT
000250                                           BY REFERENCE TITLE-TXT
000260                                           BY VALUE     1
000270                                      RETURNING RET.
000280     EXIT PROGRAM.
000290 END PROGRAM SAMPLE.
```

H.2　修飾

COBOLでは、集団項目に従属するデータ名や段落名などで同名の利用者語を使用することができます。たとえば、日付情報のように、ある決まった型を持つ情報を複数の項目で使用する場合、データ名が統一されていると読みやすいコーディングが可能になります。

データ名が重複する場合、データ名を一意にするために用いられるのが修飾です。OFまたはINを利用して、そのデータが属する集団名を明に指定します。

例

```
000010 @OPTIONS MAIN
000020*---------------------------------------------------------------
000030* 一意修飾により、同名のデータを使い分けることができます。
000040*---------------------------------------------------------------
000050 IDENTIFICATION   DIVISION.
000060 PROGRAM-ID.      SAMPLE.
000070 DATA             DIVISION.
000080  WORKING-STORAGE SECTION.
000090*---------------------------------------------------------------
000100* 同じ意味を持つデータには同じデータ名を与えてもかまいません。
000110*---------------------------------------------------------------
000120  01 BIRTHDAY.
000130    02 YYYY         PIC 9(4).
000140    02 MMDD         PIC 9(4).
000150  01 TODAY.
000160    02 YYYY         PIC 9(4).
000170    02 MMDD         PIC 9(4).
000180*---------------------------------------------------------------
000190  01 AGE          PIC ZZ9.
000200 PROCEDURE        DIVISION.
000210     DISPLAY "あなたの誕生日は？ 例: 19690123 >> " WITH NO ADVANCING.
000220     ACCEPT BIRTHDAY.
000230     MOVE FUNCTION CURRENT-DATE TO TODAY.
000240*---------------------------------------------------------------
000250* データ参照時に一意となるよう、OFで修飾します。
000260*---------------------------------------------------------------
000270     IF MMDD OF BIRTHDAY <= MMDD OF TODAY THEN
000280       COMPUTE AGE = YYYY OF TODAY - YYYY OF BIRTHDAY
000290     ELSE
000300       COMPUTE AGE = YYYY OF TODAY - YYYY OF BIRTHDAY - 1
000310     END-IF.
000320*---------------------------------------------------------------
000330     DISPLAY "あなたの年齢は、" AGE " です。".
000340     EXIT PROGRAM.
000350 END PROGRAM SAMPLE.
```

H.3 添字付け

同じデータ構造の繰返しは、OCCURS句を利用してデータ定義し、添字付けによって参照します。

添字付けの際に注意していただきたいのが、添字範囲外の設定および参照です。実際の繰返し数を超えた添字付けは、領域破壊や誤った領域の参照による実行時異常終了の原因になります。

例

```
000010 @OPTIONS MAIN
000020*---------------------------------------------------------------
000030* 配列宣言されたデータは添字参照によって参照できます。
000040*---------------------------------------------------------------
000050 IDENTIFICATION   DIVISION.
000060 PROGRAM-ID.      SAMPLE.
000070 DATA             DIVISION.
000080  WORKING-STORAGE SECTION.
000090  01 ALP          PIC X(26) VALUE "ABCDEFGHIJKLMNOPQRSTUVWXYZ".
000100  01             REDEFINES ALP.
000110    02 CHAR         OCCURS 26 TIMES PIC X.
000120  01 IN-DATA      PIC X.
000130  01 COUNTER      PIC 9(2).
```

```
000140 PROCEDURE        DIVISION.
000150     DISPLAY "英大文字を1字、入力してください。 >> " WITH NO ADVANCING.
000160     ACCEPT  IN-DATA.
000170     PERFORM TEST BEFORE
000180             VARYING COUNTER FROM 1 BY 1 UNTIL COUNTER > 26
000190*-----------------------------------------------------------------
000200* COUNTERを添字にして、入力された文字と比較します。
000220*-----------------------------------------------------------------
000230       IF IN-DATA = CHAR(COUNTER) THEN
000240*-----------------------------------------------------------------
000250         EXIT PERFORM
000260       END-IF
000270     END-PERFORM.
000280     IF COUNTER <= 26 THEN
000290       DISPLAY IN-DATA "は、アルファベットで" COUNTER "番目の文字です。"
000300     ELSE
000310       DISPLAY "入力文字に誤りがあります。"
000320     END-IF.
000330 END PROGRAM SAMPLE.
```

一意修飾と添字参照を組み合わせることもできます。

例

```
000010 @OPTIONS MAIN
000020*-----------------------------------------------------------------
000030* 配列宣言されたデータを一意修飾します。
000040*-----------------------------------------------------------------
000050 IDENTIFICATION   DIVISION.
000060 PROGRAM-ID.      SAMPLE.
000070 DATA             DIVISION.
000080  WORKING-STORAGE SECTION.
000090  01 UPPER-CASE.
000100    02 ALP        PIC X(26) VALUE "ABCDEFGHIJKLMNOPQRSTUVWXYZ".
000110    02           REDEFINES ALP.
000120      03 CHAR       OCCURS 26 TIMES PIC X.
000130  01 LOWER-CASE.
000140    02 ALP        PIC X(26) VALUE "abcdefghijklmnopqrstuvwxyz".
000150    02           REDEFINES ALP.
000160      03 CHAR       OCCURS 26 TIMES PIC X.
000170  01 IN-DATA      PIC X.
000180  01 COUNTER      PIC 9(2).
000190 PROCEDURE        DIVISION.
000200     DISPLAY "英大文字を1字、入力してください。 >> " WITH NO ADVANCING.
000210     ACCEPT  IN-DATA.
000220     PERFORM TEST BEFORE
000230             VARYING COUNTER FROM 1 BY 1 UNTIL COUNTER > 26
000240*-----------------------------------------------------------------
000250* COUNTERを添字にして、入力された文字と比較します。
000260* 一意修飾されたデータを添字参照する場合は，添字を修飾の後ろに
000270* 指定します。
000280*-----------------------------------------------------------------
000290       IF IN-DATA = CHAR OF UPPER-CASE (COUNTER) THEN
000300*-----------------------------------------------------------------
000310         EXIT PERFORM
000320       END-IF
000330     END-PERFORM.
000340     IF COUNTER <= 26 THEN
000350       DISPLAY IN-DATA "に対応する英小文字は"
000360               CHAR OF LOWER-CASE (COUNTER) "です。"
000370     ELSE
```

```
000380       DISPLAY "入力文字に誤りがあります。"
000390     END-IF.
000400 END PROGRAM SAMPLE.
```

H.4 部分参照

文字データの一部分を取り出したい場合は、部分参照を利用します。

部分参照の際に注意していただきたいのが、範囲外の設定および参照です。実際の領域長を超えた部分参照は、領域破壊や誤った領域の参照による実行時異常終了の原因になります。

例

```
000010 @OPTIONS MAIN
000020*---------------------------------------------------------------
000030* 文字列データを部分的に切り出す場合は部分参照で実現します。
000040*---------------------------------------------------------------
000050 IDENTIFICATION   DIVISION.
000060 PROGRAM-ID.      SAMPLE.
000070 DATA             DIVISION.
000080  WORKING-STORAGE SECTION.
000090  01 ALP          PIC X(26) VALUE "ABCDEFGHIJKLMNOPQRSTUVWXYZ".
000100  01 IN-DATA      PIC X.
000110  01 COUNTER      PIC 9(2).
000120 PROCEDURE        DIVISION.
000130     DISPLAY "英大文字を1字、入力してください。>> " WITH NO ADVANCING.
000140     ACCEPT  IN-DATA.
000150     PERFORM TEST BEFORE
000160             VARYING COUNTER FROM 1 BY 1 UNTIL COUNTER > 26
000170*---------------------------------------------------------------
000180* 部分参照は、(開始位置:長さ)で文字データを切り出すことができます。
000190*---------------------------------------------------------------
000200       IF IN-DATA = ALP(COUNTER:1) THEN
000210*---------------------------------------------------------------
000220         EXIT PERFORM
000230       END-IF
000240     END-PERFORM.
000250     IF COUNTER <= 26 THEN
000260       DISPLAY IN-DATA "は、アルファベットで" COUNTER "番目の文字です。"
000270     ELSE
000280       DISPLAY "入力文字に誤りがあります。"
000290     END-IF.
000300 END PROGRAM SAMPLE.
```

一意修飾と部分参照を組み合わせることもできます。

例

```
000010 @OPTIONS MAIN
000020*---------------------------------------------------------------
000030* 一意修飾が必要なデータ名を部分参照することもできます。
000040*---------------------------------------------------------------
000050 IDENTIFICATION   DIVISION.
000060 PROGRAM-ID.      SAMPLE.
000070 DATA             DIVISION.
000080  WORKING-STORAGE SECTION.
000090  01 UPPER-CASE.
000100    02 ALP        PIC X(26) VALUE "ABCDEFGHIJKLMNOPQRSTUVWXYZ".
000110  01 LOWER-CASE.
```

```
000120    02 ALP        PIC X(26) VALUE "abcdefghijklmnopqrstuvwxyz".
000130  01 IN-DATA      PIC X.
000140  01 COUNTER      PIC 9(2).
000150 PROCEDURE        DIVISION.
000160     DISPLAY "英大文字を1字、入力してください。 >> " WITH NO ADVANCING.
000170     ACCEPT  IN-DATA.
000180     PERFORM TEST BEFORE
000190             VARYING COUNTER FROM 1 BY 1 UNTIL COUNTER > 26
000200*-----------------------------------------------------------------
000210* COUNTERを添字にして、入力された文字と比較します。
000220* 一意修飾されたデータを部分参照する場合は，部分参照子を修飾の
000230* 後ろに指定します。
000240*-----------------------------------------------------------------
000250       IF IN-DATA = ALP OF UPPER-CASE (COUNTER:1) THEN
000260*-----------------------------------------------------------------
000270         EXIT PERFORM
000280       END-IF
000290     END-PERFORM.
000300     IF COUNTER <= 26 THEN
000310       DISPLAY IN-DATA "に対応する英小文字は"
000320               ALP OF LOWER-CASE (COUNTER:1) "です。"
000330     ELSE
000340       DISPLAY "入力文字に誤りがあります。"
000350     END-IF.
000360 END PROGRAM SAMPLE.
```

H.5 ポインタ付け

あるアドレスをベースにデータ項目を参照したい場合、ポインタ付けを使用します。その際、データの属性やオフセットは基底場所節で定義します。

C言語連携などでアドレスがインタフェースとなるような場合によく利用します。

例

```
000010 @OPTIONS MAIN
000020*-----------------------------------------------------------------
000030* ポインタ付けにより、任意の枠でデータ参照が可能になります。
000040*-----------------------------------------------------------------
000050 IDENTIFICATION   DIVISION.
000060 PROGRAM-ID.      SAMPLE.
000070 DATA             DIVISION.
000080*-----------------------------------------------------------------
000090* 基底場所節で、任意の型(データ参照時の枠のようなもの)を定義します。
000100*-----------------------------------------------------------------
000110  BASED-STORAGE   SECTION.
000120  01 TYPE-DATE.
000130    02              PIC X(8).
000140    02 CR-HOUR      PIC 9(2).
000150    02 CR-MINUTE    PIC 9(2).
000160    02 CR-SEC       PIC 9(2).
000170*-----------------------------------------------------------------
000180  WORKING-STORAGE SECTION.
000190  01 CR-DATE       PIC X(21).
000200  01 PTR           POINTER.
000210 PROCEDURE         DIVISION.
000220     MOVE FUNCTION CURRENT-DATE TO CR-DATE.
000230*-----------------------------------------------------------------
000240* 枠の先頭アドレスをポインタ項目に設定し、任意のデータを枠に
000250* はめて(ポインタ付けして)参照します。
000260*-----------------------------------------------------------------
```

```
000270     MOVE FUNCTION ADDR(CR-DATE) TO PTR.
000280     DISPLAY "只今の時刻は、" PTR->CR-HOUR   "時"
000290                              PTR->CR-MINUTE "分"
000300                              PTR->CR-SEC    "秒です。".
000310*-----------------------------------------------------------------
000320 END PROGRAM SAMPLE.
```

H.6 条件名の一意参照

条件名も修飾することができます。

ある汎用的な状態に条件名を与えた場合、修飾によって一意参照できます。

例

```
000010 @OPTIONS MAIN
000020*-----------------------------------------------------------------
000030* 条件名を修飾することもできます。
000040*-----------------------------------------------------------------
000050 IDENTIFICATION   DIVISION.
000060 PROGRAM-ID.      SAMPLE.
000070 DATA             DIVISION.
000080  WORKING-STORAGE SECTION.
000090*-----------------------------------------------------------------
000100* 月、日の正しい範囲内をいずれも CORRECT で定義します。
000110*-----------------------------------------------------------------
000120  01 BONUS.
000130    02 B-MONTH      PIC 9(2).
000140      88 CORRECT      VALUE 1 THRU 12.
000150    02 B-DAY        PIC 9(2).
000160      88 CORRECT      VALUE 1 THRU 31.
000170*-----------------------------------------------------------------
000180  01 CR-DATE.
000190    02              PIC 9(4).
000200    02 CR-MONTH     PIC 9(2).
000210    02 CR-DAY       PIC 9(2).
000220  01 ANS           PIC 9(3).
000230 PROCEDURE         DIVISION.
000240     DISPLAY "ボーナス支給日は？ 例:1210 >> " WITH NO ADVANCING.
000250     ACCEPT BONUS.
000260     MOVE FUNCTION CURRENT-DATE TO CR-DATE.
000270*-----------------------------------------------------------------
000280* 条件名を OF で修飾して一意参照します。
000290*-----------------------------------------------------------------
000300     IF CORRECT OF B-MONTH AND
000310        CORRECT OF B-DAY   THEN
000320*-----------------------------------------------------------------
000330       IF B-MONTH = CR-MONTH AND B-DAY >= CR-DAY THEN
000340         COMPUTE ANS = B-DAY - CR-DAY
000350       ELSE
000360         COMPUTE ANS = 30 - CR-DAY
000370         COMPUTE CR-MONTH = CR-MONTH + 1
000380         IF CR-MONTH > 12 THEN
000390           COMPUTE CR-MONTH = CR-MONTH - 12
000400         END-IF
000410         IF B-MONTH >= CR-MONTH THEN
000420           COMPUTE ANS = ANS + ((B-MONTH - CR-MONTH) * 30) + B-DAY
000430         ELSE
000440           COMPUTE ANS = ANS + ((12 - CR-MONTH) * 30) + (B-MONTH * 30) + B-DAY
000450         END-IF
000460       END-IF
```

```
000470       DISPLAY "ボーナス支給日まで、残り約 " ANS "日あります。"
000480     ELSE
000490       DISPLAY "入力データに誤りがあります。"
000500     END-IF.
000510 END PROGRAM SAMPLE.
```

H.7 行のつなぎ

語の並びが1行に収まらない場合や、複数行に分割した方が読みやすい場合、継続行によって次行へ継続することができます。

例

```
000010 @OPTIONS MAIN
000020*---------------------------------------------------------------
000030* 継続行を利用して、複数行に分けてコーディングすることができます。
000040*
000050* 継続行の仕様は、正書法によっても変わります。ここでは、最も多く
000060* 利用される可変長形式の継続について説明します。
000070*---------------------------------------------------------------
000080 IDENTIFICATION   DIVISION.
000090 PROGRAM-ID.      SAMPLE.
000100 DATA             DIVISION.
000110  WORKING-STORAGE SECTION.
000120*---------------------------------------------------------------
000130* 定数中で継続する場合、
000140* 継続される行(170行)は引用符で閉じずに改行します。
000150* 継続する行(180行)は標識領域に "-" を指定し、引用符から続けます。
000160*---------------------------------------------------------------
000170  01 COL-LINE     PIC X(60) VALUE "----+----1----+----2
000180-                               "----+----3----+----4
000190-                               "----+----5----+----6".
000200*---------------------------------------------------------------
000210  01 STR-DATA     PIC X(60).
000220 PROCEDURE        DIVISION.
000230*---------------------------------------------------------------
000240* 予約語や利用者語も継続できます。
000250*---------------------------------------------------------------
000260     DISPLAY "入力した文字列をカラム下に表示します >> " WITH NO AD
000270-            VANCING.
000280     ACCEPT  STR-D
000290-                ATA FROM CONSOLE.
000300*---------------------------------------------------------------
000310     DISPLAY " ".
000320     DISPLAY COL-LINE.
000330     DISPLAY STR-DATA.
000340 END PROGRAM SAMPLE.
```

H.8 自由形式の正書法

自由形式を採ることにより、行番号や標識領域、A領域、B領域などを気にすることなく、コーディングできます。

なお、自由形式のソースを翻訳する場合、必ず翻訳オプションSRF(FREE)を指定する必要があります。注意してください。

このサンプルは[Win32][WinIPF][Winx64][Solaris][Linux][LinuxIPF][Linux64][.NET]で動作します。

例

```
       @OPTIONS MAIN,SRF(FREE)
*>-----------------------------------------------------------------
*> 自由形式の記述例です。
*>
*> 自由形式のソースを翻訳する場合は、翻訳オプション"SRF(FREE)"を
*> 指定してください。
*>-----------------------------------------------------------------
IDENTIFICATION   DIVISION.
PROGRAM-ID.      SAMPLE.
DATA             DIVISION.
 WORKING-STORAGE SECTION.
 77 MSG          PIC X(60).
PROCEDURE        DIVISION.
*>-----------------------------------------------------------------
*> コメントには注記("*>")を利用します。
*> 注記は、任意のカラム位置に記述することができます。
*>-----------------------------------------------------------------
  DISPLAY " ".     *> A領域やB領域を気にする必要もありません。
*>-----------------------------------------------------------------
*> 空白行も記述できます。
*>-----------------------------------------------------------------

*>-----------------------------------------------------------------
*> 文字定数の継続も可能です。
*> - 可変長形式の場合と異なり、引用符で一度閉じます。
*> - 閉じた引用符の直後に "-" を記述します。
*> - "-" の後ろには、1個以上の空白を記述して改行します。
*> - 後続の行は、引用符から開始します。
*>-----------------------------------------------------------------
  MOVE "Hello, I'm a"-
       " NetCOBOL. Ha"-
       "ve a good job !!" TO MSG.
  DISPLAY MSG.
END PROGRAM SAMPLE.
```

H.9 PROGRAM COLLATING SEQUENCE句

プログラム内の文字順序(通常は、ASCIIの文字順序に従います)を変更したい場合、PROGRAM COLLATING SEQUENCE句を利用します。

例

```
000010 @OPTIONS MAIN
000020*-----------------------------------------------------------------
000030* PROGRAM COLLATING SEQUENCE句を利用して、文字をEBCDICコードの
000040* 大小順序で比較します。
000050*-----------------------------------------------------------------
000060 IDENTIFICATION   DIVISION.
000070 PROGRAM-ID.      SAMPLE.
000080 ENVIRONMENT      DIVISION.
000090  CONFIGURATION   SECTION.
000100*-----------------------------------------------------------------
000110* ALPHABET句でEBCDICに対して符号系名(EBCDIC-CODE)を宣言し、
000120* PROGRAM COLLATING SEQUENCE句に符号系名を指定します。
000130*-----------------------------------------------------------------
000140  OBJECT-COMPUTER. FM-V
000150          PROGRAM COLLATING SEQUENCE IS EBCDIC-CODE.
```

```
000160  SPECIAL-NAMES.
000170          ALPHABET EBCDIC-CODE IS EBCDIC.
000180*-----------------------------------------------------------------
000190 DATA              DIVISION.
000200  WORKING-STORAGE SECTION.
000210  01 DATA-1 PIC X(3).
000220  01 DATA-2 PIC X(3).
000230**
000240 PROCEDURE         DIVISION.
000250        DISPLAY "PLEASE INPUT 3 CHARACTERS (DATA-1) >> ".
000260        ACCEPT DATA-1.
000270        DISPLAY "PLEASE INPUT 3 CHARACTERS (DATA-2) >> ".
000280        ACCEPT DATA-2.
000290**
000300        DISPLAY " ".
000310        DISPLAY "***  RESULT OF RELATION INPUT-DATA  ***".
000320        EVALUATE TRUE
000330        WHEN DATA-1 = DATA-2
000340          DISPLAY DATA-1 "(DATA-1) = " DATA-2 "(DATA-2)"
000350        WHEN DATA-1 < DATA-2
000360          DISPLAY DATA-1 "(DATA-1) < " DATA-2 "(DATA-2)"
000370        WHEN DATA-1 > DATA-2
000380          DISPLAY DATA-1 "(DATA-1) > " DATA-2 "(DATA-2)"
000390        END-EVALUATE.
000400 END PROGRAM SAMPLE.
```

H.10 機能名-1句

入出力操作に関連する機能名に対応する呼び名を定義します。

ここで定義した呼び名は、以下のいずれかで指定します。

- ACCEPT文のFROM指定または、DISPLAY文のUPON指定
- WRITE文のADVANCING指定
- CHARACTER TYPE句の呼び名指定

例

```
000010 @OPTIONS MAIN
000020*-----------------------------------------------------------------
000030* CHARACTER TYPE句の呼び名指定により、印字属性を制御します。
000040*-----------------------------------------------------------------
000050 IDENTIFICATION   DIVISION.
000060 PROGRAM-ID.      SAMPLE.
000070 ENVIRONMENT      DIVISION.
000080  CONFIGURATION   SECTION.
000090*-----------------------------------------------------------------
000100* 印字属性に対する呼び名を定義します。
000110* YD-9P-12  : Y(横書き) D-9P(9ポの文字を6CPIで) 12(長体文字)
000120* GTX-12P-21: G(ゴシック体) T(縦書き) X-12P(12ポの文字) 21(平体文字)
000130*
000140* 詳細については、文法書5.4.2 CHARACTER TYPE句を参照してください。
000150*-----------------------------------------------------------------
000160  SPECIAL-NAMES.
000170          YD-9P-12   IS ATTR-1
000180          GTX-12P-21 IS ATTR-2.
000190*-----------------------------------------------------------------
000200  INPUT-OUTPUT    SECTION.
000210  FILE-CONTROL.
000220          SELECT PRT-FILE ASSIGN TO PRINTER.
```

```
000230 DATA             DIVISION.
000240  FILE            SECTION.
000250  FD PRT-FILE.
000260  01 PRT-REC      PIC N(10).
000270  WORKING-STORAGE SECTION.
000280*-----------------------------------------------------------------
000290* 機能名-1で定義した印字属性を持つ項目を宣言します。
000300*-----------------------------------------------------------------
000310  01 DATA-1        PIC N(7) CHARACTER TYPE ATTR-1.
000320  01 DATA-2        PIC N(7) CHARACTER TYPE ATTR-2.
000330*-----------------------------------------------------------------
000340 PROCEDURE        DIVISION.
000350       DISPLAY "Now printing .....".
000360       OPEN OUTPUT PRT-FILE.
000370         MOVE NC"富士通株式会社" TO DATA-1 DATA-2.
000380         WRITE PRT-REC FROM DATA-1 AFTER ADVANCING PAGE.
000390         WRITE PRT-REC FROM DATA-2 AFTER ADVANCING 2 LINES.
000400       CLOSE PRT-FILE.
000410 END PROGRAM SAMPLE.
```

H.11 機能名-2句

機能名-2により、外部スイッチを利用することができます。

外部スイッチは実行時に切り換えることができる情報です。

- [Win32][WinIPF][Winx64][.NET]ではアプリケーション起動時に実行時オプション“@GOPT”で指定します。
- [Solaris][Linux][LinuxIPF][Linux64]では環境変数“GOPT”で指定します。
- [Win32][WinIPF][Winx64][Solaris][Linux][LinuxIPF][Linux64][.NET]以外でも、実行時オプション“-CBL”で指定することができます。

以下のサンプルの場合、実行時オプションに“s10000000”が指定された場合は日本語で、“s00000000”が指定された場合または何も指定されない場合は英語でメッセージが出力されます。

例

```
000010 @OPTIONS MAIN
000020*-----------------------------------------------------------------
000030* 外部スイッチによって、メッセージ言語を切り換えます。
000040*-----------------------------------------------------------------
000050 IDENTIFICATION   DIVISION.
000060 PROGRAM-ID.      SAMPLE.
000070 ENVIRONMENT      DIVISION.
000080  CONFIGURATION   SECTION.
000090*-----------------------------------------------------------------
000100* 外部スイッチを定義します。
000110* ここではひとつしか定義していませんが、最大8個まで定義できます。
000120*-----------------------------------------------------------------
000130  SPECIAL-NAMES.
000140          SWITCH-0 IS SW0 ON  STATUS IS LANG-J
000150                          OFF STATUS IS LANG-E.
000160*-----------------------------------------------------------------
000170 PROCEDURE        DIVISION.
000180     IF LANG-J THEN
000190       DISPLAY NC"こんにちは。ＮｅｔＣＯＢＯＬです。"
000200     ELSE
000210       DISPLAY "Hello, My name is NetCOBOL."
000220     END-IF
000230 END PROGRAM SAMPLE.
```

H.12 機能名-3句

機能名-3を利用して、コマンド行引数や環境変数を操作することができます。

コマンド行引数とは、アプリケーション起動時に渡される引数のことで、ACCEPT文およびDISPLAY文と組み合わせることによって引数の数や値を取り出すことができます。

また、同様にACCEPT文およびDISPLAY文と組み合わせて環境変数の値を操作することもできます。

例

