Enterprise PL/I for z/OS

コンパイラーおよびランタイム マイグレーション・ガイド

バージョン *5* リリース *1*

GC43-3420-00

Enterprise PL/I for z/OS

コンパイラーおよびランタイム マイグレーション・ガイド

バージョン *5* リリース *1*

GC43-3420-00

注記

本書および本書で紹介する製品をご使用になる前に、 219 ページの『特記事項』に記載されている情報をお読みください。

本書は、Enterprise PL/I for z/OS バージョン 5 リリース 1 (5655-PL5)、および新しい版またはテクニカル・ニュースレターで明記されていない限り、以降のすべてのリリースに適用されます。製品のレベルに合った正しい版をご使用ください。

資料のご注文方法については、http://www.ibm.com/jp/manuals の「ご注文について」をご覧ください。 (URL は変更になる場合があります)

IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、自ら適切と信ずる方法で、使用もしくは配布することができるものとします。

お客様の環境によっては、資料中の円記号がバックスラッシュと表示されたり、バックスラッシュが円記号と表示されたりする場合があります。

原典： GC27-8930-00
Enterprise PL/I for z/OS
Compiler and Run-Time Migration Guide
Version 5 Release 1
First Edition (June 2016)

発行： 日本アイ・ビー・エム株式会社

担当： トランスレーション・サービス・センター

© Copyright IBM Corporation 1999, 2016.

目次

表 ix

図 xi

本書について xiii
資料の使用 xiii
ご意見の送付方法 xiv
アクセシビリティー xv

第 1 部 概説 1

第 1 章 再コンパイルの必要性 3
マイグレーションについての基本事項 4
ランタイムのマイグレーション - 言語環境プログラムへの移行 4
コンパイラーのマイグレーション 4
マイグレーション・ロードマップ 5
OS PL/I および PL/I for MVS & VM に対するサービス・サポート. 6

第 2 章 新しいコンパイラーおよびランタイムの紹介. 7
製品間の関係 - コンパイラー、ランタイム、Debug Tool 7
PL/I コンパイラーの一般情報 7
他のプログラムに対する言語環境プログラムのランタイム・サポート 8
新しいコンパイラーおよびランタイムの利点. . . . 8
新しいコンパイラーおよびランタイムの主な変更点 10
一般的な移行作業 11
戦略の計画 11
言語環境プログラム・ランタイムへの移行 . . . 11
Enterprise PL/I によるソースの再コンパイル. . 11
Enterprise PL/I プログラムを既存アプリケーションに追加 12

第 2 部 移行の戦略. 13

第 3 章 言語環境プログラムへの移行の計画 15
言語環境プログラム・ランタイム・ライブラリーへの移行を準備. 15
言語環境プログラムのインストール 15
ストレージ要件の評価. 16
言語環境プログラムについてのプログラマーの教育 16
アプリケーションのインベントリーの作成 17
ベンダー製のツール、パッケージ、および製品 . 17
PL/I アプリケーション 17
既存の PL/I ロード・モジュール 18
段階的に言語環境プログラムに移行する方法の決定 19
複数言語の変換処理 19
アプリケーションがライブラリーにアクセスする方法の決定 19
レグレッション・テスト手順のセットアップ . . . 22
パフォーマンス測定の実行 23
実動使用への切り換え. 24

第 4 章 新しいコンパイラーへの移行の計画 25
新しいコンパイラーへのソースの移行の準備 . . . 25
Enterprise PL/I のインストール 25
ストレージ要件の評価. 25
新しいコンパイラーの機能についてのプログラマーの教育 26
アプリケーションのインベントリーの作成 26
ベンダー製ツール、パッケージ、および製品のインベントリーの作成 27
PL/I アプリケーションのインベントリーの作成 27
アプリケーションの優先順位付け 27
移行の優先順位の決定. 28
移行カテゴリーと非移行カテゴリーのセットアップ 28
アプリケーション・プログラムの更新 28

第 3 部 既存アプリケーションの言語環境プログラムへの移行. 31

第 5 章 言語環境プログラムの下での既存アプリケーションの実行. 33
既存のアプリケーションの起動. 33
CICS 以外のアプリケーションに関する考慮事項 33
CICS アプリケーションに関する考慮事項 . . . 34
既存のアプリケーションのリンク・エディット . . 34

第 6 章 マイグレーション前の考慮事項 37
ランタイム・オプションにおける相違点. 37
削除されたランタイム・オプション 37
置き換えられたランタイム・オプション. . . . 37
新しいランタイム・オプション. 38
条件処理における相違点 40
タイミングの相違点 40
一部の高精度固定小数点除算の実行における相違点 41
未処理条件の相違点 41
IBMBXITA および IBMBEER の相違点 41
ABEND U4039 の相違点. 41
重大度の相違点 42
PLICALLA と PLICALLB のサポートの相違点 . . 42
PLICALLA に関する考慮事項 42

© Copyright IBM Corp. 1999, 2016

PLICALLB に関する考慮事項 43
事前初期設定サポートにおける相違点 45
PLISRTx のサポートの相違点 46
マルチタスキング・サポートにおける相違点 . . . 46
OS PL/I 共用ライブラリーのサポートの相違点 . . 46
DATE/TIME 組み込み関数における相違点. . . . 46
ユーザー戻りコードの相違点 47
ランタイム・メッセージにおける相違点. 47
PLIDUMP の相違点 48
ストレージ報告書における相違点 49
言語間通信 (ILC) のサポートの相違点 49
アセンブラー・サポートにおける相違点. 51

第 7 章 オブジェクト・モジュールおよびロード・モジュールに関する考慮事項 . . 53
OS PL/I バージョン 1 のオブジェクト・モジュールおよびロード・モジュールの互換性 53
OS PL/I バージョン 1 リリース 5.1. 53
OS PL/I バージョン 1 リリース 5 55
OS PL/I バージョン 1 リリース 3.0 ～ リリース 4.0 55
バージョン 1 リリース 3.0 より前の OS PL/I 56
OS PL/I バージョン 2 のオブジェクト・モジュールおよびロード・モジュールの互換性 56
OS PL/I オブジェクト・モジュールおよびロード・モジュールのサポートの要約 56

第 8 章 リンク・エディットに関する考慮事項 59
SCEERUN. 59
シンボル・テーブルに関する考慮事項 59
NCAL リンケージ・エディター・オプション . . . 60
ENTRY カード 60
OS PL/I 数学ルーチンの使用 60

第 9 章 サブシステムに関する考慮事項 61
CICS に関する考慮事項 61
CICS システム定義 (CSD) ファイルの更新. . . 61
エラー処理 62
CICS 環境でのユーザー作成条件ハンドラーの制限 62
マクロ・レベル・インターフェース 62
PL/I MAIN プロシージャーの FETCH 62
STACK ランタイム・オプション 62
ランタイム出力 63
CICS 環境で PL/I によって使用される異常終了コード 63
IMS に関する考慮事項 63
IMS へのインターフェース 63
SYSTEM(IMS) コンパイル時オプション 63
IMS での PLICALLA サポート 64
サポートされている PSB 言語オプション . . . 64
ストレージの使用に関する考慮事項 64
IMS 環境での調整条件処理 65
ライブラリー保存 (LRR) によるパフォーマンスの向上 66
DB2 に関する考慮事項 66

第 4 部 新しいコンパイラーへの移行 67

第 10 章 新しいコンパイラーの制限について 69
言語環境プログラム要件