```
000010 @OPTIONS MAIN
000020*---------------------------------------------------------------
000030* 機能名-3を使用してアプリケーション実行時の引数を受け取り、
000040* ソートして表示します。
000050*
000060* サンプル起動例: SAMPLE.EXE  USA England Japan Italy China
000070*---------------------------------------------------------------
000080 IDENTIFICATION   DIVISION.
000090 PROGRAM-ID.      SAMPLE.
000100 ENVIRONMENT      DIVISION.
000110  CONFIGURATION   SECTION.
000120*---------------------------------------------------------------
000130* 引数の数(ARGUMENT-NUMBER)および引数の値(ARGUMENT-VALUE)に対して
000140* 呼び名を割り当てます。
000150*---------------------------------------------------------------
000160  SPECIAL-NAMES.
000170          ARGUMENT-NUMBER IS ARG-NUM
000180          ARGUMENT-VALUE  IS ARG-VAL.
000190*---------------------------------------------------------------
000200  INPUT-OUTPUT    SECTION.
000210  FILE-CONTROL.
000220       SELECT SORT-FILE ASSIGN TO SORT-WORK.
000230 DATA             DIVISION.
000240  FILE            SECTION.
000250  SD SORT-FILE.
000260  01 SORT-REC.
000270    02 SORT-DATA    PIC X(10).
000280  WORKING-STORAGE SECTION.
000290  01 CNT          PIC 9(4) BINARY.
000300  01 ELM-NUM      PIC 9(4) BINARY.
000310 PROCEDURE       DIVISION.
000320*---------------------------------------------------------------
000330* 引数の数は、ACCEPT文によって求めることができます。
000340*---------------------------------------------------------------
000350      ACCEPT ELM-NUM FROM ARG-NUM.
000360*---------------------------------------------------------------
000370      DISPLAY "**  引数をソートします。  **".
000380      SORT SORT-FILE ON ASCENDING KEY SORT-DATA
000390                        INPUT PROCEDURE IS IN-PROC
000400                        OUTPUT PROCEDURE IS OUT-PROC.
000410 IN-PROC.
000420      PERFORM TEST BEFORE
000430              VARYING CNT FROM 1 BY 1 UNTIL CNT > ELM-NUM
000440*---------------------------------------------------------------
000450* 引数の値は、まずDISPLAY文によって何番目かを指定し、その後ACCEPT文
000460* を実行することによって取り出すことができます。
000470*---------------------------------------------------------------
000480        DISPLAY CNT UPON ARG-NUM
000490        ACCEPT SORT-DATA FROM ARG-VAL
000500*---------------------------------------------------------------
000510        RELEASE SORT-REC
```

```
000520      END-PERFORM.
000530 OUT-PROC.
000540      RETURN SORT-FILE AT END GO TO P-EXIT.
000550      DISPLAY SORT-DATA.
000560      GO TO OUT-PROC.
000570 P-EXIT.
000580      EXIT PROGRAM.
000590 END PROGRAM SAMPLE.
```

H.13 ALPHABET句

ALPHABET句は、文字の大小順序に名前(符号系名)をつけます。

ここで定義した符号系名は、PROGRAM COLLATING SEQUENCE句またはSORT文、MERGE文に指定することで有効になります。

例

```
000010 @OPTIONS MAIN
000020*---------------------------------------------------------------
000030* ALPHABET句を利用し、任意の大小順序で文字列をソートします。
000040*---------------------------------------------------------------
000050 IDENTIFICATION   DIVISION.
000060 PROGRAM-ID.      SAMPLE.
000070 ENVIRONMENT      DIVISION.
000080  CONFIGURATION   SECTION.
000090*---------------------------------------------------------------
000100* 通常、ASCIIコードの大小順序は、数字 < 英大 < 英小 ですが、
000110* 英小 < 英大 < 数字 となるオリジナルの符号系名を宣言しています。
000120*---------------------------------------------------------------
000130  SPECIAL-NAMES.
000140         ALPHABET ORG-SEQ IS "a" THRU "z"
000150                             "A" THRU "Z"
000160                             "0" THRU "9".
000170*---------------------------------------------------------------
000180  INPUT-OUTPUT    SECTION.
000190  FILE-CONTROL.
000200       SELECT SORT-FILE ASSIGN TO SORT-WORK.
000210 DATA             DIVISION.
000220  FILE            SECTION.
000230  SD SORT-FILE.
000240  01 SORT-REC.
000250    02 SORT-DATA    PIC X(10).
000260 PROCEDURE       DIVISION.
000270      DISPLAY "**  入力された文字列をソートします。 **".
000280*---------------------------------------------------------------
000290* オリジナルの大小順序でソートします。
000300*---------------------------------------------------------------
000310      SORT SORT-FILE ON ASCENDING KEY SORT-DATA
000320                        COLLATING SEQUENCE IS ORG-SEQ
000330                        INPUT PROCEDURE IS IN-PROC
000340                        OUTPUT PROCEDURE IS OUT-PROC.
000350*---------------------------------------------------------------
000360 IN-PROC.
000370      DISPLAY "Please input the data (If space then end) >> " WITH NO ADVANCING.
000380      ACCEPT SORT-DATA FROM CONSOLE.
000390      IF SORT-DATA NOT = SPACE THEN RELEASE SORT-REC
000400                                   GO TO IN-PROC.
000410      DISPLAY "----------".
000420 OUT-PROC.
000430      RETURN SORT-FILE AT END GO TO P-EXIT.
```

```
000440       DISPLAY SORT-DATA.
000450       GO TO OUT-PROC.
000460 P-EXIT.
000470       EXIT PROGRAM.
000480 END PROGRAM SAMPLE.
```

H.14 CLASS句

CLASS句は、任意の文字集合に対して名前(字類名)をつけます。

定義した字類名は、条件式で利用することができます。

例

```
000010 @OPTIONS MAIN
000020*---------------------------------------------------------------
000030* CLASS句を利用して、入力データの正当性を判定します。
000040*---------------------------------------------------------------
000050 IDENTIFICATION   DIVISION.
000060 PROGRAM-ID.      SAMPLE.
000070 ENVIRONMENT      DIVISION.
000080  CONFIGURATION   SECTION.
000090*---------------------------------------------------------------
000100* 16進文字範囲の字類名を定義します。
000110*---------------------------------------------------------------
000120  SPECIAL-NAMES.
000130          CLASS HEX IS "0" THRU "9"
000140                       "A" THRU "F".
000150*---------------------------------------------------------------
000160 DATA             DIVISION.
000170  WORKING-STORAGE SECTION.
000180  01 IN-DATA      PIC  X(4).
000190 PROCEDURE        DIVISION.
000200       DISPLAY "16進データ(4桁)を入力してください。>> " WITH NO ADVANCING.
000210       ACCEPT IN-DATA FROM CONSOLE.
000220*---------------------------------------------------------------
000230* 定義した字類名は、条件式で使用することができます。
000240*---------------------------------------------------------------
000250       IF IN-DATA IS HEX THEN
000260         DISPLAY "16進数として正しい値です。"
000270       ELSE
000280         DISPLAY "入力データに誤りがあります。"
000290       END-IF
000300*---------------------------------------------------------------
000310 END PROGRAM SAMPLE.
```

H.15 CURRENCY SIGN句

通常、PICTURE句における通貨記号は"¥"ですが、CURRENCY SIGN句によって任意の文字に変更することができます。ただし、指定できない文字もあるので注意してください。

例

```
000010 @OPTIONS MAIN
000020*---------------------------------------------------------------
000030* CURRENCY SIGN句を利用して、日本円をドルに変えて表示します。
000040*---------------------------------------------------------------
000050 IDENTIFICATION   DIVISION.
```

```
000060 PROGRAM-ID.       SAMPLE.
000070 ENVIRONMENT       DIVISION.
000080  CONFIGURATION    SECTION.
000090*-----------------------------------------------------------------
000100* 通貨記号を"$"に変更します。なお、省略時は"¥"です。
000110*-----------------------------------------------------------------
000120   SPECIAL-NAMES.
000130           CURRENCY SIGN IS "$".
000140*-----------------------------------------------------------------
000150 DATA             DIVISION.
000160  WORKING-STORAGE SECTION.
000170  01 YEN          PIC 9(8).
000180*-----------------------------------------------------------------
000190* PICTURE句には"$"を指定します。
000200*-----------------------------------------------------------------
000210  01 DOLLAR       PIC $,$$$,$$$9.99.
000220*-----------------------------------------------------------------
000230 PROCEDURE              DIVISION.
000240     DISPLAY "円をドルに変換します。".
000250     DISPLAY "日本円を入力してください(最大8桁) >> " WITH NO ADVANCING.
000260     ACCEPT YEN.
000270     COMPUTE DOLLAR = YEN / 110.
000280     DISPLAY "**                                    = " DOLLAR " ($1 = ¥110)".
000290 END PROGRAM SAMPLE.
```

H.16 DECIMAL-POINT IS COMMA句

DECIMAL-POINT IS COMMA句を指定すると、コンマ(",")と小数点(".")の機能が入れ替わります。

例

```
000010 @OPTIONS MAIN
000020*-----------------------------------------------------------------
000030*  DECIMAL-POINT IS COMMA句を利用して、コンマと小数点の機能を
000040*  入れ替えます。
000050*-----------------------------------------------------------------
000060 IDENTIFICATION   DIVISION.
000070 PROGRAM-ID.       SAMPLE.
000080 ENVIRONMENT       DIVISION.
000090  CONFIGURATION    SECTION.
000100*-----------------------------------------------------------------
000110* DECIMAL-POINT IS COMMA句を指定します。
000120*-----------------------------------------------------------------
000130  SPECIAL-NAMES.
000140          DECIMAL-POINT IS COMMA.
000150*-----------------------------------------------------------------
000160 DATA             DIVISION.
000170  WORKING-STORAGE  SECTION.
000180  01 走行距離      PIC 9(4).
000190  01 消費燃料      PIC 9(3).
000200*-----------------------------------------------------------------
000210* 小数点は","で表現します。
000220*-----------------------------------------------------------------
000230  01 燃費          PIC 999,99.
000240*-----------------------------------------------------------------
000250 PROCEDURE         DIVISION.
000260     DISPLAY "燃費を計算します。"
000270     DISPLAY "走行距離を入力してください(単位:Km) >> " WITH NO ADVANCING.
000280     ACCEPT  走行距離.
000290     DISPLAY "消費燃料を入力してください(単位:L)  >> " WITH NO ADVANCING.
```

```
000300      ACCEPT  消費燃料.
000310      COMPUTE 燃費 = 走行距離 / 消費燃料.
000320      DISPLAY "燃費は、" 燃費 " Km/L です。".
000330 END PROGRAM SAMPLE.
```

H.17 POSITIONING UNIT句

POSITIONING UNIT句は、印刷位置を決定するための単位名(位置決め単位名)を定義する際に指定します。定義した位置決め単位名は、データ記述項のPRINTING POSITION句に指定します。

例

```
000010 @OPTIONS MAIN
000020*-----------------------------------------------------------------
000030* POSITIONING UNIT句を利用して、印刷位置を制御します。
000040*-----------------------------------------------------------------
000050 IDENTIFICATION   DIVISION.
000060 PROGRAM-ID.      SAMPLE.
000070 ENVIRONMENT      DIVISION.
000080  CONFIGURATION   SECTION.
000090*-----------------------------------------------------------------
000100* 位置決め単位を定義します。
000110* "5 CPI"は、1インチに5文字印字できる間隔を表現しています。
000120*-----------------------------------------------------------------
000130   SPECIAL-NAMES.
000140           POSITIONING UNIT CPI-5 IS 5 CPI.
000150*-----------------------------------------------------------------
000160  INPUT-OUTPUT    SECTION.
000170   FILE-CONTROL.
000180        SELECT PRT-FILE ASSIGN TO PRINTER.
000190 DATA             DIVISION.
000200  FILE            SECTION.
000210  FD PRT-FILE.
000220  01 PRT-REC      PIC X(80).
000230  WORKING-STORAGE SECTION.
000240*-----------------------------------------------------------------
000250* 項目の印字位置を指定します。
000260* - ADDRは、1インチに5文字印字できる間隔を1カラムとして、
000270*   11カラム(=3インチ)目から、
000280* - TEL は、1インチに5文字印字できる間隔を1カラムとして、
000290*   26カラム(=6インチ)目から、印刷されます。
000300*-----------------------------------------------------------------
000310  01 PROF-DATA.
000320    02 NAME         PIC X(16).
000330    02 ADDR         PIC X(32) PRINTING POSITION IS 11 BY CPI-5.
000340    02 TEL          PIC X(15) PRINTING POSITION IS 26 BY CPI-5.
000350*-----------------------------------------------------------------
000360 PROCEDURE        DIVISION.
000370     DISPLAY "Please input your name    >> " WITH NO ADVANCING.
000380     ACCEPT NAME FROM CONSOLE.
000390     DISPLAY "                  address >> " WITH NO ADVANCING.
000400     ACCEPT ADDR FROM CONSOLE.
000410     DISPLAY "                  tel     >> " WITH NO ADVANCING.
000420     ACCEPT TEL  FROM CONSOLE.
000430     DISPLAY "Now printing .....".
000440**
000450     OPEN OUTPUT PRT-FILE.
000460       WRITE PRT-REC FROM PROF-DATA AFTER ADVANCING PAGE.
```

```
000470     CLOSE PRT-FILE.
000480 END PROGRAM SAMPLE.
```

H.18 PRINTING MODE句

PRINTING MODE句は、印刷時の印字属性に名前(印字モード名)を与える際に指定します。定義した印字モード名は、データ記述項のCHARACTER TYPE句に指定します。

例

```
000010 @OPTIONS MAIN
000020*---------------------------------------------------------------
000030* PRINTING MODE句を利用して、印字属性を与えます。
000040*---------------------------------------------------------------
000050 IDENTIFICATION   DIVISION.
000060 PROGRAM-ID.      SAMPLE.
000070 ENVIRONMENT      DIVISION.
000080  CONFIGURATION   SECTION.
000090*---------------------------------------------------------------
000100* 印字属性を定義します。
000110* SIZE 12 POINT: 文字サイズは12ポで、
000120* PITCH 2 CPI  : 文字ピッチは2CPIで、
000130* FONT GOTHIC  : フォントはゴシック体で、
000140* ANGLE 90     : 文字を90度回転(つまり縦書き)させて、
000150* FORM F0201   : 平体で印刷することを表しています。
000160*---------------------------------------------------------------
000170  SPECIAL-NAMES.
000180          PRINTING MODE PRT-ATTR FOR ALL
000190                                 IN SIZE 12 POINT
000200                                 AT PITCH 2 CPI
000210                                 WITH FONT GOTHIC
000220                                 AT ANGLE 90 DEGREES
000230                                 BY FORM F0201.
000240*---------------------------------------------------------------
000250  INPUT-OUTPUT    SECTION.
000260   FILE-CONTROL.
000270        SELECT PRT-FILE ASSIGN TO PRINTER.
000280 DATA             DIVISION.
000290  FILE            SECTION.
000300  FD PRT-FILE.
000310  01 PRT-REC      PIC X(80).
000320  WORKING-STORAGE SECTION.
000330*---------------------------------------------------------------
000340* 定義した印字属性は、CHARACTER TYPE句に指定します。
000350*---------------------------------------------------------------
000360  01 PRT-DATA     PIC N(10) CHARACTER TYPE IS PRT-ATTR.
000370*---------------------------------------------------------------
000380 PROCEDURE        DIVISION.
000390     DISPLAY "Now printing .....".
000400     OPEN OUTPUT PRT-FILE
000410       MOVE NC"富士通株式会社" TO PRT-DATA
000420       WRITE PRT-REC FROM PRT-DATA AFTER ADVANCING PAGE.
000430     CLOSE PRT-FILE.
000440 END PROGRAM SAMPLE.
```

H.19 SYMBOLIC CHARACTERS句

SYMBOLIC CHARACTERS句は、文字コードに名前を与える(記号文字)際に指定します。制御文字に名前を与えたい場合などに利用できます。

例

```
000010 @OPTIONS MAIN
000020*---------------------------------------------------------------------
000030* SYMBOLIC CHARACTERS句を利用して、TABコードを制御します。
000040* このサンプルを実行すると、カレントフォルダ（ディレクトリ）に
000050*  "SAMPLE.TXT" の名前で行順ファイルが作成されます。
000060*---------------------------------------------------------------------
000070 IDENTIFICATION   DIVISION.
000080 PROGRAM-ID.      SAMPLE.
000090 ENVIRONMENT      DIVISION.
000100  CONFIGURATION   SECTION.
000110*---------------------------------------------------------------------
000120* SYMBOLIC CHARACTERS句に指定する整数は文字の順序位置、つまり、
000130* ASCIIコード表の X"00" を1とした文字の並びの順番です。
000140* TABコードは X"09" のため、整数には10を指定します。
000150*---------------------------------------------------------------------
000160  SPECIAL-NAMES.
000170          SYMBOLIC CHARACTERS TAB IS 10.
000180*---------------------------------------------------------------------
000190  INPUT-OUTPUT    SECTION.
000200  FILE-CONTROL.
000210       SELECT TXT-FILE ASSIGN TO "SAMPLE.TXT"
000220                       ORGANIZATION IS LINE SEQUENTIAL.
000230 DATA             DIVISION.
000240  FILE            SECTION.
000250  FD TXT-FILE.
000260  01 TXT-REC      PIC X(80).
000270 PROCEDURE        DIVISION.
000280     OPEN OUTPUT TXT-FILE.
000290*---------------------------------------------------------------------
000300* 定義した記号文字は、通常の文字定数と同じように利用できます。
000310* 以下では連結式に使用しています。
000320*---------------------------------------------------------------------
000330       MOVE "富士通株式会社" & TAB & "営業部" & TAB & "富士通太郎" TO TXT-REC.
000340*---------------------------------------------------------------------
000350       WRITE TXT-REC.
000360     CLOSE TXT-FILE.
000370 END PROGRAM SAMPLE.
```

H.20 SYMBOLIC CONSTANT句

SYMBOLIC CONSTANT句は、定数に名前を与える(記号定数)際に利用します。定数値変更時の影響が局所化できることから、保守性(拡張性)が向上します。

例

```
000010 @OPTIONS MAIN
000020*---------------------------------------------------------------------
000030* SYMBOLIC CONSTANT句で定数に名前をつけます。
000040*---------------------------------------------------------------------
000050 IDENTIFICATION   DIVISION.
000060 PROGRAM-ID.      SAMPLE.
000070 ENVIRONMENT      DIVISION.
```

```
000080  CONFIGURATION   SECTION.
000090*-----------------------------------------------------------------
000100* 消費税率を定義しておきます。
000110* 記号定数化しておくことによって、将来の変化に容易に対応できます。
000120* →税率が上がった場合、記号定数を更新して再翻訳するだけです。
000130*-----------------------------------------------------------------
000140  SPECIAL-NAMES.
000150          SYMBOLIC CONSTANT
000160          消費税率 IS 0.05.
000170*-----------------------------------------------------------------
000180 DATA              DIVISION.
000190  WORKING-STORAGE SECTION.
000200  01 値段          PIC 9(8).
000210  01 消費税        PIC ZZZZZ9.
000220 PROCEDURE        DIVISION.
000230     DISPLAY "消費税を計算します。".
000240     DISPLAY "あなたの欲しいものの値段は？ >> " WITH NO ADVANCING.
000250     ACCEPT  値段.
000260*-----------------------------------------------------------------
000270* 記号定数は、通常の定数と同じように使用できます。
000280*-----------------------------------------------------------------
000290     COMPUTE 消費税 = 値段 * 消費税率.
000300     DISPLAY "消費税率は " 消費税率 " で、税額は " 消費税 " 円です。".
000310*-----------------------------------------------------------------
000320     EXIT PROGRAM.
000330 END PROGRAM SAMPLE.
```

H.21 RENAMES句

RENAMES句は、一連のデータの並びに対して別名を与えることができます。

レベル番号66の記述項に、名前と範囲を指定してください。

例

```
000010 @OPTIONS MAIN
000020*-----------------------------------------------------------------
000030*  RENAMES句によって、領域を再命名することができます。
000040*-----------------------------------------------------------------
000050 IDENTIFICATION   DIVISION.
000060 PROGRAM-ID.      SAMPLE.
000070 DATA             DIVISION.
000080  WORKING-STORAGE SECTION.
000090  01 従業員データ.
000100     02 従業員番号   PIC 9(8)  VALUE 19990120.
000110     02             PIC X(2)  VALUE SPACE.
000120     02 所属         PIC X(24) VALUE "営業部".
000130     02 氏名         PIC X(24) VALUE "富士通太郎".
000140     02 住所         PIC X(24) VALUE "静岡県沼津市".
000150     02 電話番号     PIC X(15) VALUE "(0123)45-6789".
000160*-----------------------------------------------------------------
000170* RENAMES句を利用することによって、別の枠組みを宣言できます。
000180* 従業員情報: 従業員番号から氏名までを再命名しています。
000190* 個人情報  : 氏名から電話番号までを再命名しています。
000200*-----------------------------------------------------------------
000210     66 従業員情報   RENAMES 従業員番号 THRU 氏名.
000220     66 個人情報     RENAMES 氏名       THRU 電話番号.
000230*-----------------------------------------------------------------
000240  77 情報種別     PIC 9.
000250 PROCEDURE        DIVISION.
000260     DISPLAY "情報種別を入力してください。"
```

```
000270      DISPLAY "従業員情報:1 個人情報:2 全情報:3 >> " WITH NO ADVANCING.
000280      ACCEPT 情報種別.
000290*-----------------------------------------------------------------
000300* 再命名されたデータ名は、通常の集団項目と同じように利用できます。
000310*-----------------------------------------------------------------
000320      EVALUATE 情報種別
000330      WHEN 1
000340        DISPLAY 従業員情報
000350      WHEN 2
000360        DISPLAY 個人情報
000370      WHEN 3
000380        DISPLAY 従業員データ
000390      WHEN OTHER
000400        DISPLAY "入力データに異常があります。"
000410      END-EVALUATE.
000420*-----------------------------------------------------------------
000430 END PROGRAM SAMPLE.
```

H.22 VALUE句

ある項目の値に対して条件名を与えたい場合、レベル番号88の記述項で条件名を定義することができます。定義された条件名は条件式で利用できます。

例

```
000010 @OPTIONS MAIN
000020*-----------------------------------------------------------------
000030* 88レベル項目を利用して、データ宣言時に条件名を定義しておくこと
000040* ができます。
000050*-----------------------------------------------------------------
000060 IDENTIFICATION   DIVISION.
000070 PROGRAM-ID.      SAMPLE.
000080 DATA             DIVISION.
000090  WORKING-STORAGE SECTION.
000100  01 TODAY.
000110    02            PIC 9(2).
000120*-----------------------------------------------------------------
000130* 値に合わせて条件名を定義しておきます。
000140*-----------------------------------------------------------------
000150    02 MONTH      PIC 9(2).
000160      88 SPRING-M   VALUE  3 THRU  5.
000170      88 SUMMER-M   VALUE  6 THRU  8.
000180      88 AUTUMN-M   VALUE  9 THRU 11.
000190      88 WINTER-M   VALUE 12, 1, 2.
000200*-----------------------------------------------------------------
000210    02            PIC 9(2).
000220 PROCEDURE        DIVISION.
000230      ACCEPT TODAY FROM DATE.
000240*-----------------------------------------------------------------
000250* 条件名は条件式で利用することができます。
000260*-----------------------------------------------------------------
000270      EVALUATE TRUE
000280      WHEN SPRING-M
000290        DISPLAY "Now in the SPRING.  Let's go hiking!!"
000300      WHEN SUMMER-M
000310        DISPLAY "Now in the SUMMER.  Let's go swimming!!"
000320      WHEN AUTUMN-M
000330        DISPLAY "Now in the AUTUMN.  Let's go tennis!!"
000340      WHEN WINTER-M
000350        DISPLAY "Now in the WINTER.  Let's go skiing!!"
```

```
000360       END-EVALUATE
000370*-----------------------------------------------------------------
000380 END PROGRAM SAMPLE.
```

H.23 BLANK WHEN ZERO句

BLANK WHEN ZERO句は、値がゼロのとき空白に置き換えます。

値がゼロの場合には無効データとしたい場合などに利用します。

例

```
000010 @OPTIONS MAIN
000020*-----------------------------------------------------------------
000030* BLANK WHEN ZERO句は、値が０の場合に空白とします。
000040*-----------------------------------------------------------------
000050 IDENTIFICATION   DIVISION.
000060 PROGRAM-ID.      SAMPLE.
000070 DATA             DIVISION.
000080  WORKING-STORAGE SECTION.
000090*-----------------------------------------------------------------
000100* 選択科目(日本史/世界史)には、BLANK WHEN ZERO句を指定しておきます。
000110*-----------------------------------------------------------------
000120  01 テスト結果.
000130    02 英語         PIC 9(3).
000140    02              PIC X(3) VALUE SPACE.
000150    02 数学         PIC 9(3).
000160    02              PIC X(5) VALUE SPACE.
000170    02 日本史       PIC 9(3) BLANK WHEN ZERO.
000180    02              PIC X(5) VALUE SPACE.
000190    02 世界史       PIC 9(3) BLANK WHEN ZERO.
000200*-----------------------------------------------------------------
000210  77 TEMP         PIC 9(3).
000220 PROCEDURE        DIVISION.
000230     DISPLAY "テスト結果を入力してください。".
000240     DISPLAY "なお、選択してない教科には 0 を入力してください。".
000250     DISPLAY "英語   >> " WITH NO ADVANCING
000260     ACCEPT 英語.
000270     DISPLAY "数学   >> " WITH NO ADVANCING
000280     ACCEPT 数学.
000290     DISPLAY "日本史 >> " WITH NO ADVANCING
000300     ACCEPT TEMP.
000310     MOVE TEMP TO 日本史.
000320     DISPLAY "世界史 >> " WITH NO ADVANCING
000330     ACCEPT TEMP.
000340     MOVE TEMP TO 世界史.
000350**
000360     DISPLAY " ".
000370     DISPLAY "      英語  数学  日本史  世界史".
000380     DISPLAY "結果:  " テスト結果.
000390 END PROGRAM SAMPLE.
```

H.24 GLOBAL句

GLOBAL句を指定したデータ項目には、大域属性が与えられます。

大域属性が与えられたデータ項目は子プログラムからも参照できるため、共通に使いたいデータを定義する場合に利用します。

例

```
000010 @OPTIONS MAIN
000020*---------------------------------------------------------------------
000030* GLOBAL句によって、大域属性を持つ変数を定義することができます。
000040*---------------------------------------------------------------------
000050 IDENTIFICATION   DIVISION.
000060 PROGRAM-ID.      SAMPLE.
000070 DATA             DIVISION.
000080  WORKING-STORAGE SECTION.
000090*---------------------------------------------------------------------
000100* GLOBAL句を指定して、大域属性を宣言します。
000110* 大域属性を持った変数は、子プログラムで参照することができます。
000120*---------------------------------------------------------------------
000130  01 IN-DATA      IS GLOBAL PIC X(80).
000140  01 ED-DATA      IS GLOBAL.
000150    02 ELM          OCCURS 8 TIMES PIC X(16).
000160*---------------------------------------------------------------------
000170 PROCEDURE       DIVISION.
000180     CALL "IN-PROC".
000190     CALL "ED-PROC".
000200     CALL "OUT-PROC".
000210     EXIT PROGRAM.
000220**
000230 IDENTIFICATION   DIVISION.
000240 PROGRAM-ID.      IN-PROC.
000250 PROCEDURE        DIVISION.
000260     DISPLAY "入力された文字列を空白で分離します。"
000270     DISPLAY "文字列を入力してください >> " WITH NO ADVANCING.
000280*---------------------------------------------------------------------
000290* 大域属性を持った項目 IN-DATA に値を設定します。
000300*---------------------------------------------------------------------
000310     ACCEPT IN-DATA FROM CONSOLE.
000320*---------------------------------------------------------------------
000330     EXIT PROGRAM.
000340 END PROGRAM IN-PROC.
000350**
000360 IDENTIFICATION   DIVISION.
000370 PROGRAM-ID.      ED-PROC.
000380 PROCEDURE        DIVISION.
000390*---------------------------------------------------------------------
000400* 大域属性を持った項目 IN-DATA を編集します。
000410*---------------------------------------------------------------------
000420     UNSTRING IN-DATA DELIMITED BY SPACE
000430                      INTO ELM(1) ELM(2) ELM(3) ELM(4)
000440                           ELM(5) ELM(6) ELM(7) ELM(8).
000450*---------------------------------------------------------------------
000460     EXIT PROGRAM.
000470 END PROGRAM ED-PROC.
000480**
000490 IDENTIFICATION   DIVISION.
000500 PROGRAM-ID.      OUT-PROC.
000510 PROCEDURE        DIVISION.
000520     DISPLAY " ".
000530*---------------------------------------------------------------------
000540* 大域属性を持った項目 ED-DATA を参照します。
000550*---------------------------------------------------------------------
000560     DISPLAY "分解データ: " ED-DATA.
000570*---------------------------------------------------------------------
000580     EXIT PROGRAM.
```

```
000590 END PROGRAM OUT-PROC.
000600 END PROGRAM SAMPLE.
```

H.25 JUSTIFIED句

通常、文字データは左よせで転記されますが、JUSTIFIED句を指定した項目は右よせで転記されます。データを右よせで表示したい場合などに利用します。

例

```
000010 @OPTIONS MAIN
000020*---------------------------------------------------------------
000030*  JUSTIFIED RIGHT句は、文字列の右よせ転記を指示します。
000040*---------------------------------------------------------------
000050 IDENTIFICATION   DIVISION.
000060 PROGRAM-ID.      SAMPLE.
000070 DATA             DIVISION.
000080  WORKING-STORAGE SECTION.
000090*---------------------------------------------------------------
000100* 右寄せで格納したい項目には JUSTIFIED RIGHT句を指定します。
000110* 表示上、右寄せで統一したいような場合に利用します。
000120*---------------------------------------------------------------
000130  01 CARD.
000140    02 TEAM-1       PIC X(10).
000150    02              PIC X(02) VALUE "VS".
000160    02 TEAM-2       PIC X(10) JUSTIFIED RIGHT.
000170    02              PIC X(02) VALUE SPACE.
000180    02 PLACE        PIC X(12).
000190*---------------------------------------------------------------
000200 PROCEDURE        DIVISION.
000210     DISPLAY "** 本日の対戦カード **".
000220     DISPLAY " ".
000230     MOVE "Japan"   TO TEAM-1.
000240*---------------------------------------------------------------
000250* 文字列は右よせで格納されます。
000260*---------------------------------------------------------------
000270     MOVE "Brazil"  TO TEAM-2.
000280*---------------------------------------------------------------
000290     MOVE "国立"  TO PLACE.
000300     DISPLAY CARD.
000310**
000320     MOVE "Italia"     TO TEAM-1.
000330     MOVE "Argentina"  TO TEAM-2.
000340     MOVE "横浜国際" TO PLACE.
000350     DISPLAY CARD.
000360 END PROGRAM SAMPLE.
```

H.26 OCCURS句

OCCURS句は、繰返しデータを定義する際に指定します。

繰返しデータの入れ子(最大7階層まで)や、可変の繰返し回数も定義可能なため、規則化された繰返しデータの集合を簡単に定義することができます。

OCCURS句(書き方1) 一定の繰返し回数

例

```
000010 @OPTIONS MAIN
000020*---------------------------------------------------------------
000030* OCCURS句 【書き方1】
000040* 配列データを定義します。
000050*---------------------------------------------------------------
000060 IDENTIFICATION   DIVISION.
000070 PROGRAM-ID.      SAMPLE.
000080 DATA             DIVISION.
000090  WORKING-STORAGE SECTION.
000100  01 住人.
000110    02              PIC N(8) VALUE NC"１０１号室：鈴木".
000120    02              PIC N(8) VALUE NC"１０２号室：中村".
000130    02              PIC N(8) VALUE NC"１０３号室：佐藤".
000140    02              PIC N(8) VALUE NC"２０１号室：山本".
000150    02              PIC N(8) VALUE NC"２０２号室：木村".
000160    02              PIC N(8) VALUE NC"２０３号室：田中".
000170*---------------------------------------------------------------
000180* 住人データを配列化します。
000190* コーポ富士通は2階建てで、1階につき3部屋あります(2次元配列)。
000200*---------------------------------------------------------------
000210  01              REDEFINES 住人.
000220    02 フロア       OCCURS 2 TIMES.
000230      03 入居者       OCCURS 3 TIMES PIC N(8).
000240*---------------------------------------------------------------
000250  77 フロア番号    PIC 9(1).
000260  77 部屋番号      PIC 9(1).
000270 PROCEDURE        DIVISION.
000280     DISPLAY "コーポ富士通 入居者案内です。".
000290     DISPLAY "部屋は何階ですか？   (1～2) >> " WITH NO ADVANCING.
000300     ACCEPT フロア番号.
000310     DISPLAY "左から何番目ですか？ (1～3) >> " WITH NO ADVANCING.
000320     ACCEPT 部屋番号.
000330     DISPLAY " ".
000340*---------------------------------------------------------------
000350* 配列化されたデータは、添字によって参照できます。
000360* この場合、2次元配列なので、2つの添字によってデータを特定します。
000370*---------------------------------------------------------------
000380     DISPLAY 入居者(フロア番号, 部屋番号) "さんが入居しています。".
000390*---------------------------------------------------------------
000400 END PROGRAM SAMPLE.
```

OCCURS句(書き方2) 可変の繰返し回数

OCCURS句(書き方2)は、OCCURS句を指定したデータ項目の反復回数が可変である時に使用します。

OCCURS句を指定したデータ項目の大きさが可変なのではなく、反復回数が可変であることに注意してください。

例

```
000010 @OPTIONS MAIN
000020*---------------------------------------------------------------
000030* OCCURS句 【書き方2】
000040* 繰返し数を動的に変更できます。
000050*---------------------------------------------------------------
000060 IDENTIFICATION   DIVISION.
000070 PROGRAM-ID.      SAMPLE.
000080 DATA             DIVISION.
000090  WORKING-STORAGE SECTION.
000100*---------------------------------------------------------------
```

```
000110* DEPENDING ON句にデータ名を指定することによって、プログラム実行時
000120* に動的に繰返し数を制御することができます。
000130*-----------------------------------------------------------------
000140  01 DSP-DATA.
000150    02              OCCURS 80 TIMES DEPENDING ON LEN.
000160      03              PIC X VALUE "*".
000170  77 LEN          PIC 9(2).
000180*-----------------------------------------------------------------
000190 PROCEDURE        DIVISION.
000200     DISPLAY "何バイト表示しますか？ (1～80) >> " WITH NO ADVANCING.
000210     ACCEPT LEN.
000220*-----------------------------------------------------------------
000230* DEPENDING ON句に指定したデータ(LEN)の値が繰返し数になります。
000240*-----------------------------------------------------------------
000250     DISPLAY DSP-DATA.
000260*-----------------------------------------------------------------
000270 END PROGRAM SAMPLE.
```

H.27 PICTURE句

数字編集項目を利用して、数字データを任意の形式に編集することができます。

このサンプルは[Win32]で動作します。

例

```
000010 @OPTIONS MAIN
000020*-----------------------------------------------------------------
000030* PICTURE句で数字編集項目文字を定義します。
000040*-----------------------------------------------------------------
000050 IDENTIFICATION   DIVISION.
000060 PROGRAM-ID.      SAMPLE.
000070 DATA             DIVISION.
000080  WORKING-STORAGE SECTION.
000090*-----------------------------------------------------------------
000100* 数字編集項目の定義例です。
000110* 実行させて、動作を確認してください。
000130*-----------------------------------------------------------------
000140  01 NUMEDIT-1    PIC   ----,--9.
000150  01 NUMEDIT-2    PIC   -ZZZ,ZZ9.
000160  01 NUMEDIT-3    PIC    ¥¥¥,¥¥9.
000170  01 NUMEDIT-4    PIC 9999/99/99.
000180  01 NUMEDIT-5    PIC   +++9.999.
000190  01 NUMEDIT-6    PIC *********9.
000200*-----------------------------------------------------------------
000210  01 WDATE.
000220    02 TODAY        PIC 9(8).
000230 PROCEDURE      DIVISION.
000240     DISPLAY "INPUT -> PICTURE        = OUTPUT"
000250     DISPLAY "-------------------------------------"
000260**
000270     MOVE -3000  TO  NUMEDIT-1
000280     DISPLAY "-3000 -> PIC   ----,--9 =   "  NUMEDIT-1
000290**
000300     MOVE   980  TO  NUMEDIT-2
000310     DISPLAY "  980 -> PIC   -ZZZ,ZZ9 =   "  NUMEDIT-2
000320**
000330     MOVE  3210  TO  NUMEDIT-3
000340     DISPLAY " 3210 -> PIC    ¥¥¥,¥¥9 =    " NUMEDIT-3
000350**
000360     MOVE FUNCTION CURRENT-DATE  TO  WDATE
```

```
000370      MOVE TODAY  TO  NUMEDIT-4
000380      DISPLAY "TODAY -> PIC 9999/99/99 = "    NUMEDIT-4
000390**
000400      MOVE 12.34  TO  NUMEDIT-5
000410      DISPLAY "12.34 -> PIC   +++9.999 =   "  NUMEDIT-5
000420**
000430      MOVE 67890  TO  NUMEDIT-6
000440      DISPLAY "67890 -> PIC *********9 = "    NUMEDIT-6
000450 END PROGRAM SAMPLE.
```

H.28 REDEFINES句

REDEFINES句は、データ項目を再定義する場合に指定します。

ある項目を別の項目属性で参照する場合や、ひとつのデータ域に別の役割を持たせる場合などに利用します。

例

```
000010 @OPTIONS MAIN
000020*---------------------------------------------------------------
000030* REDEFINES句は、項目を再定義する際に利用します。
000040*---------------------------------------------------------------
000050 IDENTIFICATION   DIVISION.
000060 PROGRAM-ID.      SAMPLE.
000070 DATA             DIVISION.
000080  WORKING-STORAGE SECTION.
000090*---------------------------------------------------------------
000100* REDEFINES句には、再定義される項目の名前を指定します。
000110* 再定義する項目の属性は、再定義される項目の属性と異なっていても
000120* かまいませんが、内部表現を意識した上でご使用ください。
000130*---------------------------------------------------------------
000140  01 IN-CHAR      PIC X(1).
000150  01 BOOL         REDEFINES IN-CHAR PIC 1(8) BIT.
000160*---------------------------------------------------------------
000170 PROCEDURE        DIVISION.
000180      DISPLAY "入力された文字をビット表現で表示します。".
000190      DISPLAY "文字(1文字)を入力してください。>> " WITH NO ADVANCING.
000200      ACCEPT IN-CHAR.
000210      DISPLAY "文字""" IN-CHAR """のビット表現は、" BOOL " です。".
000220 END PROGRAM SAMPLE.
```

H.29 SIGN句

数字データに符号を持たせる場合、PICTURE句の文字列にSを付加します。

このとき、符号にはいくつかの表現形式(データ中の符号部で管理したり、符号文字を保持するなど)があるため、SIGN句によって指定します。

例

```
000010 @OPTIONS MAIN
000020*---------------------------------------------------------------
000030* SIGN句は、符号の形式を指定します。
000040*---------------------------------------------------------------
000050 IDENTIFICATION   DIVISION.
000060 PROGRAM-ID.      SAMPLE.
000070 DATA             DIVISION.
000080  WORKING-STORAGE SECTION.
```

```
000090  77 UNSIGNED         PIC  9(4).
000100  77 SIGNED           PIC S9(4).
000110  77 TRAIL-SIGNED     PIC S9(4) SIGN TRAILING.
000120  77 LEAD-SIGNED      PIC S9(4) SIGN LEADING.
000130  77 TRAIL-SEP        PIC S9(4) SIGN TRAILING SEPARATE.
000140  77 LEAD-SEP         PIC S9(4) SIGN LEADING  SEPARATE.
000150 PROCEDURE        DIVISION.
000160*-----------------------------------------------------------------
000170* SIGNなしの場合、値は絶対値で格納されます。
000180* 以下の場合、項目UNSIGNEDの内容は、X"31323334" になります。
000190*-----------------------------------------------------------------
000200      MOVE +1234 TO UNSIGNED.
000210      DISPLAY "+1234 ->  9(4)                          =  " UNSIGNED.
000220*-----------------------------------------------------------------
000230* SIGN付きの場合、符号はTRAILING SIGN形式と同じように、最下位桁の
000240* 上位4ビットで管理します(正:X"4x" 負:X"5x")。
000250* 以下の場合、項目SIGNEDの内容は、X"31323344" になります。
000260*-----------------------------------------------------------------
000270      MOVE +1234 TO SIGNED.
000280      DISPLAY "+1234 -> S9(4)                          = " SIGNED.
000290*-----------------------------------------------------------------
000300* TRAILING指定の場合、符号は最下位桁の上位4ビットで管理します
000310* (正:X"4x" 負:X"5x")。
000320* 以下の場合、項目TRAIL-SIGNEDの内容は、X"31323344" になります。
000330*-----------------------------------------------------------------
000340      MOVE +1234 TO TRAIL-SIGNED.
000350      DISPLAY "+1234 -> S9(4) SIGN TRAILING          = " TRAIL-SIGNED.
000360*-----------------------------------------------------------------
000370* LEADING指定の場合、符号は最上位桁の上位4ビットで管理します
000380* (正:X"4x" 負:X"5x")。
000390* 以下の場合、項目LEAD-SIGNEDの内容は、X"41323334" になります。
000400*-----------------------------------------------------------------
000410      MOVE +1234 TO LEAD-SIGNED.
000420      DISPLAY "+1234 -> S9(4) SIGN LEADING           = " LEAD-SIGNED.
000430*-----------------------------------------------------------------
000440* TRAILING SEPARATE指定の場合、符号は最下位桁の右に文字情報で付加
000450* されます。(正:X"2B" 負:X"2D")。
000460* 以下の場合、項目TRAIL-SEPの内容は、X"313233342B" になります。
000470*-----------------------------------------------------------------
000480      MOVE +1234 TO TRAIL-SEP.
000490      DISPLAY "+1234 -> S9(4) SIGN TRAILING SEPARATE = " TRAIL-SEP.
000500*-----------------------------------------------------------------
000510* LEADING SEPARATE指定の場合、符号は最上位桁の左に文字情報で付加
000520* されます。(正:X"2B" 負:X"2D")。
000530* 以下の場合、項目LEAD-SEPの内容は、X"2B31323334" になります。
000540*-----------------------------------------------------------------
000550      MOVE +1234 TO LEAD-SEP.
000560      DISPLAY "+1234 -> S9(4) SIGN LEADING SEPARATE  = " LEAD-SEP.
000570 END PROGRAM SAMPLE.
```

H.30 TYPEDEF句

TYPEDEF句によって、任意のデータ型を定義できます。

定義した型は、TYPE句で参照します。

なお、TYPEDEF句およびTYPE句は[Win32][WinIPF][Winx64][Solaris][Linux][LinuxIPF][Linux64][.NET]固有機能です。

例

```
000010 @OPTIONS MAIN
000020*---------------------------------------------------------------
000030* TYPEDEF句は、任意のデータ型を定義できます。
000040* 定義した型は、TYPE句によって参照できます。
000050*
000060* このサンプルでは、日付型を利用してADD文の実行性能を計測します。
000070*---------------------------------------------------------------
000080 IDENTIFICATION   DIVISION.
000090 PROGRAM-ID.      SAMPLE.
000100 DATA             DIVISION.
000110  WORKING-STORAGE SECTION.
000120*---------------------------------------------------------------
000130* TYPEDEF句を利用して、日付型(DATE-EDITED)を定義します。
000140*---------------------------------------------------------------
000150  01 DATE-EDITED  TYPEDEF.
000160    02 YEARS        PIC 9(4).
000170    02             PIC X(1) VALUE "/".
000180    02 MONTHS       PIC 9(2).
000190    02             PIC X(1) VALUE "/".
000200    02 DAYS         PIC 9(2).
000210    02             PIC X(1) VALUE "/".
000220    02 HOURS        PIC 9(2).
000230    02             PIC X(1) VALUE ":".
000240    02 MINUTES      PIC 9(2).
000250    02             PIC X(1) VALUE ":".
000260    02 SECONDS      PIC 9(2).
000270    02             PIC X(1) VALUE ".".
000280    02 M-SECS       PIC 9(2).
000290*---------------------------------------------------------------
000300* 上で定義した日付型は、TYPE句によって利用できます。
000310*---------------------------------------------------------------
000320  01 STARTED      TYPE DATE-EDITED.
000330  01 ENDED        TYPE DATE-EDITED.
000340*---------------------------------------------------------------
000350  01 WK-DATE.
000360    02  YEARS       PIC  9(4).
000370    02  MONTHS      PIC  9(2).
000380    02  DAYS        PIC  9(2).
000390    02  HOURS       PIC  9(2).
000400    02  MINUTES     PIC  9(2).
000410    02  SECONDS     PIC  9(2).
000420    02  M-SECS      PIC  9(2).
000430  01 COUNTER      PIC S9(8) VALUE 0.
000440 PROCEDURE        DIVISION.
000450      MOVE FUNCTION CURRENT-DATE TO WK-DATE.
000460      MOVE CORR WK-DATE          TO STARTED.
000470      DISPLAY "STARTED-TIME IS " STARTED.
000480**
000490      PERFORM 1000000 TIMES
000500        ADD 1 TO  COUNTER
000510      END-PERFORM.
000520**
000530      MOVE FUNCTION CURRENT-DATE TO  WK-DATE.
000540      MOVE CORR WK-DATE          TO ENDED.
000550      DISPLAY "ENDED-TIME   IS " ENDED.
000560      EXIT PROGRAM.
000570 END PROGRAM SAMPLE.
```

H.31 TYPE句

TYPEDEF句によって定義したデータ型をTYPE句に指定することで、その型を持つデータ項目を定義することができます。

なお、TYPEDEF句およびTYPE句は[Win32][WinIPF][Winx64][Solaris][Linux][LinuxIPF][Linux64][.NET]固有機能です。

例

```
000010 @OPTIONS MAIN
000020*---------------------------------------------------------------
000030* TYPEDEF句は、任意のデータ型を定義できます。
000040* 定義した型は、TYPE句によって参照できます。
000050*
000060* このサンプルでは、日付型を利用してADD文の実行性能を計測します。
000070*---------------------------------------------------------------
000080 IDENTIFICATION   DIVISION.
000090 PROGRAM-ID.      SAMPLE.
000100 DATA             DIVISION.
000110  WORKING-STORAGE SECTION.
000120*---------------------------------------------------------------
000130* TYPEDEF句を利用して、日付型(DATE-EDITED)を定義します。
000140*---------------------------------------------------------------
000150  01 DATE-EDITED  TYPEDEF.
000160     02 YEARS        PIC 9(4).
000170     02             PIC X(1) VALUE "/".
000180     02 MONTHS       PIC 9(2).
000190     02             PIC X(1) VALUE "/".
000200     02 DAYS         PIC 9(2).
000210     02             PIC X(1) VALUE "/".
000220     02 HOURS        PIC 9(2).
000230     02             PIC X(1) VALUE ":".
000240     02 MINUTES      PIC 9(2).
000250     02             PIC X(1) VALUE ":".
000260     02 SECONDS      PIC 9(2).
000270     02             PIC X(1) VALUE ".".
000280     02 M-SECS       PIC 9(2).
000290*---------------------------------------------------------------
000300* 上で定義した日付型は、TYPE句によって利用できます。
000310*---------------------------------------------------------------
000320  01 STARTED      TYPE DATE-EDITED.
000330  01 ENDED        TYPE DATE-EDITED.
000340*---------------------------------------------------------------
000350  01 WK-DATE.
000360     02  YEARS       PIC  9(4).
000370     02  MONTHS      PIC  9(2).
000380     02  DAYS        PIC  9(2).
000390     02  HOURS       PIC  9(2).
000400     02  MINUTES     PIC  9(2).
000410     02  SECONDS     PIC  9(2).
000420     02  M-SECS      PIC  9(2).
000430  01 COUNTER      PIC S9(8) VALUE 0.
000440 PROCEDURE        DIVISION.
000450       MOVE FUNCTION CURRENT-DATE TO WK-DATE.
000460       MOVE CORR WK-DATE          TO STARTED.
000470       DISPLAY "STARTED-TIME IS " STARTED.
000480**
000490       PERFORM 1000000 TIMES
000500         ADD 1 TO  COUNTER
000510       END-PERFORM.
000520**
```

```
000530       MOVE FUNCTION CURRENT-DATE TO  WK-DATE.
000540       MOVE CORR WK-DATE          TO ENDED.
000550       DISPLAY "ENDED-TIME   IS " ENDED.
000560       EXIT PROGRAM.
000570 END PROGRAM SAMPLE.
```

H.32 BASED ON句

通常、基底場所節で定義したデータ項目はポインタ付けして参照しますが、BASED ON句で特定のポインタデータ項目を指定することにより、ポインタ付けすることなく、データを参照できるようになります。

例

```
000010 @OPTIONS MAIN
000020*---------------------------------------------------------------
000030* BASED ON句は、暗にポインタ付けして参照することを可能にします。
000040*---------------------------------------------------------------
000050 IDENTIFICATION   DIVISION.
000060 PROGRAM-ID.      SAMPLE.
000070 DATA             DIVISION.
000080*---------------------------------------------------------------
000090* BASED ON句を指定することによって、定義した項目を暗にポインタ
000100* 付けして参照できます。
000110*---------------------------------------------------------------
000120  BASED-STORAGE   SECTION.
000130  01              BASED ON MENU-PTR.
000140    02 MENU         OCCURS 3 TIMES.
000150      03 M-NAME       PIC X(20).
000160      03 M-DETAIL     PIC X(30).
000170*---------------------------------------------------------------
000180  WORKING-STORAGE SECTION.
000190  01  MENU-NO     PIC 9(1).
000200  01  MENU-PTR    POINTER.
000210  CONSTANT        SECTION.
000220  01 SAMPLE-DATA.
000230    02  MENU-1.
000240      03        PIC X(20)  VALUE "A-Lunch".
000250      03        PIC X(30)  VALUE "Curry rice,Salad,Fruit".
000260    02  MENU-2.
000270      03        PIC X(20)  VALUE "B-Lunch".
000280      03        PIC X(30)  VALUE "Sandwich,Salad,Coffee".
000290    02  MENU-3.
000300      03        PIC X(20)  VALUE "C-Lunch".
000310      03        PIC X(30)  VALUE "Spaghetti,Salad,Ice cream".
000320 PROCEDURE        DIVISION.
000330     DISPLAY "*** Today's Lunch Menu ***".
000340     MOVE FUNCTION ADDR(SAMPLE-DATA) TO MENU-PTR.
000350     PERFORM TEST BEFORE
000360             VARYING MENU-NO FROM 1 BY 1 UNTIL MENU-NO > 3
000370*---------------------------------------------------------------
000380* 暗にポインタ付けして参照できます。
000390* 以下の場合、MENU-NAMEはMENU-PTR->M-NAMEと、
000400* M-DETAILはMENU-PTR->M-DETAILと記述した場合と同じ扱いになります。
000410*---------------------------------------------------------------
000420       DISPLAY " "
000430       DISPLAY "Name    : " M-NAME(MENU-NO)
000440       DISPLAY "Details: " M-DETAIL(MENU-NO)
000450*---------------------------------------------------------------
000460     END-PERFORM.
```

```
000470     EXIT PROGRAM.
000480 END PROGRAM SAMPLE.
```

H.33 ブール式

プログラムをコーディングしていると、しばしばブール項目による論理演算を利用したい局面があります。COBOLでは、ブール式により実現しています。

例

```
000010 @OPTIONS MAIN
000020*---------------------------------------------------------------
000030* ブール項目の演算には、ブール式を利用します。
000040*---------------------------------------------------------------
000050 IDENTIFICATION   DIVISION.
000060 PROGRAM-ID.      SAMPLE.
000070 DATA             DIVISION.
000080  WORKING-STORAGE SECTION.
000090  01 IN-DATA      PIC S9(4) COMP-5.
000100  01 CNT          PIC S9(4) COMP-5.
000110  01 .
000120    02 RESULT       PIC 1(12) BIT VALUE ALL B"0".
000130    02 RES-BIT      REDEFINES RESULT OCCURS 12 PIC 1(1) BIT.
000140  CONSTANT        SECTION.
000150  01 ELM-TBL.
000160    02              PIC X(10) VALUE "USA".
000170    02              PIC X(10) VALUE "Korea".
000180    02              PIC X(10) VALUE "Germany".
000190    02              PIC X(10) VALUE "Russia".
000200    02              PIC X(10) VALUE "England".
000210    02              PIC X(10) VALUE "Japan".
000220    02              PIC X(10) VALUE "Spain".
000230    02              PIC X(10) VALUE "France".
000240    02              PIC X(10) VALUE "Kenya".
000250    02              PIC X(10) VALUE "China".
000260    02              PIC X(10) VALUE "Brazil".
000270    02              PIC X(10) VALUE "Italy".
000280  01              REDEFINES ELM-TBL.
000290    02 ELM-NAME     PIC X(10)  OCCURS 12.
000300*---------------------------------------------------------------
000310* 各国のカテゴリ情報をブール列で定義しておきます。
000320* UNSC: 国連の常任理事国に1を設定。
000330* NATO: NATO加盟国に1を設定。
000340*---------------------------------------------------------------
000350  01 SUBSET-TBL.
000360    02 UNSC         PIC 1(12) BIT VALUE B"100110010100".
000370    02 NATO         PIC 1(12) BIT VALUE B"101010110001".
000380*---------------------------------------------------------------
000390 PROCEDURE        DIVISION.
000400     DISPLAY "以下の国々をカテゴライズします。".
000410     PERFORM TEST BEFORE
000420             VARYING CNT FROM 1 BY 1 UNTIL CNT > 12
000430       IF CNT = 6 OR 12 THEN
000440         DISPLAY ELM-NAME(CNT)
000450       ELSE
000460         DISPLAY ELM-NAME(CNT) WITH NO ADVANCING
000470       END-IF
000480     END-PERFORM.
000490     DISPLAY " ".
000500     DISPLAY "＜カテゴリ＞".
```

```
000510     DISPLAY "  国連の常任理事国 :1".
000520     DISPLAY "  ＮＡＴＯ加盟国   :2".
000530     DISPLAY "  １かつ２の国     :3".
000540     DISPLAY "  いずれでもない国 :4".
000550     DISPLAY  " ".
000560     DISPLAY "カテゴリを選択してください。>> " WITH NO ADVANCING.
000570     ACCEPT  IN-DATA.
000580*-----------------------------------------------------------------
000590* ANDやORなどのブール演算子により、選択された条件に合う国を求め
000600* ます。このブール演算子を使用した式をブール式と呼びます。
000610*-----------------------------------------------------------------
000620     EVALUATE IN-DATA
000630     WHEN 1  COMPUTE RESULT = UNSC
000640     WHEN 2  COMPUTE RESULT = NATO
000650     WHEN 3  COMPUTE RESULT = UNSC  AND  NATO
000660     WHEN 4  COMPUTE RESULT = NOT (UNSC  OR  NATO)
000670     END-EVALUATE.
000680*-----------------------------------------------------------------
000690     PERFORM TEST BEFORE
000700             VARYING CNT FROM 1 BY 1 UNTIL CNT > 12
000710       IF RES-BIT(CNT) =  B"1" THEN
000720         DISPLAY ELM-NAME(CNT)
000730       END-IF
000740     END-PERFORM.
000750 END PROGRAM SAMPLE.
```

H.34 字類条件

データ項目の内容を、たとえば「数字だけで構成されているか」を検査する場合、字類条件を利用します。

以下のサンプルでは、あらかじめ定義された字類(英字検査および数字検査)を利用していますが、特殊名段落のCLASS句で字類を定義することにより、任意の字類で検査することも可能です。

例

```
000010 @OPTIONS MAIN
000020*-----------------------------------------------------------------
000030*  字類条件により、文字データの種別を検査できます。
000040*-----------------------------------------------------------------
000050 IDENTIFICATION   DIVISION.
000060 PROGRAM-ID.      SAMPLE.
000070 DATA             DIVISION.
000080  WORKING-STORAGE SECTION.
000090  01 名前         PIC X(20).
000100  01 生年月日     PIC X(8).
000110 PROCEDURE        DIVISION.
000120     DISPLAY "名前を英字で入力してください。>> " WITH NO ADVANCING.
000130     ACCEPT 名前 FROM CONSOLE.
000140*-----------------------------------------------------------------
000150* ALPHABETIC条件は、英字検査に利用します。
000160*-----------------------------------------------------------------
000170     IF 名前 IS NOT ALPHABETIC THEN
000180       DISPLAY "入力データに誤りがあります。"
000190       EXIT PROGRAM
000200     END-IF.
000210*-----------------------------------------------------------------
000220     DISPLAY "生年月日を入力してください。例:19690123 >> " WITH NO ADVANCING.
000230     ACCEPT 生年月日 FROM CONSOLE.
000240*-----------------------------------------------------------------
000250* NUMERIC条件は、数字検査に利用します。
000260*-----------------------------------------------------------------
```

```
000270     IF 生年月日 IS NOT NUMERIC THEN
000280       DISPLAY "入力データに誤りがあります。"
000290       EXIT PROGRAM
000300     END-IF.
000310*-----------------------------------------------------------------
000320     DISPLAY " ".
000330     DISPLAY "名前     : " 名前.
000340     DISPLAY "生年月日: " 生年月日.
000350 END PROGRAM SAMPLE.
```

H.35 組合せ比較条件の略記法

組合せ比較条件(複数の条件式)において、左辺または右辺のいずれか片方が同じ場合、略記することができます。略記によって、ソースが読みやすくなります。

例

```
000010 @OPTIONS MAIN
000020*-----------------------------------------------------------------
000030* 条件によっては、組合せ比較条件を略記することができます。
000040*-----------------------------------------------------------------
000050 IDENTIFICATION   DIVISION.
000060 PROGRAM-ID.      SAMPLE.
000070 DATA             DIVISION.
000080  WORKING-STORAGE SECTION.
000090  01 IN-DATA      PIC 9(1).
000100  01 COUNTER      PIC 9(1).
000110  CONSTANT        SECTION.
000120  01 SPORTS-DATA.
000130    02              PIC N(5) VALUE NC"マラソン".
000140    02              PIC N(5) VALUE NC"野球".
000150    02              PIC N(5) VALUE NC"テニス".
000160    02              PIC N(5) VALUE NC"スキー".
000170    02              PIC N(5) VALUE NC"柔道".
000180    02              PIC N(5) VALUE NC"サッカー".
000190  01               REDEFINES SPORTS-DATA.
000200    02 SPORTS       OCCURS 6 PIC N(5).
000210 PROCEDURE         DIVISION.
000220     PERFORM TEST BEFORE
000230             VARYING COUNTER FROM 1 BY 1 UNTIL COUNTER > 6
000240       DISPLAY COUNTER "." SPORTS(COUNTER) WITH NO ADVANCING
000250     END-PERFORM.
000260     DISPLAY " ".
000270     DISPLAY " ".
000280     DISPLAY "ボールを使うスポーツは何番でしょう。 >> " WITH NO ADVANCING.
000290     ACCEPT  IN-DATA.
000300     DISPLAY " ".
000310*-----------------------------------------------------------------
000320* 以下のように、左辺が同一で右辺が変化する複数の条件式は、略記
000330* で表現することができます。
000340*
000350* 略記を使用しない場合、以下のように表記しなければなりません。
000360* IF IN-DATA = 2 OR
000370*    IN-DATA = 3 OR
000380*    IN-DATA = 6 THEN
000390*-----------------------------------------------------------------
000400     IF IN-DATA = 2 OR 3 OR 6 THEN
000410       DISPLAY "正解です。"
000420     ELSE
000430       DISPLAY "違います。"
```

```
000440     END-IF.
000450*----------------------------------------------------------------
000460 END PROGRAM SAMPLE.
```

H.36 COMPUTE文

ROUNDED指定

通常、COBOLの算術式では、演算結果を格納する際、データ項目の桁数に従って切捨てが行われますが、ROUNDEDを指定することにより、四捨五入で結果を求めることができます。

例

```
000010 @OPTIONS MAIN
000020*----------------------------------------------------------------
000030* ROUNDED指定により、演算結果を四捨五入で求めることができます。
000040*----------------------------------------------------------------
000050 IDENTIFICATION   DIVISION.
000060 PROGRAM-ID.      SAMPLE.
000070 DATA             DIVISION.
000080  WORKING-STORAGE SECTION.
000090  01 入力         PIC S9(4) VALUE ZERO.
000100  01 合計         PIC S9(8) VALUE ZERO.
000110  01 平均値       PIC S9(4).
000120  01 カウンタ     PIC  9(4) BINARY.
000130 PROCEDURE        DIVISION.
000140     DISPLAY "入力データ（最大４桁）の平均値を四捨五入で求めます。".
000150     PERFORM TEST AFTER
000160             VARYING カウンタ FROM 1 BY 1 UNTIL 入力 = 0
000170       DISPLAY "数値を入力してください(0で終了)。 >> " WITH NO ADVANCING
000180       ACCEPT 入力
000190       COMPUTE 合計 = 合計 + 入力
000200     END-PERFORM.
000210     IF カウンタ > 1 THEN
000220*----------------------------------------------------------------
000230* ROUNDED指定により、結果は四捨五入されます。
000240* なお、ROUNDED指定がない場合、切り捨てとなります。
000250*----------------------------------------------------------------
000260       COMPUTE 平均値 ROUNDED = 合計 / (カウンタ - 1)
000270*----------------------------------------------------------------
000280       DISPLAY " "
000290       DISPLAY "平均値は、" 平均値 " です。"
000300     END-IF.
000310 END PROGRAM SAMPLE.
```

ON SIZE ERROR指定

ON SIZE ERROR指定により、演算結果を格納するデータ項目が桁あふれを起こした場合の処理を定義することができます。COBOLでは、桁あふれが発生しても処理を続行するため、実行時ループや処理結果異常の原因となることがあります。ON SIZE ERROR指定によって、明に桁あふれ時の動作を規定することによって、これらの障害を防ぐことができます。

例

```
000010 @OPTIONS MAIN
000020*----------------------------------------------------------------
000030* ON SIZE ERROR指定により、桁あふれ時の処理を定義できます。
000040*----------------------------------------------------------------
000050 IDENTIFICATION   DIVISION.
```

```
000060 PROGRAM-ID.       SAMPLE.
000070 DATA              DIVISION.
000080  WORKING-STORAGE SECTION.
000090  01 入力          PIC S9(4) VALUE ZERO.
000100  01 合計          PIC S9(4) VALUE ZERO.
000110  01 平均値        PIC S9(4).
000120  01 カウンタ      PIC  9(4) BINARY.
000130 PROCEDURE         DIVISION.
000140     DISPLAY "入力データ（最大４桁）の平均値を求めます。".
000150     PERFORM TEST AFTER
000160             VARYING カウンタ FROM 1 BY 1 UNTIL 入力 = 0
000170       DISPLAY "数値を入力してください(0で終了)。 >> " WITH NO ADVANCING
000180       ACCEPT 入力
000190*----------------------------------------------------------------
000200* 数値の合計を保持する項目が桁あふれを起こした場合、PERFORM文を
000210* 抜けるようロジックを組み込みます。
000220*----------------------------------------------------------------
000230       COMPUTE 合計 = 合計 + 入力
000240         ON SIZE ERROR DISPLAY "中間データが許容範囲を超えました。"
000250                       MOVE ZERO TO カウンタ
000260                       EXIT PERFORM
000270       END-COMPUTE
000280*----------------------------------------------------------------
000290     END-PERFORM.
000300     IF カウンタ > 1 THEN
000310       COMPUTE 平均値 = 合計 / (カウンタ - 1)
000320       DISPLAY " "
000330       DISPLAY "平均値は、" 平均値 " です。"
000340     END-IF.
000350 END PROGRAM SAMPLE.
```

H.37 INITIALIZE文

データの初期化にはINITIALIZE文を利用します。とくに集団項目に従属する項目をすべて初期化したいような場合に便利です。

例

```
000010 @OPTIONS MAIN
000020*----------------------------------------------------------------
000030* データ項目の初期化にはINITIALIZE文を利用します。
000040*----------------------------------------------------------------
000050 IDENTIFICATION   DIVISION.
000060 PROGRAM-ID.       SAMPLE.
000070 DATA              DIVISION.
000080  WORKING-STORAGE SECTION.
000090  01 従業員データ.
000100    02 個人情報     OCCURS 5 TIMES.
000110      03 番号         PIC 9(8).
000120      03             PIC X(1) VALUE ":".
000130      03 名前         PIC N(8).
000140      03             PIC X(1) VALUE ":".
000150      03 所属         PIC N(8).
000160  01 カウンタ      PIC S9(4) BINARY.
000170  CONSTANT        SECTION.
000180  01 ヘッダ        PIC X(32) VALUE "<番号>   <名前>           <住所>".
000190 PROCEDURE         DIVISION.
000200*----------------------------------------------------------------
000210* INITIALIZE文に集団項目を指定すると、従属する項目の属性に合わせ
000220* て適切な初期値が設定されます。
```

```
000230*-----------------------------------------------------------------
000240      INITIALIZE 従業員データ.
000250*-----------------------------------------------------------------
000260      PERFORM 表示処理.
000270*-----------------------------------------------------------------
000280* また、初期値を任意に指定することもできます。
000290*-----------------------------------------------------------------
000300      INITIALIZE 従業員データ REPLACING NUMERIC  DATA BY 99999999
000310                                        NATIONAL DATA BY ALL NC"ー".
000320      PERFORM 表示処理.
000330      EXIT PROGRAM.
000340**
000350 表示処理          SECTION.
000360      DISPLAY " ".
000370      DISPLAY ヘッダ.
000380      PERFORM TEST BEFORE
000390              VARYING カウンタ FROM 1 BY 1 UNTIL カウンタ > 5
000400        DISPLAY 個人情報(カウンタ)
000410      END-PERFORM.
000420 END PROGRAM SAMPLE.
```

H.38 INSPECT文

データ項目中の文字や文字列の出現回数をカウントしたい場合、INSPECT文(書き方1)を利用します。

例

```
000010 @OPTIONS MAIN
000020*-----------------------------------------------------------------
000030* INSPECT文(書き方1)を利用し、文字列の出現回数を求めます。
000040*-----------------------------------------------------------------
000050 IDENTIFICATION   DIVISION.
000060 PROGRAM-ID.      SAMPLE.
000070 DATA             DIVISION.
000080  WORKING-STORAGE SECTION.
000090  01 男性数       PIC 9(4) VALUE ZERO.
000100  01 女性数       PIC 9(4) VALUE ZERO.
000110  CONSTANT        SECTION.
000120  01 参加者名     PIC X(50)
000130                  VALUE "Mr.Brown Mrs.Brown Mr.Jones Mrs.Margaret Mr.Smith".
000140 PROCEDURE        DIVISION.
000150      DISPLAY "旅行参加者名: " 参加者名.
000160      DISPLAY " ".
000170*-----------------------------------------------------------------
000180* INSPECT文を文字列の検索に利用しています。
000190*-----------------------------------------------------------------
000200      INSPECT 参加者名 TALLYING 男性数 FOR ALL "Mr.".
000210      DISPLAY "男性は、" 男性数 "名です。".
000220      INSPECT 参加者名 TALLYING 女性数 FOR ALL "Mrs.".
000230      DISPLAY "女性は、" 女性数 "名です。".
000240*-----------------------------------------------------------------
000250 END PROGRAM SAMPLE.
```

さらに、検索した文字や文字列を他の文字や文字列で置き換える(書き方2)こともできます。

例

```
000010 @OPTIONS MAIN
000020*-----------------------------------------------------------------
```

```
000030* INSPECT文(書き方2)を利用し、文字列を置換します。
000040*-----------------------------------------------------------------
000050 IDENTIFICATION   DIVISION.
000060 PROGRAM-ID.      SAMPLE.
000070 DATA             DIVISION.
000080  WORKING-STORAGE SECTION.
000090  01 CR-DATE.
000100    02 CR-YEAR      PIC 9(4).
000110    02 CR-MON       PIC 9(2).
000120    02 CR-DAY       PIC 9(2).
000130  01 COPYRIGHT    PIC X(60)
000140                   VALUE "Copyright yyyy FUJITSU LIMITED".
000150 PROCEDURE        DIVISION.
000160     MOVE FUNCTION CURRENT-DATE TO CR-DATE.
000170*-----------------------------------------------------------------
000180* 文字列から目的の語を検索し、指定された語で置き換えます。
000190*-----------------------------------------------------------------
000200     INSPECT COPYRIGHT REPLACING ALL "yyyy" BY CR-YEAR.
000210*-----------------------------------------------------------------
000220     DISPLAY COPYRIGHT.
000230 END PROGRAM SAMPLE.
```

H.39 MOVE文

CORRESPONDING指定により、集団項目転記時、同名のデータ項目だけに作用する転記を記述することができます。

例

```
000010 @OPTIONS MAIN
000020*-----------------------------------------------------------------
000030* CORRESPONDING指定によって、対応するデータ項目を処理対象とする
000040* 文を実行できます。
000050*-----------------------------------------------------------------
000060 IDENTIFICATION   DIVISION.
000070 PROGRAM-ID.      SAMPLE.
000080 DATA             DIVISION.
000090  WORKING-STORAGE SECTION.
000100  01 編集表示.
000110    02 年           PIC 9(4).
000120    02             PIC X(1) VALUE "/".
000130    02 月           PIC 9(2).
000140    02             PIC X(1) VALUE "/".
000150    02 日           PIC 9(2).
000170    02             PIC X(1) VALUE SPACE.
000180    02 時           PIC 9(2).
000190    02             PIC X(1) VALUE ".".
000200    02 分           PIC 9(2).
000210    02             PIC X(1) VALUE ".".
000220    02 秒           PIC 9(2).
000240  01 現時刻.
000250    02  年          PIC 9(4).
000260    02  月          PIC 9(2).
000270    02  日          PIC 9(2).
000280    02  時          PIC 9(2).
000290    02  分          PIC 9(2).
000300    02  秒          PIC 9(2).
000310 PROCEDURE        DIVISION.
000320      MOVE FUNCTION CURRENT-DATE TO 現時刻.
000330*-----------------------------------------------------------------
000340* CORRESPONDING指定により、対応する(同名の)データ項目へ転記します。
```

```
000350*-----------------------------------------------------------------
000360       MOVE CORRESPONDING 現時刻 TO 編集表示.
000370*-----------------------------------------------------------------
000380       DISPLAY "現在の時刻は、" 編集表示 " です。".
000390 END PROGRAM SAMPLE.
```

H.40 SEARCH文

表(OCCURS句によって配列宣言されたデータの集合)から、特定の要素を検索したい場合、SEARCH文を利用します。

SEARCH文には、逐次表引き(書き方1)と非逐次表引き(書き方2)があります。

SEARCH文（書き方1）

例

```
000010 @OPTIONS MAIN
000020*-----------------------------------------------------------------
000030* SEARCH文(書き方1)を利用して、表から特定の要素を取り出します。
000040*-----------------------------------------------------------------
000050 IDENTIFICATION   DIVISION.
000060 PROGRAM-ID.      SAMPLE.
000070 DATA             DIVISION.
000080  WORKING-STORAGE SECTION.
000090  01 GET-GOODS    PIC X(10).
000100  01 GET-NUM      PIC 9(4) BINARY.
000110  01 TOTAL        PIC 9(4) BINARY VALUE ZERO.
000120  01 COUNTER      PIC 9(1).
000130  01 PRICE-ED     PIC ¥¥¥¥9.
000140  01 TOTAL-ED     PIC ZZZZ9.
000150  CONSTANT        SECTION.
000160  01 GOODS-DATA.
000170    02              PIC X(10) VALUE "NOTEBOOK".
000180    02              PIC 9(4)  VALUE 150.
000190    02              PIC X(10) VALUE "PENCIL".
000200    02              PIC 9(4)  VALUE 30.
000210    02              PIC X(10) VALUE "ERASER".
000220    02              PIC 9(4)  VALUE 50.
000230    02              PIC X(10) VALUE "RULER".
000240    02              PIC 9(4)  VALUE 100.
000250    02              PIC X(10) VALUE "CUTTER".
000260    02              PIC 9(4)  VALUE 200.
000270*-----------------------------------------------------------------
000280* SEARCH文で利用する表には、指標名を指定しておきます。
000290*-----------------------------------------------------------------
000300  01              REDEFINES GOODS-DATA.
000310    02 GOODS        OCCURS 5 TIMES INDEXED BY IX.
000320      03 NAME         PIC X(10).
000330      03 PRICE        PIC 9(4).
000340*-----------------------------------------------------------------
000350 PROCEDURE        DIVISION.
000360     PERFORM TEST BEFORE
000370             VARYING COUNTER FROM 1 BY 1 UNTIL COUNTER > 5
000380       MOVE PRICE(COUNTER) TO PRICE-ED
000390       DISPLAY COUNTER "." NAME(COUNTER) PRICE-ED
000400     END-PERFORM.
000410     DISPLAY " ".
000420     DISPLAY "何を買いますか ? 商品名 >> " WITH NO ADVANCING.
000430     ACCEPT GET-GOODS FROM CONSOLE.
000440     DISPLAY "何個必要ですか ?        >> " WITH NO ADVANCING.
000450     ACCEPT GET-NUM.
000460*-----------------------------------------------------------------
```

```
000470* SEARCH文には、検索該当時に実行する処理を記述します。
000480* 検索処理は指標により逐次実行されるため、指標に値を設定して検索
000490* 開始位置を指示します。また、検索処理は検索対象が見つかった時点
000500* で終了し、指標にはその時点の値が設定されています。このため、続
000510* けてSEARCH文を記述すると、続きから検索できます。
000520*-----------------------------------------------------------------
000530     SET IX TO 1.
000540     SEARCH GOODS
000550       WHEN NAME(IX) = GET-GOODS
000560         MOVE PRICE(IX) TO TOTAL
000570     END-SEARCH.
000580*-----------------------------------------------------------------
000590     DISPLAY " ".
000600     IF TOTAL NOT = ZERO THEN
000610       COMPUTE TOTAL = TOTAL * GET-NUM
000620       MOVE TOTAL TO TOTAL-ED
000630       DISPLAY "合計金額は、" TOTAL-ED "円です。"
000640     ELSE
000650       DISPLAY "入力データに誤りがあります。"
000660     END-IF.
000670 END PROGRAM SAMPLE.
```

SEARCH文(書き方2)

例

```
000010 @OPTIONS MAIN
000020*-----------------------------------------------------------------
000030* SEARCH文(書き方2)により、非逐次表引きも可能です。
000040*-----------------------------------------------------------------
000050 IDENTIFICATION   DIVISION.
000060 PROGRAM-ID.      SAMPLE.
000070 DATA             DIVISION.
000080  WORKING-STORAGE SECTION.
000090  01 GET-GOODS    PIC X(10).
000100  01 GET-NUM      PIC 9(4) BINARY.
000110  01 TOTAL        PIC 9(4) BINARY VALUE ZERO.
000120  01 COUNTER      PIC 9(1).
000130  01 PRICE-ED     PIC ¥¥¥¥9.
000140  01 TOTAL-ED     PIC ZZZZ9.
000150  CONSTANT        SECTION.
000160  01 GOODS-DATA.
000170    02              PIC X(10) VALUE "CUTTER".
000180    02              PIC 9(4)  VALUE 200.
000190    02              PIC X(10) VALUE "ERASER".
000200    02              PIC 9(4)  VALUE 50.
000210    02              PIC X(10) VALUE "NOTEBOOK".
000220    02              PIC 9(4)  VALUE 150.
000230    02              PIC X(10) VALUE "PENCIL".
000240    02              PIC 9(4)  VALUE 30.
000250    02              PIC X(10) VALUE "RULER".
000260    02              PIC 9(4)  VALUE 100.
000270*-----------------------------------------------------------------
000280* SEARCH文(書き方2)で利用する表には、検索キーと指標名を指定します。
000290* なお、検索キーとなるデータは昇順(ASCENDING)または降順(DESCENDING)
000300* に並んでいる必要があります。
000310*-----------------------------------------------------------------
000320  01             REDEFINES GOODS-DATA.
000330    02 GOODS       OCCURS 5 TIMES
000340                   ASCENDING KEY IS NAME INDEXED BY IX.
000350      03 NAME        PIC X(10).
000360      03 PRICE       PIC 9(4).
```

```
000370*-----------------------------------------------------------------
000380 PROCEDURE        DIVISION.
000390     PERFORM TEST BEFORE
000400             VARYING COUNTER FROM 1 BY 1 UNTIL COUNTER > 5
000410       MOVE PRICE(COUNTER) TO PRICE-ED
000420       DISPLAY COUNTER "." NAME(COUNTER) PRICE-ED
000430     END-PERFORM.
000440     DISPLAY " ".
000450     DISPLAY "何を買いますか ? 商品名 >> " WITH NO ADVANCING.
000460     ACCEPT GET-GOODS FROM CONSOLE.
000470     DISPLAY "何個必要ですか ?        >> " WITH NO ADVANCING.
000480     ACCEPT GET-NUM.
000490*-----------------------------------------------------------------
000500* SEARCH文(書き方2)の場合、指標の初期値は無視され、検索処理におけ
000510* る指標の値はコンパイラの定めた表引き方法に従って変化します。
000520* 局面に合わせて、書き方1と使い分けてください。
000530*-----------------------------------------------------------------
000540     SEARCH ALL GOODS
000550       WHEN NAME(IX) = GET-GOODS
000560         MOVE PRICE(IX) TO TOTAL
000570     END-SEARCH.
000580*-----------------------------------------------------------------
000590     DISPLAY " ".
000600     IF TOTAL NOT = ZERO THEN
000610       COMPUTE TOTAL = TOTAL * GET-NUM
000620       MOVE TOTAL TO TOTAL-ED
000630       DISPLAY "合計金額は、" TOTAL-ED "円です。"
000640     ELSE
000650       DISPLAY "入力データに誤りがあります。"
000660     END-IF.
000670 END PROGRAM SAMPLE.
```

H.41 STRING文

いくつかの文字データを連結したい場合、STRING文を利用します。

任意の区切り文字までの連結や、任意の長さでの連結が可能です。

例

```
000010 @OPTIONS MAIN
000020*-----------------------------------------------------------------
000030* 文字データの連結にはSTRING文を利用します。
000040*-----------------------------------------------------------------
000050 IDENTIFICATION   DIVISION.
000060 PROGRAM-ID.      SAMPLE.
000070 DATA             DIVISION.
000080  WORKING-STORAGE SECTION.
000090  01 LISTING      PIC X(30) VALUE SPACE.
000100  01 COUNTER      PIC 9(4)  BINARY.
000110  CONSTANT        SECTION.
000120  01 SHOPPING-LIST.
000130    02              PIC X(15) VALUE "Apple,3,¥100".
000140    02              PIC X(15) VALUE "Bread,1,¥200".
000150    02              PIC X(15) VALUE "Eggs,1,¥200".
000160    02              PIC X(15) VALUE "Tomato,5,¥50".
000170    02              PIC X(15) VALUE "Milk,1,¥200".
000180  01              REDEFINES SHOPPING-LIST.
000190    02 ELM          OCCURS 5 TIMES PIC X(15).
000200 PROCEDURE        DIVISION.
000210     PERFORM TEST BEFORE
```

```
000220             VARYING COUNTER FROM 1 BY 1 UNTIL COUNTER > 5
000230*-----------------------------------------------------------------
000240* STRING文には連結するデータを指定します。
000250* この例では、買い物リストから品名だけを取り出し、一覧表示します。
000260*-----------------------------------------------------------------
000270       STRING LISTING      DELIMITED BY SPACE
000280              ELM(COUNTER) DELIMITED BY ","
000290              ","          DELIMITED BY SIZE  INTO LISTING
000300*-----------------------------------------------------------------
000310     END-PERFORM.
000320     DISPLAY "Shopping-list: " LISTING.
000330 END PROGRAM SAMPLE.
```

H.42 UNSTRING文

文字列データを任意の区切り文字で分割し、別々のデータ項目へ格納したい場合、UNSTRING文を利用します。

例

```
000010 @OPTIONS MAIN
000020*-----------------------------------------------------------------
000030* 文字列を任意の区切り文字で分割したい場合、UNSTRING文を利用します。
000040*-----------------------------------------------------------------
000050 IDENTIFICATION   DIVISION.
000060 PROGRAM-ID.      SAMPLE.
000070 DATA             DIVISION.
000080  WORKING-STORAGE SECTION.
000090  01 .
000100    02 WORD         OCCURS 10 TIMES PIC X(10).
000110  01 COUNTER      PIC 9(4) BINARY.
000120  01 SRCH-POS     PIC 9(4) BINARY VALUE 1.
000130  01 WORD-NUM     PIC 9(4) BINARY VALUE 0.
000140  CONSTANT        SECTION.
000150  77 STR-DATA     PIC X(25) VALUE "Are you hungry? Yes,I am.".
000160 PROCEDURE        DIVISION.
000170     DISPLAY "String: " STR-DATA.
000180     DISPLAY " ".
000190     PERFORM TEST BEFORE
000200             VARYING COUNTER FROM 1 BY 1 UNTIL COUNTER > 10
000210*-----------------------------------------------------------------
000220* UNSTRING文には、
000230* - 区切り文字(DELIMITED BY)
000240* - 格納域(INTO)
000250* - 検索開始位置(WITH POINTER)
000260* - 分割した個数の格納域(TALLYING IN)
000270* などを指定できます。
000280*-----------------------------------------------------------------
000290       UNSTRING STR-DATA
000300                DELIMITED BY ALL SPACE OR "," OR "."
000310                INTO WORD(COUNTER)
000320                WITH POINTER SRCH-POS
000330                TALLYING IN  WORD-NUM
000340       END-UNSTRING
000350*-----------------------------------------------------------------
000360     END-PERFORM.
000370     PERFORM TEST BEFORE
000380             VARYING COUNTER FROM 1 BY 1 UNTIL COUNTER > WORD-NUM
000390       DISPLAY COUNTER ". " WORD(COUNTER)
000400     END-PERFORM.
000410 END PROGRAM SAMPLE.
```

H.43 USE文

例外(ここでは入出力エラーを例に説明)が発生した場合に実行する手続きを記述したい場合、USE文を利用します。エラーリカバリ処理やエラー通知処理などを定義することができます。

例

```
000010 @OPTIONS MAIN
000020*---------------------------------------------------------------------
000030* 入出力エラーを捕まえたい場合は、USE文を記述します。
000040*---------------------------------------------------------------------
000050 IDENTIFICATION   DIVISION.
000060 PROGRAM-ID.      SAMPLE.
000070 ENVIRONMENT      DIVISION.
000080  INPUT-OUTPUT    SECTION.
000090  FILE-CONTROL.
000100       SELECT TMP-FILE ASSIGN TO "C:¥NONFILE"
000110                       FILE STATUS IS F-STAT.
000120 DATA             DIVISION.
000130  FILE            SECTION.
000140  FD TMP-FILE.
000150  01 TMP-REC      PIC X(80).
000160  WORKING-STORAGE SECTION.
000170  01 F-STAT.
000180    02 STAT-1       PIC X(1).
000190    02 STAT-2       PIC X(1).
000200 PROCEDURE        DIVISION.
000210*---------------------------------------------------------------------
000220* 入出力誤り手続き(USE文)は、宣言節に記述します。
000230*---------------------------------------------------------------------
000240 DECLARATIVES.
000250*---------------------------------------------------------------------
000260* ここでは、TMP-FILEに対する入出力誤り手続きを定義しています。
000270* この手続きが存在する場合、TMP-FILEで何らかの入出力エラーがあると、
000280* ランタイムシステムのエラーを出力せずに、この手続きの先頭へ制御が
000290* 移ります。
000300*---------------------------------------------------------------------
000310 IO-ERR           SECTION.
000320     USE AFTER EXCEPTION PROCEDURE TMP-FILE.
000330     EVALUATE F-STAT
000340     WHEN "35"
000350       DISPLAY "ファイルが存在しません。"
000360     WHEN "47"
000370       DISPLAY "未オープンファイルに対してREAD文が実行されました。"
000380     WHEN "42"
000390       DISPLAY "未オープンファイルに対してCLOSE文が実行されました。"
000400     WHEN OTHER
000410       DISPLAY "その他の入出力エラーが発生しました。"
000420     END-EVALUATE.
000430     DISPLAY ">>>> 処理を継続します。".
000440 END DECLARATIVES.
000450*---------------------------------------------------------------------
000460     OPEN INPUT TMP-FILE.
000470     READ TMP-FILE AT END GO TO P-EXIT.
000480     CLOSE TMP-FILE.
000490 P-EXIT.
000500     EXIT PROGRAM.
000510 END PROGRAM SAMPLE.
```

H.44 SIN関数

SIN関数は、引数で指定した角度に対する正弦の近似値を返却します。

このサンプルは[Win32]で動作します。

例

```
000010 @OPTIONS MAIN
000020*---------------------------------------------------------------
000030* 正弦(SIN)、余弦(COS)、正接(TAN)のグラフを描画します。
000040*---------------------------------------------------------------
000050 IDENTIFICATION   DIVISION.
000060 PROGRAM-ID.      SAMPLE.
000070 DATA             DIVISION.
000080  WORKING-STORAGE SECTION.
000090  01 PI           PIC S9(3)V9(15) VALUE 3.141592653589793.
000100  01 VAL          PIC S9(3)V9(15).
000110  01 LINE-POS     PIC  9(2).
000120  01 COL-POS      PIC  9(2).
000130  01 GRAPH-CODE   PIC  X(1).
000140  01 COUNTER      PIC  9(4) BINARY.
000150  01 S-COUNTER    PIC S9(4) BINARY.
000160 PROCEDURE        DIVISION.
000170     DISPLAY "どのグラフを描きますか？(SIN:S, COS:C, TAN:T) >> "
000180                                                 WITH NO ADVANCING.
000190     ACCEPT GRAPH-CODE.
000200     PERFORM TEST BEFORE
000210             VARYING COUNTER FROM 1 BY 1 UNTIL COUNTER > 80
000220       DISPLAY "-" AT LINE 13 COLUMN COUNTER
000230     END-PERFORM.
000240     PERFORM TEST BEFORE
000250             VARYING COUNTER FROM 1 BY 1 UNTIL COUNTER = 26
000260       DISPLAY "|" AT LINE COUNTER COLUMN 40
000270     END-PERFORM.
000280     DISPLAY "+" AT LINE 13 COLUMN 40.
000290*---------------------------------------------------------------
000300* 正弦(SIN)のグラフを描画します。
000310*---------------------------------------------------------------
000320     EVALUATE GRAPH-CODE
000330     WHEN "S"
000340       PERFORM TEST BEFORE
000350               VARYING S-COUNTER FROM -39 BY 1 UNTIL S-COUNTER = 40
000360         COMPUTE VAL = 12 * (FUNCTION SIN(PI / 39 * S-COUNTER))
000370         COMPUTE LINE-POS ROUNDED = 13 - VAL
000380         COMPUTE COL-POS = 40 + S-COUNTER
000390         DISPLAY "*" AT LINE LINE-POS COLUMN COL-POS
000400       END-PERFORM
000410*---------------------------------------------------------------
000420* 余弦(COS)のグラフを描画します。
000430*---------------------------------------------------------------
000440     WHEN "C"
000450       PERFORM TEST BEFORE
000460               VARYING S-COUNTER FROM -39 BY 1 UNTIL S-COUNTER = 40
000470         COMPUTE VAL = 12 * (FUNCTION COS(PI / 39 * S-COUNTER))
000480         COMPUTE LINE-POS ROUNDED = 13 - VAL
000490         COMPUTE COL-POS = 40 + S-COUNTER
000500         DISPLAY "*" AT LINE LINE-POS COLUMN COL-POS
000510       END-PERFORM
000520*---------------------------------------------------------------
000530* 正接(TAN)のグラフを描画します。
000540*---------------------------------------------------------------
```

```
000550     WHEN "T"
000560       PERFORM TEST BEFORE
000570               VARYING S-COUNTER FROM -38 BY 1 UNTIL S-COUNTER = 39
000580         COMPUTE VAL = 0.5 * (FUNCTION TAN(PI / 2 / 39 * S-COUNTER))
000590         COMPUTE LINE-POS ROUNDED = 13 - VAL
000600         COMPUTE COL-POS = 40 + S-COUNTER
000610         DISPLAY "*" AT LINE LINE-POS COLUMN COL-POS
000620       END-PERFORM
000630     END-EVALUATE.
000640 END PROGRAM SAMPLE.
```

H.45 COS関数

COS関数は、引数で指定した角度に対する余弦の近似値を返却します。

このサンプルは[Win32]で動作します。

例

```
000010 @OPTIONS MAIN
000020*---------------------------------------------------------------
000030* 正弦(SIN)、余弦(COS)、正接(TAN)のグラフを描画します。
000040*---------------------------------------------------------------
000050 IDENTIFICATION   DIVISION.
000060 PROGRAM-ID.      SAMPLE.
000070 DATA             DIVISION.
000080  WORKING-STORAGE SECTION.
000090  01 PI           PIC S9(3)V9(15) VALUE 3.141592653589793.
000100  01 VAL          PIC S9(3)V9(15).
000110  01 LINE-POS     PIC  9(2).
000120  01 COL-POS      PIC  9(2).
000130  01 GRAPH-CODE   PIC  X(1).
000140  01 COUNTER      PIC  9(4) BINARY.
000150  01 S-COUNTER    PIC S9(4) BINARY.
000160 PROCEDURE        DIVISION.
000170     DISPLAY "どのグラフを描きますか？(SIN:S, COS:C, TAN:T) >> "
000180                                              WITH NO ADVANCING.
000190     ACCEPT GRAPH-CODE.
000200     PERFORM TEST BEFORE
000210             VARYING COUNTER FROM 1 BY 1 UNTIL COUNTER > 80
000220       DISPLAY "-" AT LINE 13 COLUMN COUNTER
000230     END-PERFORM.
000240     PERFORM TEST BEFORE
000250             VARYING COUNTER FROM 1 BY 1 UNTIL COUNTER = 26
000260       DISPLAY "|" AT LINE COUNTER COLUMN 40
000270     END-PERFORM.
000280     DISPLAY "+" AT LINE 13 COLUMN 40.
000290*---------------------------------------------------------------
000300* 正弦(SIN)のグラフを描画します。
000310*---------------------------------------------------------------
000320     EVALUATE GRAPH-CODE
000330     WHEN "S"
000340       PERFORM TEST BEFORE
000350               VARYING S-COUNTER FROM -39 BY 1 UNTIL S-COUNTER = 40
000360         COMPUTE VAL = 12 * (FUNCTION SIN(PI / 39 * S-COUNTER))
000370         COMPUTE LINE-POS ROUNDED = 13 - VAL
000380         COMPUTE COL-POS = 40 + S-COUNTER
000390         DISPLAY "*" AT LINE LINE-POS COLUMN COL-POS
000400       END-PERFORM
000410*---------------------------------------------------------------
000420* 余弦(COS)のグラフを描画します。
```

```
000430*-----------------------------------------------------------------
000440     WHEN "C"
000450       PERFORM TEST BEFORE
000460               VARYING S-COUNTER FROM -39 BY 1 UNTIL S-COUNTER = 40
000470         COMPUTE VAL = 12 * (FUNCTION COS(PI / 39 * S-COUNTER))
000480         COMPUTE LINE-POS ROUNDED = 13 - VAL
000490         COMPUTE COL-POS = 40 + S-COUNTER
000500         DISPLAY "*" AT LINE LINE-POS COLUMN COL-POS
000510       END-PERFORM
000520*-----------------------------------------------------------------
000530* 正接(TAN)のグラフを描画します。
000540*-----------------------------------------------------------------
000550     WHEN "T"
000560       PERFORM TEST BEFORE
000570               VARYING S-COUNTER FROM -38 BY 1 UNTIL S-COUNTER = 39
000580         COMPUTE VAL = 0.5 * (FUNCTION TAN(PI / 2 / 39 * S-COUNTER))
000590         COMPUTE LINE-POS ROUNDED = 13 - VAL
000600         COMPUTE COL-POS = 40 + S-COUNTER
000610         DISPLAY "*" AT LINE LINE-POS COLUMN COL-POS
000620       END-PERFORM
000630     END-EVALUATE.
000640 END PROGRAM SAMPLE.
```

H.46 TAN関数

TAN関数は、引数で指定した角度に対する正接の近似値を返却します。

このサンプルは[Win32]で動作します。

例

```
000010 @OPTIONS MAIN
000020*-----------------------------------------------------------------
000030* 正弦(SIN)、余弦(COS)、正接(TAN)のグラフを描画します。
000040*-----------------------------------------------------------------
000050 IDENTIFICATION   DIVISION.
000060 PROGRAM-ID.      SAMPLE.
000070 DATA             DIVISION.
000080  WORKING-STORAGE SECTION.
000090  01 PI           PIC S9(3)V9(15) VALUE 3.141592653589793.
000100  01 VAL          PIC S9(3)V9(15).
000110  01 LINE-POS     PIC  9(2).
000120  01 COL-POS      PIC  9(2).
000130  01 GRAPH-CODE   PIC  X(1).
000140  01 COUNTER      PIC  9(4) BINARY.
000150  01 S-COUNTER    PIC S9(4) BINARY.
000160 PROCEDURE        DIVISION.
000170     DISPLAY "どのグラフを描きますか？(SIN:S, COS:C, TAN:T) >> "
000180                                              WITH NO ADVANCING.
000190     ACCEPT GRAPH-CODE.
000200     PERFORM TEST BEFORE
000210             VARYING COUNTER FROM 1 BY 1 UNTIL COUNTER > 80
000220       DISPLAY "-" AT LINE 13 COLUMN COUNTER
000230     END-PERFORM.
000240     PERFORM TEST BEFORE
000250             VARYING COUNTER FROM 1 BY 1 UNTIL COUNTER = 26
000260       DISPLAY "|" AT LINE COUNTER COLUMN 40
000270     END-PERFORM.
000280     DISPLAY "+" AT LINE 13 COLUMN 40.
000290*-----------------------------------------------------------------
000300* 正弦(SIN)のグラフを描画します。
```

```
000310*-----------------------------------------------------------------
000320     EVALUATE GRAPH-CODE
000330     WHEN "S"
000340       PERFORM TEST BEFORE
000350               VARYING S-COUNTER FROM -39 BY 1 UNTIL S-COUNTER = 40
000360         COMPUTE VAL = 12 * (FUNCTION SIN(PI / 39 * S-COUNTER))
000370         COMPUTE LINE-POS ROUNDED = 13 - VAL
000380         COMPUTE COL-POS = 40 + S-COUNTER
000390         DISPLAY "*" AT LINE LINE-POS COLUMN COL-POS
000400       END-PERFORM
000410*-----------------------------------------------------------------
000420* 余弦(COS)のグラフを描画します。
000430*-----------------------------------------------------------------
000440     WHEN "C"
000450       PERFORM TEST BEFORE
000460               VARYING S-COUNTER FROM -39 BY 1 UNTIL S-COUNTER = 40
000470         COMPUTE VAL = 12 * (FUNCTION COS(PI / 39 * S-COUNTER))
000480         COMPUTE LINE-POS ROUNDED = 13 - VAL
000490         COMPUTE COL-POS = 40 + S-COUNTER
000500         DISPLAY "*" AT LINE LINE-POS COLUMN COL-POS
000510       END-PERFORM
000520*-----------------------------------------------------------------
000530* 正接(TAN)のグラフを描画します。
000540*-----------------------------------------------------------------
000550     WHEN "T"
000560       PERFORM TEST BEFORE
000570               VARYING S-COUNTER FROM -38 BY 1 UNTIL S-COUNTER = 39
000580         COMPUTE VAL = 0.5 * (FUNCTION TAN(PI / 2 / 39 * S-COUNTER))
000590         COMPUTE LINE-POS ROUNDED = 13 - VAL
000600         COMPUTE COL-POS = 40 + S-COUNTER
000610         DISPLAY "*" AT LINE LINE-POS COLUMN COL-POS
000620       END-PERFORM
000630     END-EVALUATE.
000640 END PROGRAM SAMPLE.
```

H.47 ADDR関数

データ項目のアドレスを求める場合、ADDR関数を利用します。

求めたアドレスは、ポインタ修飾やアドレスインタフェースの他言語連携などで使用します。

例

```
000010 @OPTIONS MAIN
000020*-----------------------------------------------------------------
000030* ADDR関数によって求めたポインタと基底場所節に定義した２つの
000040* フォームを使って、作業域にデータを格納します。
000050* また、データ項目長はLENG関数によって求めています。
000060*-----------------------------------------------------------------
000070 IDENTIFICATION   DIVISION.
000080 PROGRAM-ID.      SAMPLE.
000090 DATA             DIVISION.
000100  BASED-STORAGE   SECTION.
000110  01 LONG-FORM.
000120     02 FIRST-NAME-L   PIC X(8).
000130     02 LAST-NAME-L    PIC X(8).
000140     02 AGE-L          PIC 9(2).
000150     02 SEPARATER-L    PIC X(1).
000160     02 NEXT-TOP-L     PIC X(1).
000170  01 SHORT-FORM.
000180     02 LAST-NAME-S    PIC X(8).
```

```
000190    02 AGE-S          PIC 9(2).
000200    02 SEPARATER-S    PIC X(1).
000210    02 NEXT-TOP-S     PIC X(1).
000220  WORKING-STORAGE SECTION.
000230  01 PTR             USAGE POINTER.
000240  01 WORK            PIC X(255).
000250  01 FORMS           PIC 9(1).
000260  01 COUNT-L         PIC 9(2) VALUE 0.
000270  01 COUNT-S         PIC 9(2) VALUE 0.
000280  01 TOTAL-SIZE      PIC 9(3).
000290 PROCEDURE           DIVISION.
000300*-----------------------------------------------------------------
000310* ADDR関数で作業域(データ格納域)の先頭アドレスを求めます。
000320*-----------------------------------------------------------------
000330     MOVE FUNCTION ADDR(WORK) TO PTR.
000340*-----------------------------------------------------------------
000350     PERFORM TEST AFTER
000360             UNTIL FORMS = 0
000370       DISPLAY " "
000380       DISPLAY "どちらのフォームを選択しますか？"
000390       DISPLAY "1: ロングフォーム　(姓 名 年齢)"
000400       DISPLAY "2: ショートフォーム(名 年齢)"
000410       DISPLAY "0: 処理終了                    >> " WITH NO ADVANCING
000420       ACCEPT FORMS
000430       EVALUATE FORMS
000440       WHEN "1"
000450         DISPLAY "姓   >> " WITH NO ADVANCING
000460         ACCEPT  PTR->FIRST-NAME-L FROM CONSOLE
000470         DISPLAY "名   >> " WITH NO ADVANCING
000480         ACCEPT  PTR->LAST-NAME-L FROM CONSOLE
000490         DISPLAY "年齢 >> " WITH NO ADVANCING
000500         ACCEPT  PTR->AGE-L FROM CONSOLE
000510         MOVE    "/" TO PTR->SEPARATER-L
000520         COMPUTE COUNT-L = COUNT-L + 1
000530         MOVE FUNCTION ADDR(PTR->NEXT-TOP-L) TO PTR
000540       WHEN "2"
000550         DISPLAY "名   >> " WITH NO ADVANCING
000560         ACCEPT  PTR->LAST-NAME-S FROM CONSOLE
000570         DISPLAY "年齢 >> " WITH NO ADVANCING
000580         ACCEPT  PTR->AGE-S FROM CONSOLE
000590         MOVE    "/" TO PTR->SEPARATER-S
000600         COMPUTE COUNT-S = COUNT-S + 1
000610         MOVE FUNCTION ADDR(PTR->NEXT-TOP-S) TO PTR
000620       END-EVALUATE
000630     END-PERFORM.
000640*-----------------------------------------------------------------
000650* LENG関数によって、データ項目長を求めます。
000660* LENG関数を利用することによって、項目追加(集団項目長の変化)の
000670* 影響を受けないコーディングが可能になります。
000680*-----------------------------------------------------------------
000690     COMPUTE TOTAL-SIZE = ( FUNCTION LENG(PTR->LONG-FORM)  - 1 ) * COUNT-L
000700                        + ( FUNCTION LENG(PTR->SHORT-FORM) - 1 ) * COUNT-S.
000710*-----------------------------------------------------------------
000720     DISPLAY "DATA           :" WORK.
000730     DISPLAY "TOTAL DATA SIZE:" TOTAL-SIZE.
000740 END PROGRAM SAMPLE.
```

H.48 LENG関数

データ項目の領域長(バイト数)を求める場合、LENG関数を利用します。

似た機能を持つ関数にLENGTH関数がありますが、LENGTH関数の場合、文字位置の個数、つまり、文字数(厳密には文字数という表現は誤りですが)を返却します。

例

```
000010 @OPTIONS MAIN
000020*---------------------------------------------------------------
000030* ADDR関数によって求めたポインタと基底場所節に定義した２つの
000040* フォームを使って、作業域にデータを格納します。
000050* また、データ項目長はLENG関数によって求めています。
000060*---------------------------------------------------------------
000070 IDENTIFICATION   DIVISION.
000080 PROGRAM-ID.      SAMPLE.
000090 DATA             DIVISION.
000100  BASED-STORAGE   SECTION.
000110  01 LONG-FORM.
000120    02 FIRST-NAME-L   PIC X(8).
000130    02 LAST-NAME-L    PIC X(8).
000140    02 AGE-L          PIC 9(2).
000150    02 SEPARATER-L    PIC X(1).
000160    02 NEXT-TOP-L     PIC X(1).
000170  01 SHORT-FORM.
000180    02 LAST-NAME-S    PIC X(8).
000190    02 AGE-S          PIC 9(2).
000200    02 SEPARATER-S    PIC X(1).
000210    02 NEXT-TOP-S     PIC X(1).
000220  WORKING-STORAGE SECTION.
000230  01 PTR            USAGE POINTER.
000240  01 WORK           PIC X(255).
000250  01 FORMS          PIC 9(1).
000260  01 COUNT-L        PIC 9(2) VALUE 0.
000270  01 COUNT-S        PIC 9(2) VALUE 0.
000280  01 TOTAL-SIZE     PIC 9(3).
000290 PROCEDURE          DIVISION.
000300*---------------------------------------------------------------
000310* ADDR関数で作業域(データ格納域)の先頭アドレスを求めます。
000320*---------------------------------------------------------------
000330     MOVE FUNCTION ADDR(WORK) TO PTR.
000340*---------------------------------------------------------------
000350     PERFORM TEST AFTER
000360             UNTIL FORMS = 0
000370       DISPLAY " "
000380       DISPLAY "どちらのフォームを選択しますか？"
000390       DISPLAY "1: ロングフォーム　(姓 名 年齢)"
000400       DISPLAY "2: ショートフォーム(名 年齢)"
000410       DISPLAY "0: 処理終了                    >> " WITH NO ADVANCING
000420       ACCEPT FORMS
000430       EVALUATE FORMS
000440       WHEN "1"
000450         DISPLAY "姓    >> " WITH NO ADVANCING
000460         ACCEPT  PTR->FIRST-NAME-L FROM CONSOLE
000470         DISPLAY "名    >> " WITH NO ADVANCING
000480         ACCEPT  PTR->LAST-NAME-L FROM CONSOLE
000490         DISPLAY "年齢 >> " WITH NO ADVANCING
000500         ACCEPT  PTR->AGE-L FROM CONSOLE
000510         MOVE    "/" TO PTR->SEPARATER-L
000520         COMPUTE COUNT-L = COUNT-L + 1
000530         MOVE FUNCTION ADDR(PTR->NEXT-TOP-L) TO PTR
000540       WHEN "2"
000550         DISPLAY "名    >> " WITH NO ADVANCING
000560         ACCEPT  PTR->LAST-NAME-S FROM CONSOLE
000570         DISPLAY "年齢 >> " WITH NO ADVANCING
```

```
000580          ACCEPT  PTR->AGE-S FROM CONSOLE
000590          MOVE    "/" TO PTR->SEPARATER-S
000600          COMPUTE COUNT-S = COUNT-S + 1
000610          MOVE FUNCTION ADDR(PTR->NEXT-TOP-S) TO PTR
000620        END-EVALUATE
000630      END-PERFORM.
000640*-----------------------------------------------------------------
000650* LENG関数によって、データ項目長を求めます。
000660* LENG関数を利用することによって、項目追加(集団項目長の変化)の
000670* 影響を受けないコーディングが可能になります。
000680*-----------------------------------------------------------------
000690      COMPUTE TOTAL-SIZE = ( FUNCTION LENG(PTR->LONG-FORM)  - 1 ) * COUNT-L
000700                         + ( FUNCTION LENG(PTR->SHORT-FORM) - 1 ) * COUNT-S.
000710*-----------------------------------------------------------------
000720      DISPLAY "DATA           :" WORK.
000730      DISPLAY "TOTAL DATA SIZE:" TOTAL-SIZE.
000740 END PROGRAM SAMPLE.
```

H.49 NATIONAL関数

引数で指定されたデータ中の日本語文字を除くCOBOL文字集合(つまり、英字、数字、特殊文字)を対応する日本語文字に変換します。

例

```
000010 @OPTIONS MAIN
000020*-----------------------------------------------------------------
000030* NATIONAL関数を利用して、英数字文字列を日本語文字列に変換します。
000040*-----------------------------------------------------------------
000050 IDENTIFICATION   DIVISION.
000060 PROGRAM-ID.      SAMPLE.
000070 DATA             DIVISION.
000080  WORKING-STORAGE SECTION.
000090  01 混在データ   PIC X(40).
000130  01 住所         PIC N(40).
000140 PROCEDURE        DIVISION.
000150      DISPLAY "住所を入力してください。".
000180      DISPLAY "住所(全角/半角混在可) >> " WITH NO ADVANCING.
000190      ACCEPT 混在データ FROM CONSOLE.
000200*-----------------------------------------------------------------
000210* 住所データには、番地やマンション名など半角データが入りがちです。
000220* そのような混在データを日本語(全角)で統一して管理したい場合、
000230* NATIONAL関数を利用すると便利です。
000240*-----------------------------------------------------------------
000250      MOVE FUNCTION NATIONAL(混在データ) TO 住所.
000260*-----------------------------------------------------------------
000270      DISPLAY " ".
000280      DISPLAY "住所: " 住所.
000290 END PROGRAM SAMPLE.
```

H.50 CURRENT-DATE関数

現在の日付と時刻を求めることができます。

日付や時刻はACCEPT文のFROM指定でも求められますが、ACCEPT文の場合、西暦の下2桁が返却されるため、4桁で求めたい場合(2000年対応など)は、このCURRENT-DATE関数を使用してください。

例

```
000010 @OPTIONS MAIN
000020*---------------------------------------------------------------
000030* CURRENT-DATE関数で得た日付、時刻、曜日およびグリニッジ標準時
000040* との時差を表示します。
000050*---------------------------------------------------------------
000060 IDENTIFICATION   DIVISION.
000070 PROGRAM-ID.      SAMPLE.
000080 DATA             DIVISION.
000090  WORKING-STORAGE SECTION.
000100  01 TODAY.
000110    02 CR-YEAR      PIC X(4).
000120    02 CR-MON       PIC X(2).
000130    02 CR-DAY       PIC X(2).
000140    02 CR-HOUR      PIC X(2).
000150    02 CR-MIN       PIC X(2).
000160    02 CR-SEC       PIC X(2).
000170    02 CR-MSEC      PIC X(2).
000180    02 LOCAL-TIME.
000190      03 TIME-DIF     PIC X(1).
000200      03 TIME-DIF-H   PIC X(2).
000210      03 TIME-DIF-M   PIC X(2).
000220  01 CR-DOW         PIC 9(1).
000230  CONSTANT         SECTION.
000240  01 DOW-TABLE.
000250    02               PIC N(3) VALUE NC"月曜日".
000260    02               PIC N(3) VALUE NC"火曜日".
000270    02               PIC N(3) VALUE NC"水曜日".
000280    02               PIC N(3) VALUE NC"木曜日".
000290    02               PIC N(3) VALUE NC"金曜日".
000300    02               PIC N(3) VALUE NC"土曜日".
000310    02               PIC N(3) VALUE NC"日曜日".
000320  01               REDEFINES DOW-TABLE.
000330    02 DOW            OCCURS 7 TIMES PIC N(3).
000340 PROCEDURE        DIVISION.
000350*---------------------------------------------------------------
000360* CURRENT-DATE関数で現在の日付、時刻を求めます。
000370* なお、曜日はCURRENT-DATE関数では求められないため、ACCEPT文の
000380* DAY-OF-WEEK指定を利用します。
000390*---------------------------------------------------------------
000400     MOVE FUNCTION CURRENT-DATE TO TODAY.
000410     ACCEPT CR-DOW FROM DAY-OF-WEEK.
000420*---------------------------------------------------------------
000430     DISPLAY "日付: " CR-YEAR "年" CR-MON "月" CR-DAY "日(" DOW(CR-DOW) ")".
000440     DISPLAY "時刻: " CR-HOUR "時" CR-MIN "分" CR-SEC "秒" CR-MSEC " ".
000450     IF LOCAL-TIME NOT = 0 THEN
000460       DISPLAY "この地域のグリニッジ標準時との時差: "
000470               TIME-DIF TIME-DIF-H "時間" TIME-DIF-M "分"
000480     END-IF.
000490 END PROGRAM SAMPLE.
```

H.51 SUM関数

引数に指定された数値の合計を返却します。

例

```
000010 @OPTIONS MAIN
000020*---------------------------------------------------------------
```

```
000030* SUM関数は、引数に指定された数値の合計を返却します。
000040* なお、このサンプルでは、
000050* - MEAN関数   : 引数の算術平均値を返却
000060* - MEDIAN関数: 引数の中央値を返却
000070* も同時に使用しています。
000080*-----------------------------------------------------------------
000090 IDENTIFICATION   DIVISION.
000100 PROGRAM-ID.      SAMPLE.
000110 DATA             DIVISION.
000120  WORKING-STORAGE SECTION.
000130  01 .
000140   02 VAL         PIC S9(4) OCCURS 5 TIMES.
000150  01 TOTAL        PIC S9(8) VALUE 0.
000160  01 MEAN         PIC S9(8) VALUE 0.
000170  01 MEDIAN       PIC S9(8) VALUE 0.
000180  01 MIDRANGE     PIC S9(8) VALUE 0.
000190  01 SELECT-SW    PIC 9(1).
000200    88 SW-ALL       VALUE 1.
000210    88 SW-PART      VALUE 2.
000220  01 COUNTER      PIC 9(1).
000230 PROCEDURE        DIVISION.
000240     DISPLAY "４桁以下の数値を５回入力してください。".
000250     PERFORM TEST BEFORE
000260             VARYING COUNTER FROM 1 BY 1 UNTIL COUNTER > 5
000270       DISPLAY "数値(" COUNTER ")" " >> " WITH NO ADVANCING
000280       ACCEPT VAL(COUNTER) FROM CONSOLE
000290     END-PERFORM.
000300     DISPLAY "(" WITH NO ADVANCING.
000310     PERFORM TEST BEFORE
000320             VARYING COUNTER FROM 1 BY 1 UNTIL COUNTER > 5
000330       DISPLAY VAL(COUNTER) " " WITH NO ADVANCING
000340     END-PERFORM.
000350     DISPLAY ")".
000360     DISPLAY " ".
000370     DISPLAY "処理方法を選択してください。".
000380     DISPLAY "1 : すべて処理する。".
000390     DISPLAY "2 : 最初と最後を無視する。  >> " WITH NO ADVANCING.
000400     ACCEPT SELECT-SW.
000410*-----------------------------------------------------------------
000420* 合計, 平均, 中央値を求めます。引数に繰返し項目の要素を
000430* すべてを指定する場合, 添字ALL指定で代用できます。
000440*-----------------------------------------------------------------
000450     EVALUATE TRUE
000460     WHEN SW-ALL
000470       COMPUTE TOTAL  = FUNCTION SUM(VAL(ALL))
000480       COMPUTE MEAN   = FUNCTION MEAN(VAL(ALL))
000490       COMPUTE MEDIAN = FUNCTION MEDIAN(VAL(ALL))
000500*-----------------------------------------------------------------
000510* 合計, 平均, 中央値を求めます。
000520*-----------------------------------------------------------------
000530     WHEN SW-PART
000540       COMPUTE TOTAL  = FUNCTION SUM(VAL(2), VAL(3), VAL(4))
000550       COMPUTE MEAN   = FUNCTION MEAN(VAL(2), VAL(3), VAL(4))
000560       COMPUTE MEDIAN = FUNCTION MEDIAN(VAL(2), VAL(3), VAL(4))
000570*-----------------------------------------------------------------
000580     WHEN OTHER
000590       DISPLAY "選択に誤りがあります。"
000600       EXIT PROGRAM
000610     END-EVALUATE.
000620     DISPLAY " ".
000630     DISPLAY "合計値は" TOTAL.
000640     DISPLAY "平均値は" MEAN.
```

```
000650      DISPLAY "中央値は" MEDIAN.
000660 END PROGRAM SAMPLE.
```

H.52 REM関数

引数1を引数2で割った余りを返却します。

除算結果の余りだけが必要(意味を持つ)な場合にREM関数を利用します。

例

```
000010 @OPTIONS MAIN
000020*---------------------------------------------------------------
000030* 除算結果の余りを求めたい場合、REM関数を利用します。
000040*---------------------------------------------------------------
000050 IDENTIFICATION   DIVISION.
000060 PROGRAM-ID.      SAMPLE.
000070 DATA             DIVISION.
000080  WORKING-STORAGE SECTION.
000090  01 誕生日       PIC 9(8).
000100  01 通算日       PIC 9(8) BINARY.
000110  01 余り         PIC 9(8) BINARY.
000120  CONSTANT        SECTION.
000130  01 曜日テーブル.
000140    02             PIC N(3) VALUE NC"日曜日".
000150    02             PIC N(3) VALUE NC"月曜日".
000160    02             PIC N(3) VALUE NC"火曜日".
000170    02             PIC N(3) VALUE NC"水曜日".
000180    02             PIC N(3) VALUE NC"木曜日".
000190    02             PIC N(3) VALUE NC"金曜日".
000200    02             PIC N(3) VALUE NC"土曜日".
000210  01               REDEFINES 曜日テーブル.
000220    02 曜日         OCCURS 7 TIMES PIC N(3).
000230 PROCEDURE        DIVISION.
000240     DISPLAY "あなたの誕生日は？ 例:19690123 >> " WITH NO ADVANCING.
000250     ACCEPT 誕生日.
000260*---------------------------------------------------------------
000270* INTEGER-OF-DATE関数で1601年1月1日(月曜日)からの通算日を求め、
000280* 7で割った余りから、曜日を特定します。
000290*---------------------------------------------------------------
000300     COMPUTE 通算日 = FUNCTION INTEGER-OF-DATE(誕生日).
000310     COMPUTE 余り   = FUNCTION REM(通算日 7).
000320*---------------------------------------------------------------
000330     DISPLAY "あなたは、" 曜日(余り + 1) "生まれです。".
000340 END PROGRAM SAMPLE.
```

H.53 INTEGER-OF-DATE関数

INTEGER-OF-DATE関数は、引数に指定された標準形式の日付(yyyymmdd形式)を1601年1月1日からの通日で返却します。

例

```
000010 @OPTIONS MAIN
000020*---------------------------------------------------------------
000030* INTEGER-OF-DATE関数およびDATE-OF-INTEGER関数を利用して、
000040* 指定日数が経過した後の日付を求めます。
000050*---------------------------------------------------------------
```

```
000060 IDENTIFICATION   DIVISION.
000070 PROGRAM-ID.      SAMPLE.
000080 DATA             DIVISION.
000090  WORKING-STORAGE SECTION.
000100  01 TODAY.
000110    02 YYYYMMDD     PIC 9(8).
000120  01 OTHER-DAY1   PIC S9(9) BINARY.
000130  01 OTHER-DAY2   PIC 9(8).
000140  01 DAYS         PIC S9(4) BINARY.
000150 PROCEDURE       DIVISION.
000160     MOVE FUNCTION CURRENT-DATE TO TODAY.
000170     DISPLAY "今日は、" TODAY " です。".
000180     DISPLAY "何日後の日付を求めますか？ >> " WITH NO ADVANCING.
000190     ACCEPT DAYS.
000200*-----------------------------------------------------------------
000210* 指定された日数が経過後の日付を通日で計算します。
000220*-----------------------------------------------------------------
000230     COMPUTE OTHER-DAY1 = FUNCTION INTEGER-OF-DATE(YYYYMMDD) + DAYS.
000240*-----------------------------------------------------------------
000250* 表示のため、通日を標準形式(YYYYMMDD)に変換します。
000260*-----------------------------------------------------------------
000270     COMPUTE OTHER-DAY2 = FUNCTION DATE-OF-INTEGER(OTHER-DAY1).
000280*-----------------------------------------------------------------
000290     DISPLAY " ".
000300     DISPLAY TODAY " の " DAYS "日後は " OTHER-DAY2 " です。".
000310 END PROGRAM SAMPLE.
```

H.54 DATE-OF-INTEGER関数

DATE-OF-INTEGER関数は、引数に指定された1601年1月1日からの通日を標準形式の日付(yyyymmdd形式)で返却します。

例

```
000010 @OPTIONS MAIN
000020*-----------------------------------------------------------------
000030* INTEGER-OF-DATE関数およびDATE-OF-INTEGER関数を利用して、
000040* 指定日数が経過した後の日付を求めます。
000050*-----------------------------------------------------------------
000060 IDENTIFICATION   DIVISION.
000070 PROGRAM-ID.      SAMPLE.
000080 DATA             DIVISION.
000090  WORKING-STORAGE SECTION.
000100  01 TODAY.
000110    02 YYYYMMDD     PIC 9(8).
000120  01 OTHER-DAY1   PIC S9(9) BINARY.
000130  01 OTHER-DAY2   PIC 9(8).
000140  01 DAYS         PIC S9(4) BINARY.
000150 PROCEDURE       DIVISION.
000160     MOVE FUNCTION CURRENT-DATE TO TODAY.
000170     DISPLAY "今日は、" TODAY " です。".
000180     DISPLAY "何日後の日付を求めますか？ >> " WITH NO ADVANCING.
000190     ACCEPT DAYS.
000200*-----------------------------------------------------------------
000210* 指定された日数が経過後の日付を通日で計算します。
000220*-----------------------------------------------------------------
000230     COMPUTE OTHER-DAY1 = FUNCTION INTEGER-OF-DATE(YYYYMMDD) + DAYS.
000240*-----------------------------------------------------------------
000250* 表示のため、通日を標準形式(YYYYMMDD)に変換します。
000260*-----------------------------------------------------------------
```

```
000270     COMPUTE OTHER-DAY2 = FUNCTION DATE-OF-INTEGER(OTHER-DAY1).
000280*-----------------------------------------------------------------
000290     DISPLAY " ".
000300     DISPLAY TODAY " の " DAYS "日後は " OTHER-DAY2 " です。".
000310 END PROGRAM SAMPLE.
```

H.55 LOWER-CASE関数

引数で指定された文字列中の英大文字を英小文字に変換します。

英文字のデータを小文字で統一して管理したい場合に便利です。

例

```
000010 @OPTIONS MAIN
000020*-----------------------------------------------------------------
000030* 英字データを、
000040* - 英小文字で統一したい場合にはLOWER-CASE関数を利用します。
000050* - 英大文字で統一したい場合にはUPPER-CASE関数を利用します。
000060*-----------------------------------------------------------------
000070 IDENTIFICATION   DIVISION.
000080 PROGRAM-ID.      SAMPLE.
000090*
000100 DATA             DIVISION.
000110  WORKING-STORAGE SECTION.
000120  01 IN-STR       PIC X(40).
000130  01 LOWER-STR    PIC X(40).
000140  01 UPPER-STR    PIC X(40).
000150*
000160 PROCEDURE          DIVISION.
000170     DISPLAY "名前を英字で入力してください。>> " WITH NO ADVANCING.
000180     ACCEPT IN-STR FROM CONSOLE.
000190*-----------------------------------------------------------------
000200* 英小文字を英大文字に変換する
000210*-----------------------------------------------------------------
000220     MOVE FUNCTION LOWER-CASE(IN-STR) TO LOWER-STR.
000230     MOVE FUNCTION UPPER-CASE(IN-STR) TO UPPER-STR.
000240*-----------------------------------------------------------------
000250     DISPLAY " ".
000260     DISPLAY "英小文字表現: " LOWER-STR.
000270     DISPLAY "英大文字表現: " UPPER-STR.
000280 END PROGRAM SAMPLE.
```

H.56 UPPER-CASE関数

引数で指定された文字列中の英小文字を英大文字に変換します。

英文字のデータを大文字で統一して管理したい場合に便利です。

例

```
000010 @OPTIONS MAIN
000020*-----------------------------------------------------------------
000030* 英字データを、
000040* - 英小文字で統一したい場合にはLOWER-CASE関数を利用します。
000050* - 英大文字で統一したい場合にはUPPER-CASE関数を利用します。
000060*-----------------------------------------------------------------
000070 IDENTIFICATION   DIVISION.
000080 PROGRAM-ID.      SAMPLE.
```

```
000090*
000100 DATA             DIVISION.
000110  WORKING-STORAGE SECTION.
000120  01 IN-STR       PIC X(40).
000130  01 LOWER-STR    PIC X(40).
000140  01 UPPER-STR    PIC X(40).
000150*
000160 PROCEDURE          DIVISION.
000170     DISPLAY "名前を英字で入力してください。>> " WITH NO ADVANCING.
000180     ACCEPT IN-STR FROM CONSOLE.
000190*-----------------------------------------------------------------
000200* 英小文字を英大文字に変換する
000210*-----------------------------------------------------------------
000220     MOVE FUNCTION LOWER-CASE(IN-STR) TO LOWER-STR.
000230     MOVE FUNCTION UPPER-CASE(IN-STR) TO UPPER-STR.
000240*-----------------------------------------------------------------
000250     DISPLAY " ".
000260     DISPLAY "英小文字表現: " LOWER-STR.
000270     DISPLAY "英大文字表現: " UPPER-STR.
000280 END PROGRAM SAMPLE.
```

H.57 MAX関数

MAX関数は、引数の最大値を返却します。

例

```
000010 @OPTIONS MAIN
000020*-----------------------------------------------------------------
000030* MAX関数、MIN関数を使用して、入力した３つの数値の最大値および
000040* 最小値を求めます。
000050*-----------------------------------------------------------------
000060 IDENTIFICATION   DIVISION.
000070 PROGRAM-ID.      SAMPLE.
000080 DATA             DIVISION.
000090  WORKING-STORAGE SECTION.
000100  01 .
000110    02 VAL          OCCURS 3 TIMES PIC 9(4).
000120  01 MAX-VAL      PIC 9(5).
000130  01 MIN-VAL      PIC 9(5).
000140  01 COUNTER      PIC 9(1).
000150 PROCEDURE        DIVISION.
000160     DISPLAY "３つの数字を入力してください。".
000170     PERFORM TEST BEFORE
000180             VARYING COUNTER FROM 1 BY 1 UNTIL COUNTER > 3
000190       DISPLAY "数字(最大４桁) >> " WITH NO ADVANCING
000200       ACCEPT VAL(COUNTER)
000210     END-PERFORM.
000220     DISPLAY " ".
000230     DISPLAY "入力されたデータの最大値と最小値を求めます。"
000240*-----------------------------------------------------------------
000250* 最大値と最小値を求めます。
000260*-----------------------------------------------------------------
000270     COMPUTE MAX-VAL = FUNCTION MAX(VAL(ALL)).
000280     COMPUTE MIN-VAL = FUNCTION MIN(VAL(ALL)).
000290*-----------------------------------------------------------------
000300     DISPLAY "最大値は " MAX-VAL " です。".
000310     DISPLAY "最小値は " MIN-VAL " です。".
000320 END PROGRAM SAMPLE.
```

H.58 MIN関数

MIN関数は、引数の最小値を返却します。

例

```
000010 @OPTIONS MAIN
000020*---------------------------------------------------------------
000030* MAX関数、MIN関数を使用して、入力した３つの数値の最大値および
000040* 最小値を求めます。
000050*---------------------------------------------------------------
000060 IDENTIFICATION   DIVISION.
000070 PROGRAM-ID.      SAMPLE.
000080 DATA             DIVISION.
000090  WORKING-STORAGE SECTION.
000100  01 .
000110    02 VAL          OCCURS 3 TIMES PIC 9(4).
000120  01 MAX-VAL      PIC 9(5).
000130  01 MIN-VAL      PIC 9(5).
000140  01 COUNTER      PIC 9(1).
000150 PROCEDURE        DIVISION.
000160     DISPLAY "３つの数字を入力してください。".
000170     PERFORM TEST BEFORE
000180             VARYING COUNTER FROM 1 BY 1 UNTIL COUNTER > 3
000190       DISPLAY "数字(最大４桁) >> " WITH NO ADVANCING
000200       ACCEPT VAL(COUNTER)
000210     END-PERFORM.
000220     DISPLAY " ".
000230     DISPLAY "入力されたデータの最大値と最小値を求めます。"
000240*---------------------------------------------------------------
000250* 最大値と最小値を求めます。
000260*---------------------------------------------------------------
000270     COMPUTE MAX-VAL = FUNCTION MAX(VAL(ALL)).
000280     COMPUTE MIN-VAL = FUNCTION MIN(VAL(ALL)).
000290*---------------------------------------------------------------
000300     DISPLAY "最大値は " MAX-VAL " です。".
000310     DISPLAY "最小値は " MIN-VAL " です。".
000320 END PROGRAM SAMPLE.
```

H.59 REVERSE関数

REVERSE関数は、引数の文字列を逆順に並べ替えます。

例

```
000010 @OPTIONS MAIN
000020*---------------------------------------------------------------
000030* REVERSE関数は、引数の文字列を逆順に並べ替えます。
000040*---------------------------------------------------------------
000050 IDENTIFICATION   DIVISION.
000060 PROGRAM-ID.      SAMPLE.
000070 DATA             DIVISION.
000080  WORKING-STORAGE SECTION.
000090  01 IN-STR       PIC X(10).
000100  01 REV-STR      PIC X(10).
000110 PROCEDURE        DIVISION.
000120     DISPLAY "英字を10文字入力してください >> " WITH NO ADVANCING.
000130     ACCEPT IN-STR FROM CONSOLE.
000140*---------------------------------------------------------------
```

```
000150* 入力された文字列を逆順に並び替えます。
000160*-----------------------------------------------------------------
000170      MOVE FUNCTION REVERSE(IN-STR) TO REV-STR.
000180*-----------------------------------------------------------------
000190      DISPLAY " ".
000200      DISPLAY "入力文字列 : " IN-STR.
000210      DISPLAY "逆順文字列 : " REV-STR.
000220 END PROGRAM SAMPLE.
```

H.60 STORED-CHAR-LENGTH関数

STORED-CHAR-LENGTH関数は、引数に含まれる有効文字(末尾の空白を除いた部分)の長さを返却します。文字位置の個数が返却されるため、転記時のけた落ちのチェックや、部分参照などにより有効データを切り出すような場合に便利です。

なお、STORED-CHAR-LENGTH関数は、[Win32][WinIPF][Winx64][Solaris][Linux][LinuxIPF][Linux64][.NET]固有機能です。

例

```
000010 @OPTIONS MAIN
000020*-----------------------------------------------------------------
000030* STORED-CHAR-LENGTH関数は、データ項目中の有効データ長を返却します。
000040*-----------------------------------------------------------------
000050 IDENTIFICATION   DIVISION.
000060 PROGRAM-ID.      SAMPLE.
000070 DATA             DIVISION.
000080  WORKING-STORAGE SECTION.
000090  01 IN-DATA      PIC X(80).
000100  01 DATA-LEN     PIC 9(4) BINARY.
000110  01 BUFFER       PIC X(40) VALUE ALL "*".
000120**
000130 PROCEDURE        DIVISION.
000140      DISPLAY "文字列を入力してください。>> " WITH NO ADVANCING.
000150      ACCEPT IN-DATA FROM CONSOLE.
000160*-----------------------------------------------------------------
000170* STORED-CHAR-LENGTH関数の関数値は、引数が日本語項目の場合は文字数、
000180* 英数字項目の場合はバイト数になります。
000190* STORED-CHAR-LENGTH関数は数字関数のため、算術式にのみ記述できます。
000200* 以下でMOVE文ではなくCOMPUTE文を使用しているのはそのためです。
000210*-----------------------------------------------------------------
000220      COMPUTE DATA-LEN = FUNCTION STORED-CHAR-LENGTH(IN-DATA).
000230*-----------------------------------------------------------------
000240      IF DATA-LEN > FUNCTION LENGTH(BUFFER) THEN
000250        DISPLAY NC"入力データ長がバッファ長を超えました。"
000260      ELSE
000270        MOVE IN-DATA(1:DATA-LEN) TO  BUFFER(1:DATA-LEN)
000280        DISPLAY "BUFFER = " BUFFER
000290      END-IF.
000300 END PROGRAM SAMPLE.
```

H.61 UCS2-OF関数

UTF-8のデータをUCS-2のデータに変換します。

他言語連携やファイル入出力において、相手や媒体に合わせてデータ表現形式の変換が必要な際に利用します。たとえば、Windows関数とUnicodeで連携する場合、UCS-2でデータを送受する必要があるため、必要に応じて変換します。

このサンプルはWindows関数を呼び出しているため、リンク時にCOBOLインストールフォルダ(ディレクトリ)に格納されている"USER32.LIB"をリンクする必要があります。注意してください。

なお、UCS2-OF関数は、[Win32][WinIPF][Winx64][Solaris][Linux][LinuxIPF][Linux64][.NET]固有機能です。

このサンプルは[Win32]で動作します。

例

```
000010 @OPTIONS MAIN,ALPHAL(WORD),RCS(UCS2)
000020*---------------------------------------------------------------
000030* UCS2-OF関数は、UTF-8のデータをUCS-2に変換します。
000040*---------------------------------------------------------------
000050 IDENTIFICATION   DIVISION.
000060 PROGRAM-ID.      SAMPLE.
000070 DATA             DIVISION.
000080  WORKING-STORAGE SECTION.
000090  77 IN-DATA      PIC X(80).
000100  77 DATA-LEN     PIC S9(4) BINARY.
000110  77 MSG-TXT      PIC N(80).
000120  77 TITLE-TXT    PIC N(10).
000130  77 RET          PIC S9(9) COMP-5 VALUE 0.
000140 PROCEDURE        DIVISION.
000150     DISPLAY "メッセージを入力してください。>> " WITH NO ADVANCING.
000160     ACCEPT IN-DATA FROM CONSOLE.
000170     COMPUTE DATA-LEN = FUNCTION STORED-CHAR-LENGTH(IN-DATA).
000180*---------------------------------------------------------------
000190* 入力された英数字データ(UTF-8)を日本語データ(UCS-2)に変換します。
000200*---------------------------------------------------------------
000210     MOVE FUNCTION UCS2-OF(IN-DATA(1:DATA-LEN)) TO MSG-TXT.
000220*---------------------------------------------------------------
000230     MOVE NX"0000" TO MSG-TXT(FUNCTION STORED-CHAR-LENGTH(MSG-TXT) + 1:1).
000240     MOVE NC"Sample" & NX"0000" TO TITLE-TXT.
000250     CALL "MessageBoxW" WITH STDCALL USING BY VALUE     0
000260                                           BY REFERENCE MSG-TXT
000270                                           BY REFERENCE TITLE-TXT
000280                                           BY VALUE     1
000290                                     RETURNING RET.
000300     EXIT PROGRAM.
000310 END PROGRAM SAMPLE.
```

H.62 UTF8-OF関数

UCS-2のデータをUTF-8のデータに変換します。

他言語連携やファイル入出力において、相手や媒体に合わせてデータ表現形式の変換が必要な際に利用します。

なお、UTF8-OF関数は、[Win32][WinIPF][Winx64][Solaris][Linux][LinuxIPF][Linux64][.NET]固有です。

例

```
000010 @OPTIONS MAIN,RCS(UCS2)
000020*---------------------------------------------------------------
000030* UTF8-OF関数は、UCS-2のデータをUTF-8に変換します。
000040*---------------------------------------------------------------
000050 IDENTIFICATION   DIVISION.
000060 PROGRAM-ID.      SAMPLE.
000070 DATA             DIVISION.
000080  WORKING-STORAGE SECTION.
000090  01 データ.
000100    02 .
```

```
000110       03                PIC N(12) VALUE NC"富士通　太郎".
000120       03                PIC X(12) VALUE "0123-45-6789".
000130     02 .
000140       03                PIC N(12) VALUE NC"富士通　花子".
000150       03                PIC X(12) VALUE "9876-54-3210".
000160     02 .
000170       03                PIC N(12) VALUE NC"John Smith".
000180       03                PIC X(12) VALUE "6789-01-2345".
000190   01 電話帳         REDEFINES データ.
000200     02 エントリ       OCCURS 2 TIMES.
000210       03 名前              PIC N(12).
000220       03 電話              PIC X(12).
000230   77 番号           PIC 9(1).
000240   77 短ヘッダ       PIC X(17) VALUE "<NAME>      <TEL>".
000250   77 長ヘッダ       PIC X(29) VALUE "<NAME>                  <TEL>".
000260 PROCEDURE          DIVISION.
000270      DISPLAY "何番目の情報を表示しますか? (1～3) >> " WITH NO ADVANCING.
000280      ACCEPT 番号.
000290      DISPLAY " ".
000300*-----------------------------------------------------------------
000310* 名前データの領域は日本語項目で12文字分、用意されています。
000320* 同じ12文字でも、半角文字と全角文字では表示長が異なるため、名前
000330* が半角英字だけで構成されていた場合、短形式のヘッダを表示します。
000340* → 字類条件は日本語項目、つまりUCS-2のデータには使用できない
000350*     ため、UTF-8に変換して字類検査しています。
000360*-----------------------------------------------------------------
000370      IF FUNCTION UTF8-OF(名前(番号)) IS ALPHABETIC THEN
000380        DISPLAY 短ヘッダ
000390      ELSE
000400        DISPLAY 長ヘッダ
000410      END-IF.
000420*-----------------------------------------------------------------
000430      DISPLAY 名前(番号) 電話(番号)
000440 END PROGRAM SAMPLE.
```

H.63 WHEN-COMPILED関数

WHEN-COMPILED関数は、プログラムを翻訳したときの日付、時刻およびグリニッジ標準時との時差を返却します。

例

```
000010 @OPTIONS MAIN
000020*-----------------------------------------------------------------
000030* WHEN-COMPILED関数でプログラムの翻訳日付を求めます。
000040*-----------------------------------------------------------------
000050 IDENTIFICATION   DIVISION.
000060 PROGRAM-ID.      SAMPLE.
000070 DATA             DIVISION.
000080  WORKING-STORAGE SECTION.
000090  01 翻訳日付.
000100    02 年           PIC 9(4).
000110    02 月           PIC 9(2).
000120    02 日           PIC 9(2).
000130    02 時           PIC 9(2).
000140    02 分           PIC 9(2).
000150    02 秒           PIC 9(2).
000160  01 編集データ.
000170    02 年           PIC 9(4).
000180    02             PIC X(1) VALUE "/".
000190    02 月           PIC 9(2).
```

```
000200     02              PIC X(1) VALUE "/".
000210     02 日           PIC 9(2).
000220     02              PIC X(1) VALUE SPACE.
000230     02 時           PIC 9(2).
000240     02              PIC X(1) VALUE ":".
000250     02 分           PIC 9(2).
000260     02              PIC X(1) VALUE ":".
000270     02 秒           PIC 9(2).
000280 PROCEDURE        DIVISION.
000290*-----------------------------------------------------------------
000300* プログラムの実行日付と翻訳日付を取り出します。
000310*-----------------------------------------------------------------
000320      MOVE FUNCTION WHEN-COMPILED TO 翻訳日付.
000330*-----------------------------------------------------------------
000340      MOVE CORRESPONDING 翻訳日付 TO 編集データ.
000350      DISPLAY "このプログラムは、" 編集データ " に翻訳されました。".
000360 END PROGRAM SAMPLE.
```

H.64 COPY文

複数のプログラムで共通のデータ宣言や手続きを利用する場合、登録集を作成して必要に応じてCOPY文で取り込みます。

このとき、登録集中の語を置換して取り込んだり(書き方1)、語の部分文字列を置換して取り込む(書き方2)ことも可能です。

COPY文(書き方1)

例

登録集 "CP-SMPL1.CBL"

```
000010*-----------------------------------------------------------------
000020* 日付データを定義している登録集です。
000030*-----------------------------------------------------------------
000040  01 FMT-DATE.
000050     02 YYYY       PIC 9(4).
000060     02 MMDD.
000070        03 MM         PIC 9(2).
000080        03 DD         PIC 9(2).
```

原始プログラム

```
000010 @OPTIONS MAIN
000020*-----------------------------------------------------------------
000030* 登録集の読み込みにはCOPY文を指定します。
000040*-----------------------------------------------------------------
000050 IDENTIFICATION   DIVISION.
000060 PROGRAM-ID.      SAMPLE.
000070 DATA             DIVISION.
000080  WORKING-STORAGE SECTION.
000090*-----------------------------------------------------------------
000100* COPY文にREPLACINGを指定することによって、登録集中の語(原文語)を
000110* 置き換えて展開することができます。
000120* この例の場合、一方は集団項目名を"TODAY"に、もう一方は"BIRTHDAY"
000130* に置き換えて日付データを展開しています。
000140*-----------------------------------------------------------------
000150  COPY CP-SMPL1 REPLACING FMT-DATE BY TODAY.
000160  COPY CP-SMPL1 REPLACING FMT-DATE BY BIRTHDAY.
000170*-----------------------------------------------------------------
000180  01 AGE          PIC 9(3).
000190 PROCEDURE        DIVISION.
```

```
000200     DISPLAY "あなたの誕生日は ? 例:19690123 >> " WITH NO ADVANCING.
000210     ACCEPT BIRTHDAY.
000220     MOVE FUNCTION CURRENT-DATE TO TODAY.
000230     COMPUTE AGE = YYYY OF TODAY - YYYY OF BIRTHDAY.
000240     IF MMDD OF TODAY < MMDD OF BIRTHDAY THEN
000250       COMPUTE AGE = AGE - 1
000260     END-IF.
000270     DISPLAY " ".
000280     DISPLAY "あなたの年齢は、" AGE " 才です。".
000290 END PROGRAM SAMPLE.
```

COPY文(書き方2)

例

登録集 "CP-SMPL2.CBL"

```
000010*-----------------------------------------------------------------
000020* 日付データを定義している登録集です。
000030*-----------------------------------------------------------------
000040  01 XXX-DATE.
000050    02 XXX-YYYY   PIC 9(4).
000060    02 XXX-MMDD.
000070      03 MM         PIC 9(2).
000080      03 DD         PIC 9(2).
```

原始プログラム

```
000010 @OPTIONS MAIN
000020*-----------------------------------------------------------------
000030* 登録集を展開する際、部分文字列を置き換えることも可能です。
000040*-----------------------------------------------------------------
000050 IDENTIFICATION   DIVISION.
000060 PROGRAM-ID.      SAMPLE.
000070 DATA             DIVISION.
000080  WORKING-STORAGE SECTION.
000090*-----------------------------------------------------------------
000100* DISJOINING/JOINING指定により、登録集中に記述された原文語の部分
000110* 文字列("-"で連結された語)を置き換えて展開することも可能です。
000120*-----------------------------------------------------------------
000130  COPY CP-SMPL2 DISJOINING XXX JOINING TODAY    AS PREFIX.
000140  COPY CP-SMPL2 DISJOINING XXX JOINING BIRTHDAY AS PREFIX.
000150*-----------------------------------------------------------------
000160  01 AGE          PIC 9(3).
000170 PROCEDURE        DIVISION.
000180     DISPLAY "あなたの誕生日は ? 例:19690123 >> " WITH NO ADVANCING.
000190     ACCEPT BIRTHDAY-DATE.
000200     MOVE FUNCTION CURRENT-DATE TO TODAY-DATE.
000210     COMPUTE AGE = TODAY-YYYY - BIRTHDAY-YYYY.
000220     IF TODAY-MMDD < BIRTHDAY-MMDD THEN
000230       COMPUTE AGE = AGE - 1
000240     END-IF.
000250     DISPLAY " ".
000260     DISPLAY "あなたの年齢は、" AGE " 才です。".
000270 END PROGRAM SAMPLE.
```

索　引

[]

＆....3,17
＞＞D....37

[数字]

16進数字定数....463
16進文字定数....13
1行の構成....32,36
2進項目....178

[記号]

＊....33,34
＊＞....36
－....33,35
－－....458
－＞....3,30,35
.NETプログラミング....485
/....33,34
＝＝....3,35

[A]

ACCEPT文....257,258,259,470
ACCESS MODE句....107
ACOS関数....379
ACP-OF....587
ADDR関数....380
ADD文....260
ALL定数....15
ALPHABET句....88
ALTER....569
ALTERNATE RECORD KEY句....52,108
ALTER文....262
ANNUITY関数....380
APPLY MULTICONVERSATION-MODE句....128
APPLY SAVED-AREA句....129
ARGUMENT-NUMBER....75,88,259,277
ARGUMENT-VALUE....75,88,260
AS....79
ASCIIにおける文字の内部コード....612
ASIN関数....381
ASSIGN句....109,111
ATAN関数....382
AT END指定....255
attribute class....487
AUTO句....185
A領域....33
A領域から書き始める語....34

[B]

BACKGROUND-COLOR句....186
BASED ON句....31,182
BASED-STORAGE SECTION....133
BELL句....187
BETWEEN述語....430
BLANK LINE句....187
BLANK SCREEN句....187
BLANK WHEN ZERO句....149,188
BLINK句....188
BLOCK CONTAINS句....136
BY CONTENT指定....266
BY REFERENCE指定....266
BY VALUE指定....266
B領域....33
B領域から書き始める語....34

[C]

CALL....569
CALL文....262,452,483
CANCEL文....267
CANCEL文による初期化....61
CAST-ALPHANUMERIC関数....382
CHARACTER TYPE....528
CHARACTER TYPE句....149
CHAR関数....382
CLASS-ID....509
CLASS句....90
CLOSE文....268,441
CLR....488
CLR type data....488
CLR型データ....488
CLR型とCOBOLのデータ型の対応....534
CLS....488
COBOL original type data....488
COBOL独自型データ....488
COBOLの機能....39
COBOLの語....5,490
COBOL翻訳集団....502
COBOL文字集合....1
CODE-n....89
CODE-SET句....137
CODE句....199
COLUMN NUMBER句....189,206,467
COMMIT文....450
Common Language Runtime....488
Common Language Specification....488
COMMON句....79
COMPUTE文....271
CONFIGURATION SECTION....81,517
CONNECT文....450
CONSOLE....85,274
CONSOLE IS CRT句....464
CONSTANT SECTION....134
constructor....488
CONTINUE文....272
CONTROL RECORDS句....137
CONTROL句....200
COPY文....409
CORRESPONDING指定....254
COS関数....383
CRT STATUS句....91
CTL....85

CURRENCY SIGN句.....92
CURRENT-DATE関数.....383
CURSOR句.....92
custom-attribute.....488
CUSTOM-ATTRIBUTE.....515,518,528

[D]

D.....33,36
DATA DIVISION.....132
DATA RECORDS句.....137
DATE-COMPILED.....79
DATE-OF-INTEGER関数.....384
DAY-OF-INTEGER関数.....385
DCSQLMSG.....461
DCSQLSTATE.....460
DECIMAL-POINT IS COMMA句.....93
DECLARATIVES.....34
DELEGATE-ID.....514
DELETE文.....272,437,443
DESTINATION句.....112
DISCONNECT文.....452
DISPLAY-OF.....587
DISPLAY文.....274,275,276,473
DIVIDE文.....278
dllimport attribute.....490
DLLIMPORT属性.....490
DUPLICATES指定.....52

[E]

EBCDICにおける文字の内部コード.....611
EDIT-COLOR.....10,27,70
EDIT-CURSOR.....10,27,70
EDIT-MODE.....10,27,70
EDIT-OPTION.....10,27,70
EDIT-OPTION2.....10,27,70
EDIT-OPTION3.....10,27,70
EDIT-STATUS.....10,27,70
END-CALL指定.....266
END DECLARATIVES.....34
END KEY句.....112
END PROGRAM.....80
ENTRY.....571
ENTRY文.....280
ENUM.....488
ENUM-AND.....588
ENUM-ID.....515
ENUM-NOT.....588
ENUM-OR.....589
ENUM指定子.....521
ENUM名段落.....515
ENUM名.....491
ENVIRONMENT DIVISION.....81
ENVIRONMENT-NAME.....76,88,277
ENVIRONMENT-VALUE.....76,88,260,277
ERASE句.....189
EVALUATE文.....281,571
EVALUATE文の例.....283,284
EXCEPTION-OBJECT.....496
EXEC DCSQL BEGIN DECLARE SECTION END-EXEC 459
EXEC DCSQL END DECLARE SECTION END-EXEC….459
EXEC SQL BEGIN DECLARE SECTION END-EXEC.....417
EXEC SQL END DECLARE SECTION END-EXEC.....417
EXECUTE IMMEDIATE文.....446
EXECUTE文.....445,584
EXISTS述語.....432
EXIT PERFORM文.....285
EXIT PROGRAM.....572
EXIT PROGRAM文.....58,286
EXIT文.....284
EXTERNAL.....529
EXTERNAL句.....138,153

[F]

FACTORIAL関数.....386
FETCH文.....442
FILE-CONTROL.....97,524
FILE SECTION.....133,525
FILE STATUS句.....113
FILLER項目.....148
final class.....488
FOREGROUND-COLOR句.....190
FORMAT句.....114
FORMAT句付きの印刷ファイル.....52,114
FORMAT句なしの印刷ファイル.....52
FOR句.....435
FROM句.....433
FULL句.....191

[G]

GENERATE文.....286
GLOBAL.....529
GLOBAL句.....59,138,154
GO TO.....572
GO TO文.....288
GRID句.....467
GROUP BY句.....433
GROUP INDICATE句.....206
GROUP句.....114

[H]

HAVING句.....434
HIGHLIGHT句.....191
HIGH-VALUE.....15,90

[I]

IDENTIFICATION DIVISION.....78
IF文.....288
INDEXED指定.....52
INITIALIZE文.....47,290,573
INITIAL句.....79
INITIATE文.....292
INPUT-OUTPUT SECTION.....524
INPUT-OUTPUT SECTION.....97
INSERT文.....438
INSPECT文.....292
INSPECT文の例.....298

instance constructor....488
INTEGER-OF-DATE関数....387
INTEGER-OF-DAY関数....387
INTEGER-PART関数....388
INTEGER関数....386
INTERFACE-ID....514
INTO句....444
int型2進整数データ項目....154,168,179,275,615
INVALID KEY指定....254
INVOKE....573
IN述語....430
I-O-CONTROL....127
I制御レコード....625

[J]
JIS8単位コードにおける文字の内部コード....613
JUSTIFIED句....155,192

[L]
LABEL RECORDS句....139
LEFTLINE句....467
LENGTH関数....389
LENG関数....388
LIKE述語....431
LINAGE-COUNTER....10,26,57,141
LINAGE句....139
LINE-COUNTER....10,26,77,198
LINE NUMBER句....192,207,468
LINE NUMBER句の表記法....217
LINE SEQUENTIAL指定....52
LINKAGE SECTION....134,526
LOCK MODE句....54,115
LOCK MODE句でAUTOMATICを指定した場合のレコードのロック....55
LOCK MODE句でMANUALを指定した場合のレコードのロック....55
LOG10関数....390
LOG関数....390
LOWER-CASE関数....391
LOWLIGHT句....193
LOW-VALUE....15,90

[M]
MAX関数....391
MEAN関数....392
MEDIAN関数....392
MEMORY SIZE句....83
MERGE文....300,576
MESSAGE CLASS句....116
MESSAGE CODE句....117
MESSAGE MODE句....117
MESSAGE OWNER句....118
MESSAGE SEQUENCE句....118
metadata....490
METHOD-ID....510
method signature....488
Micro Focus固有機能....462
MIDRANGE関数....393
MIN関数....393
MOD関数....394
MOVE文....304
MULTIPLE FILE TAPE句....129
MULTIPLY文....305

[N]
name space....490
NATIONAL-OF....589
NATIONAL関数....394
.NETプログラミング機能....485
NEXT GROUP句....208
NEXT GROUP句の表記法....217
NULL....497
NULL述語....431
NUMVAL-C関数....396
NUMVAL関数....396

[O]
OBJECT....510
OBJECT-COMPUTER....82
OCCURS....530
OCCURS句....156,468
ON SIZE ERROR指定....253
OPEN文....306,310,311,441,576
OPTIONAL指定....124
ORD-MAX関数....398
ORD-MIN関数....398
ORD関数....397
ORGANIZATION句....119,120
OVERLINE句....469

[P]
PADDING CHARACTER句....120
PAGE-COUNTER....10,26,77,198
PAGE句....200
PERFORM文....311,577
PICTURE句....158,194
PICTURE句の文字列....18,161
POSITIONING UNIT句....93
PREPARE文....444
PRESENT-VALUE関数....399
PRINTER-n....110,583
PRINTING MODE句....94
PRINTING POSITION....530
PRINTING POSITION句....167
PRIVATE....536
private class....490
PROCEDURE DIVISION....225
PROCESSING CONTROL句....121
PROCESSING MODE句....121
PROCESSING TIME句....122
PROGRAM-ID....507
PROGRAM COLLATING SEQUENCE句....83
PROGRAM-ID....78
program prototype....490
program prototype definition....490
PROGRAM-STATUS....10,64

PROMPT句.....469
PROPERTY.....530
PROTECTED.....536
PUBLIC.....536

[Q]

QUOTE.....15

[R]

RAISE文.....578
RANDOM関数.....399
RANGE関数.....400
READ文.....319,325,476
RECORD DELIMITER句.....122
RECORD KEY句.....52,123
RECORD句.....141,143,144
REDEFINES.....532
REDEFINES句.....167
reference type.....488
RELATIVE KEY指定.....52
RELATIVE指定.....52
RELEASE文.....327
REM関数.....401
RENAMES記述項.....169
RENAMES句.....169
REPLACE文.....414
REPORT SECTION.....134
REPORT句.....144
REPOSITORY.....520
REQUIRED句.....195
RERUN句.....129
RESERVE句.....124
RESUME文.....578
RETURN-CODE.....10,64
RETURNING指定.....229,266
RETURN文.....327
REVERSE-VIDEO句.....195
REVERSE関数.....401
REWRITE文.....328
ROLLBACK文.....450
ROUNDED指定.....252

[S]

SAME RECORD AREA句.....57
SAME句.....130
SCREEN SECTION.....134
sealed class.....488
SEARCH文.....331
SEARCH文の例.....333
SECURE句.....196
SELECTED FUNCTION句.....125
SELECT句.....124
SELECT文.....436
SELF.....497
SEPARATE CHARACTER.....171
SEQUENTIAL指定.....52
SESSION CONTROL句.....125
SET.....579
SET CONNECTION文.....451
SET文.....335
signature.....488
SIGN句.....170,196,208
SIN関数.....402
SIZE句.....469
SORT-CORE-SIZE.....10,66
SORT-STATUS.....10,66
SORT文.....337,580
SOURCE-COMPUTER.....81
SOURCE句.....209
SPACE.....15
SPECIAL-NAMES.....518
SPECIAL-NAMES.....84
SQLCODE.....420
SQLERRD.....420
SQLMSG.....420
SQLSTATE.....420
SQL終了子.....416
SQL先頭子.....416
SQL文識別子.....425
SQL文に記述できる文字.....422
SQRT関数.....402
STANDARD-DEVIATION関数.....403
START文.....341,343,480,481
static.....489
STATIC.....510
STOP.....580
STOP RUN文.....48
STOP文.....347
STORED-CHAR-LENGTH関数.....403
STRING文.....348
STRING文の処理の流れ.....353
SUBTRACT文.....353
SUM関数.....404
SUM句.....210
SUPER.....497
SUPPRESS文.....355
SWITCH-n.....88
SYMBOLIC CHARACTERS句.....96
SYMBOLIC CONSTANT句.....96
SYMBOLIC DESTINATION句.....68,126
SYNCHRONIZED句.....20,21,23,171
SYSERR.....85,274
SYSIN.....85
SYSOUT.....85,274
SYSPUNCH.....85,274
S制御レコード.....629

[T]

TAN関数.....404
TERMINATE文.....355
TRY.....581
TYPEDEF句.....173
type object.....490
TYPE OF特殊レジスタ.....493,498
TYPE句.....172,211

[U]
UCS-2.....405
UCS2-OF関数.....405
UNDERLINE句.....196
Unicode.....14
UNICODE-OF.....590
UNIT CONTROL句.....127
UNLOCK文.....356
unmanaged code.....490
unmanaged code program.....490
UNSTRING文.....356
UNSTRING文の処理の流れ.....362
UPDATE文.....439,443
UPPER-CASE関数.....405
USAGE.....532
USAGE句.....174,197,215
USE.....582
USE BEFORE REPORTING文.....367
USE FOR DEAD-LOCK文.....368
USE文.....365
USING BY CONTENT指定.....62
USING BY REFERENCE指定.....62
USING BY VALUE指定.....62
USING句.....444
USING指定.....229,521
UTF-8.....406
UTF8-OF関数.....406

[V]
VALUE.....535
VALUE OF句.....145
value type.....487
VALUE句.....180,197,215
VARIANCE関数.....406
virtual attribute.....488

[W]
Webサービス.....485
WHEN-COMPILED関数.....407
WHERE句.....433
WITH C LINKAGE.....267
WITH DEBUGGING MODE句.....36,37,82
WITH PASCAL LINKAGE.....267
WITH STDCALL LINKAGE.....267
WORKING-STORAGE SECTION.....134
WRITE文.....369,373,376,583

[Z]
ZERO.....15
ZERO-FILL句.....470

[あ]
相手指定.....425
アクセス指定句.....536
アクセス属性.....487
脚書き領域.....140
遊びバイト.....20
遊びバイトの例.....21
遊びバイトを挿入する規則.....21
遊びビット.....20,23
遊びビットを挿入する規則.....23
値型.....487
値式.....428
値指定.....425
あて先種別.....68,126
アテンション種別.....125
後の行.....35,36
アトリビュートクラス.....487
余り.....401
暗に再定義.....148
暗にポインタ付け.....31
暗黙的なCANCEL文.....268
暗黙のFILLER項目.....21
暗黙のポインタ修飾子.....182
一意参照.....24,492
一意名.....31,492
位置決め単位名.....7,93
一連の手続きの共通の出口.....284
一連番号領域.....33
入れ子のIF文の例.....289
印字行に付ける定数.....199
印字行の上での印字位置.....206
印字項目.....205
印字項目に転記される値.....209
印字項目の値.....215
印字モード.....95
印字モード名.....7,94
インスタンスコンストラクタ.....488
インタフェース(メソッドの).....485
インタフェース指定子.....520
インタフェース指定の特殊クラス.....485
インタフェース名.....491
インタフェース名段落.....514
引用符.....4,455
受取り側項目.....350,351,358
受取り側作用対象.....304,336
右端に合わせて転記.....155
右端に合わせて表示.....192
うちPERFORM文.....43,313
うちPERFORM文の出口.....285
埋め草文字.....120
埋込みDCSQL.....458
埋込みDCSQL宣言節.....458
埋込みDCSQLの基本要素.....454
埋込みDCSQLの正書法.....458
埋込みDCSQLの注釈.....458
埋込みSQL.....416
埋込みSQL開始宣言.....417
埋込みSQL終了宣言.....417
埋込みSQL宣言節.....417,583
埋込みSQLの正書法.....416,583
埋込みSQLの定数.....422
埋込みSQL文.....416
埋込み例外宣言.....436,583
英大文字と等価な英小文字.....391

英小文字と等価な英大文字.....405
英字.....1
英字画面項目.....194
英字項目.....159
英字転記.....250
英数字・英数字編集転記.....250
英数字画面項目.....194
英数字関数.....378
英数字項目.....160
英数字編集画面項目.....194
英数字編集項目.....160
演算子.....231
演算符号.....170,196,208
送出し側項目.....350,351,358
送出し側作用対象.....304,336
オブジェクト(インスタンス).....485
オブジェクト一意名.....486
オブジェクト参照一意名.....486
オブジェクト指定子.....492,495
オブジェクト段落.....510
オブジェクト定義.....486,504
オブジェクトデータ.....486
オブジェクトプロパティ.....486,493,498
オブジェクトメソッド.....486
親クラス.....486
終わり見出し.....507,516
オーディオトーン.....187
オーバロード.....513
オープンモード.....53

[か]

階乗.....386
概数定数.....423,457
外部10進項目.....178
外部スイッチ.....87
外部スイッチの状態の設定.....335
外部属性.....59,138,153
外部データレコード.....59,154
外部ファイル結合子.....59
外部浮動小数点項目.....161
外部ブール項目.....178
外部名.....60,63,499
外部名の表記の規則.....500
改ページ.....206
各期末の現在価値.....399
カスタム属性.....488
カスタム属性名.....491
仮想属性.....488
型.....631
型が配列型であることを示すための修飾.....500
型条件.....542
型付けされた項目.....631
型の概念.....19
型パラメタ.....488
型引数.....488
型名.....9,631
型やメソッドが型パラメタを持つことを示すための修飾.....501
各国語文字の使用の可否.....422
各国語文字列定数.....423
各国語文字列定数の使用の可否.....423
活性状態.....60,539
可変長レコード形式.....57
可変反復データ項目.....44,158
可変反復データ項目の参照領域.....45
画面からデータを入力.....258,470
画面行を消去.....187
画面項目.....73
画面項目の空の文字位置に目印を付加.....469
画面項目の最左端の文字の左側に縦線を付加.....467
画面項目の上端に横線を付加.....469
画面項目の内容を点滅.....188
画面項目の背景色.....186
画面項目の前景色.....190
画面項目の前景色と背景色を反転.....195
画面項目の用途.....197
画面項目の論理的な大きさ.....469
画面項目への入力が必須.....195
画面項目を下線付きで表示.....196
画面項目を構成する各文字の左側に縦線を付加.....467
画面項目を配置する行.....192,468
画面項目を配置する列.....189,467
画面節.....134
画面全体を消去.....187
画面帳票定義体.....69
画面帳票定義体名.....114
画面データ記述項.....73,183,464
画面と画面項目.....73
画面入力状態.....74,91
画面の入出力操作.....74
仮原文.....409
仮原文区切り記号.....4,409
環境部.....81,517
環境部の構成.....81
環境変数に値の設定.....276
環境変数の値を更新する方法.....76
環境変数の値を参照する方法.....76
環境変数の操作.....75
環境変数を位置付け.....276
完結文.....228
漢字列定数.....456
関数.....379
関数一意名.....32,377,492,585
関数値.....71,585
関数値の属性と精度.....619
関数の型.....378,586
関数の全般規則.....377,585
関数の定義.....585
関数の呼出し形式.....377
関数名.....11,585
カーソル位置.....92
カーソル系データ操作文.....440,584
カーソル宣言.....440,584
カーソルの移動の自動化.....185
カーソル名.....425

記憶領域の共用.....130
記号定数.....7,96
記号文字.....7,15,96
記述の全般規定.....458
擬似乱数.....399
基底場所節.....133,148
機能名.....9,84,87,88
機能名-1句.....84
機能名-2句.....87
機能名-3句.....88
基本画面項目.....183
基本項目.....18
基本項目転記.....249
基本項目の大きさ.....161
逆順にした文字列.....401
逆正接.....382
逆余弦.....379
逆正弦.....381
行送り.....208
行カウンタ.....77,198
行カウンタの規則.....199
行順編成.....119
行順編成のファイル.....52
共通プログラム.....60,79
行内注記.....36,458
行の一部または画面の一部を消去.....189
行のつなぎ.....35,36,458,501
共用領域.....131
局所名.....59
キー項目.....302,339
キーワード.....455
空白行.....34,36
空白に置き換え.....149
空白を表示.....188
区切り文字.....350,358
組合せ条件.....239
組合せ比較条件.....244
組合せ比較条件の略記法.....244
組込み関数.....585
組込み関数機能.....71
クラス.....486
クラス指定子.....520
クラス定義.....486,503
クラス名.....486,490
クラス名段落.....509
クラス名の名前の範囲.....502
繰上げ.....211
繰返し回数.....156
繰返し処理.....43
計算機名.....9
計算機文字集合.....2
継承.....486
桁あふれ条件が発生しなかったときの動作.....254
桁あふれ条件が発生したときの動作.....253
桁あふれ条件が発生する条件.....253
言語の基本要素.....2,490
言語名.....9
現在の日付と時刻.....383
減算.....353
原始プログラム原文.....409,414
原始文操作.....409
原始文操作機能.....66
限定述語.....431
原文.....409
原文語.....409
原文名.....7
原文名定数.....17
高輝度で表示.....191
合計.....404
合計カウンタ.....210
合計報告書.....287
更新項目.....74,183,195
構成型.....488
構成節.....81,517
項目群名.....114
項類.....158,194
子クラス.....486
固定小数点属性の四則演算の中間結果の精度.....614
固定小数点属性のべき乗の中間結果の精度.....618
固定小数点定数.....11
固定挿入.....164
固定挿入文字.....164
固定長レコード形式.....57
コマンド行の引数の位置付け.....276
コマンド行引数と環境変数の操作機能.....74
コマンド行引数の操作.....75
コレクション.....488
コロン.....4
コンストラクタ.....488
コンマ.....93
コード表.....611

[さ]

最終クラス.....488
最小値.....393
最小値と最大値の算術平均値.....393
最小値を持つ引数の位置.....398
最小レコード長.....57
最大値.....391
最大値から最小値を引いた値.....400
最大値を持つ引数の位置.....398
最大の整数.....386
最大レコード長.....57
作業場所節.....134,149
索引ファイル.....51
索引編成.....120
索引編成のファイル.....52
サブクラス.....486
作用対象の重なり.....254
算術演算.....41
算術演算暫定桁数.....615
算術演算子.....231
算術式.....231,271
算術式の書き方の規則.....232

算術式の評価規則.....232
算術文.....41,252
算術文における複数個の答.....252
算術平均値.....392
参照型.....488
参照キー.....53
サービス名.....455
シグネチャ.....488
システムの定量制限.....608
システム名.....9
四則演算の中間結果.....614
四則演算の中間結果の精度.....617
実行単位.....58
実行単位の実行を終了.....347
実行用計算機段落.....82
実体化.....489
指標データ項目.....45,179
指標比較.....248
指標名.....7,45
指標名の名前の範囲.....63
自由形式の正書法.....36
集合関数指定.....427
修飾.....24
修飾語.....26
集団画面項目.....183
集団項目.....18
集団項目転記.....251
述語.....429
出力項目.....74,183,194
出力手続き.....65,302,339
出力手続きの範囲.....341
主プログラム.....59
主レコードキー.....123
順序位置.....397
順単一リール/ユニット.....269
順ファイル.....51
順ファイルのオープンモードと入出力文の関係.....53
順複数リール/ユニット.....269
順編成.....119,120
順編成のファイル.....52
順呼出し法のファイル.....108
条件完結文.....228
条件式.....234,540
条件の右辺.....235
条件の左辺.....235
条件文.....226
条件変数.....7
条件変数の値の設定.....335
条件名.....7
条件名記述項.....148
条件名条件.....237
条件名の値.....180
条件名の一意参照.....31
条件名を定義.....146
条件を評価.....288
乗算.....305
小数点.....93
小数部基準桁数.....614
小入出力.....48
少量のデータを入力.....257
少量のデータを表示.....274
初期化プログラム.....61,79
初期状態.....267
初期値.....180
除算.....278
処理種別.....121
処理の順序を変更.....262
字類条件.....236,475,540
字類の概念.....18,492
字類名.....7,90
真数定数.....422,457
スイッチ状態条件.....238
数字.....1
数字画面項目.....194
数字関数.....378
数字項目.....159
数字・数字編集転記.....251
数字定数.....11
数字比較.....247
数字編集画面項目.....194
数字編集項目.....160
数値に変換.....396
数定数.....422
スクリーン操作機能.....73,463
スタティック.....489
スタティックコンストラクタ.....489
スタティック段落.....510
スタティック定義.....489,503
スタティックデータ.....489
スタティックメソッド.....489
ストアドプロシージャ.....452
ストアドプロシージャ名.....425
スーパクラス.....486
制御脚書き報告集団.....213
制御頭書き報告集団.....213
制御切れ.....200,206
制御データ項目.....200
制御レコード.....137,625
正弦.....402
正書法.....32,33,501
整数関数.....378
整数項目.....159
整数値.....394
整数部分.....388
正接.....404
静的束縛.....486
正負条件.....238
制約.....489
整列操作.....339
整列操作の最後の段階.....339
整列操作の最初の段階.....339
整列操作または併合操作を中断させる方法.....66
整列の方法.....65
整列併合機能.....64

整列併合用ファイル.....65
整列併合用ファイル記述項.....145
節.....226
節名.....8
ゼロ抑制.....165
ゼロ抑制文字.....165
選択されたレコード.....323,478
選択主体.....282
選択処理と分岐.....42
選択対象.....282
先頭アドレス.....380
全般規定.....1
相関名.....425
早期束縛.....486
総称インタフェース.....489
総称型.....489
総称クラス.....489
総称性.....489
総称デリゲート.....489
総称メソッド.....489
相対ファイル.....51
相対ファイルおよび索引ファイルのオープンモードと入出力文の関係.....53
相対編成.....119
相対編成のファイル.....52
添字付きの修飾されたデータ名.....492
添字付け.....27
束縛.....486
そとPERFORM文.....43,313
ソース単位.....486,502
ソース要素.....486

[た]

大域属性の有無によって名前の範囲が異なる利用者語.....62
大域名.....59,138,154
対数.....390
代入時の適合.....546
タイプオブジェクト.....490
縦の位置.....207
縦の集計.....211
縦の範囲.....200
単一列指定ホスト変数.....418
探索条件.....432
単純挿入.....163
単純挿入文字.....163
単精度内部浮動小数点項目.....179
段落.....226
段落名.....8
遅延束縛.....486
中央値.....392
中核機能.....39
中間結果.....252,614
中間結果の属性と精度.....614
注記.....36
注記行.....33,34,36,458
注記項.....18
通貨編集用文字.....92,163
通信データベース.....454
通信データベース名.....455
通日に変換.....387
次の報告集団用整数退避項目.....217
強い型.....633
強く型付けされた項目.....19
強く型付けされた集団項目.....634
定義したプログラムおよびそれに含まれるプログラムで参照可能な利用者語.....62
定義したプログラムでだけ参照可能な利用者語.....62
定義済みオブジェクト一意名.....487,491,493,496
低輝度で表示.....193
定数.....11,456
定数項目.....74,183
定数項目の値.....197
定数節.....134,149
適合.....487,546
手続き部.....225,461,470,483,537
手続き部の構成.....225
手続き部の見出し.....229,537
手続き分岐文.....228
手続き名.....8
デバッグ行.....33,36,37,82
デリゲート指定子.....521
デリゲート定義.....504
デリゲート名.....491
デリゲート名段落.....514
転記.....304
転記の規則.....249
データ記述項.....146,462,526
データ記述の概念.....18,492
データ項目の初期化.....47
データ項目の表現形式.....215
データ項目の用途.....174
データ項目を初期化.....290
データの境界調整.....20
データの初期状態.....539
データの転記.....41
データ部.....132,458,525,583
データ部の構成.....132
データベース.....416,583
データ名.....8
データ名を再命名.....146
データ名を定義.....146
データを画面に表示.....275,473
テーブル名.....455
問合せ式.....434
問合せ指定.....434
動的CLOSE文.....448
動的DELETE文.....449
動的FETCH文.....448
動的OPEN文.....447
動的SELECT文.....447
動的SQL.....444,584
動的UPDATE文.....449
動的カーソル宣言.....447,584
動的束縛.....486

動的呼出し法のファイル.....108
同等な型.....20
登録集原文.....409
登録集名.....8
特殊挿入.....163
特殊な用途の定数.....17
特殊名段落.....84,464,518
特殊文字.....1
特殊文字語.....10
特殊用途語.....491
特殊用途語.....10
特殊レジスタ.....10,57,64,65,70,77,198
独立データ記述項.....134
閉じた.....489
閉じていない.....489
トークン.....424

[な]

内部10進項目.....179
内部属性.....59
内部ファイル結合子.....60
内部浮動小数点項目.....179
内部プログラム.....37
内部ブール項目.....179
内部名.....60,63,499
名前空間.....490
名前空間や他の型の入れ子となっている場合の修飾.....500
名前付き定数.....462
名前の範囲.....62,63,462,501
二次入口点.....280
二次入口名.....59,63
日本語16進文字定数.....14
日本語画面項目.....194
日本語関数.....379
日本語機能.....475
日本語項目.....160
日本語項目の使用の可否.....419
日本語定数.....14
日本語・日本語編集転記.....251
日本語編集画面項目.....194
日本語編集項目.....160
日本語文字.....2
日本語文字定数.....14
日本語文字比較.....248
日本語利用者語.....6
入出力誤り処理手続きの開始.....365
入出力管理段落.....127,525
入出力機能.....51
入出力状態.....56,69,113
入出力節.....97,524
入出力文.....476
入出力文の動作.....53,69
入出力領域の個数.....124
入力項目.....74,183,195
入力手続き.....65,339
入力手続きの範囲.....340
入力を促す.....191
年日形式の日付に変換.....385

[は]

倍精度内部浮動小数点項目.....179
バイト境界.....23
バイト数.....388
パラメタの受渡し.....61
比較述語.....429
比較条件.....234,542
比較の規則.....245,542
非管理コード.....490
非管理コードプログラム.....490
引数位置指示子.....75
引数に表を指定する場合の規則.....378
引数の値を参照する方法.....75
引数の位置を指定して値を参照する方法.....75
引数の型.....377,585
引数の個数を求める方法.....75
引数の適合.....552
引数のより適切な適合.....567
引数を指定した順に値を参照する方法.....75
非公開クラス.....490
左括弧.....3
日付変換関数.....585
必須語.....10
ビット数.....388
必要語.....10
否定条件.....239
非表示状態.....196
表.....416
表意定数.....15,492
表意定数を識別する語.....11
表式.....432
表示規則表の見かた.....216
標識変数.....425
標識領域.....33
表指定ホスト変数.....419
表示ファイル機能.....67
表示ファイルのオープンモードと入出力文の関係.....69
標準形式の日付に変換.....384
標準桁よせ規則.....20
標準偏差.....403
表操作.....44
表の検索.....46
表引き用の指標名.....333,335
表名.....425
表要素.....44
表要素の指標の設定.....335
表要素の反復回数.....44
表要素を検索.....331
非リール/ユニット.....269
ファイル位置指示子.....54
ファイルが使用可能.....307
ファイルが使用不可能.....307
ファイル管理記述項.....97,99,106
ファイル管理段落.....97
ファイル記述項.....135

ファイル結合子.....53
ファイル識別名.....9
ファイル識別名定数.....17
ファイル終了条件.....56,70
ファイル終了条件以外の例外条件が発生したときの動作...256
ファイル終了条件が発生したときの動作.....256
ファイル終了条件もその他の例外条件も発生しなかったときの動作.....256
ファイル節.....133,149,525
ファイル属性不整合条件.....57
ファイルの共用と排他.....54
ファイルの編成.....51,69
ファイルのロック.....115
ファイル名.....8,124
ファイルを併合.....300
フィールド参照.....493,499
フィールド名.....491
複合条件.....239
複合条件の書き方の規則.....240
複合条件の評価規則.....240
複合条件の評価順序の例.....241
複数行指定ホスト変数.....419
複数の条件を評価.....281
複数列指定ホスト変数.....419
副問合せ.....435
副プログラム.....59
含まれるPERFORM文.....314
含むPERFORM文.....314
副レコードキー.....108
符号系名.....8,88
符号編集用文字.....162
復帰項目.....487
復帰コードの値.....66
不定ファイル.....124
浮動小数点属性の四則演算の中間結果の精度.....618
浮動小数点属性のべき乗の中間結果の精度.....619
浮動小数点定数.....12
浮動小数点転記.....251
浮動小数点数の操作.....49
浮動挿入.....164
浮動挿入文字.....164
部分参照.....28,492
プログラミング機能.....485
プログラム終わり見出し.....80
プログラム間連絡機能.....58,483
プログラム原型.....490
プログラム原型定義.....490,503
プログラム原型名.....491
プログラム識別番号領域.....33
プログラム指定子.....521
プログラム終了文.....59
プログラム定義.....503
プログラムの入れ子.....37
プログラムの終わり.....80
プログラムの共通属性.....60
プログラムの結果.....38
プログラムの構成.....37,502
プログラムの実行の終了.....48
プログラムの初期化属性.....61
プログラムの初期状態.....60
プログラムの名前.....78
プログラムの呼出しと復帰.....58
プログラム名.....8,63
プログラム名段落.....78,507
プログラム名定数.....17
プロパティ指定子.....520
プロパティ名.....487,490
文.....226,256,568
分散.....406
文に関する共通の規則.....231,539
文の組.....313
文の組を繰り返し実行.....311
文の範囲.....228
分離符.....3
分離符号.....455
分離符とみなされない文字列.....4
分離符の空白.....3
分離符のコンマ.....3
分離符の終止符.....3
分離符のセミコロン.....3
ブール演算子.....233
ブール項目.....160
ブール式.....233,271
ブール式の書き方の規則.....233
ブール式の評価規則.....234
ブール定数.....15
ブール転記.....251
ブール比較.....248
併合操作.....302
併合操作の最後の段階.....302
併合操作の最初の段階.....302
併合の方法.....65
平方根.....402
べき乗の中間結果.....618
編集の形式.....158,194
編成.....51
ページ脚書き報告集団.....213
ページ脚書き報告集団の表示規則.....222
ページ頭書き報告集団.....213
ページ頭書き報告集団の表示規則.....219
ページカウンタ.....77,198
ページカウンタの規則.....198
ページの長さ.....200
ページ本体.....140
ページ領域.....202
ポインタ修飾記号.....4,30
ポインタ付け.....30
ポインタデータ関数.....379
ポインタデータ項目.....179
ポインタデータ転記.....251
ポインタデータ比較.....248
ポインタの操作.....48
報告集団記述項.....203
報告集団の種類.....211

報告集団の表示規則....216
報告集団の表示を抑制....355
報告集団を作成表示する直前に実行する手続きの開始....367
報告書脚書き報告集団....213
報告書脚書き報告集団の表示規則....223
報告書頭書き報告集団....213
報告書頭書き報告集団の表示規則....217
報告書記述項....198
報告書作成機能....76
報告書節....134
報告書の処理を開始....292
報告書の処理を終了....355
報告書の制御階層....200
報告書の名前....144
報告書ファイル....76,77,145
報告書名....8
報告書を作成....286
補助語....10
ホスト変数....416
ホスト変数定義....459
ホスト変数の参照....419,460
ホスト変数名....456
ボリューム指示子....54
本体集団....213
本体集団の表示規則....219
翻訳指示完結文....228
翻訳指示文....226
翻訳したときの日付と時刻....407
翻訳集団....487
翻訳単位....37
翻訳単位中のすべてのプログラムで参照可能な利用者語....62
翻訳日付....79
翻訳日付段落....79
翻訳用計算機段落....81

[ま]

毎期の均等払い額....380
前の行....35,36
右括弧....3
見出し部....507
見出し部とプログラム終わり見出し....78
見出し部の構成....78,507
無効キー条件....56
無効キー条件以外の例外条件が発生したときの動作....255
無効キー条件が発生したときの動作....255
無効キー条件もその他の例外条件も発生しなかったときの動作....255
無条件完結文....228
無条件文....226
無名項目....148
明細報告集団....213
明示範囲符....226
明にポインタ付け....31
メソッド....487
メソッド識別名....491
メソッド指定子....521
メソッド定義....487,504
メソッドデータ....487
メソッドの行内呼出し....492,494
メソッドのシグネチャ....488
メソッドの状態....539
メソッド名....487,490
メソッド名段落....510
メソッド名の名前の範囲....502
メソッド呼出し....487
メタデータ....490
文字位置の個数....178
文字集合....88
文字定数....12
文字と文字集合....1
文字の形式....149
文字の大小順序....83,88
文字比較....247
文字列操作....47
文字列定数....423,456
文字列の置き換え....292
文字列の出現回数のカウント....292
文字列を分解....356
文字列を連結....348

[や]

ユニット制御情報....127
用途が表示用の画面項目....197
用途が表示用のデータ項目....178
余弦....383
横の集計....211
横方向の印字位置....167
呼ばれるプログラム....58,264
呼ばれるプログラムの実行の開始....59
呼ばれるプログラムの実行の終了....59
呼ばれるプログラムの論理的な終わり....286
呼出し演算子....487,539
呼出し法....53,69,107
呼び名....8,84,87,88
呼ぶプログラム....58
呼ぶプログラム....264
予約語....9,491
予約語一覧....591
弱く型付けされた項目....19

[ら]

乱呼出し法のファイル....108
リポジトリ段落....520
利用者語....5,462,490
利用者語の記述規則....6
利用者語の用途....7
例外オブジェクト....487,568
例外条件....487,568
レコード....18
レコード記述項....135
レコード記述項の大きさ....143
レコードキー....52
レコード形式....57
レコードの形式....141,143,144
レコードの中の可変位置....44

レコードのロック.....55
レコード名.....8
レコード領域.....57
レコードを整列.....337
列指定.....427
列名.....425
レベルの概念.....18
レベル番号.....8
連結演算子.....4,17
連結式.....16
連絡節.....134,149,526
ロックを解除.....356
論理ページの構成.....139
論理ページの縦方向への行送り.....369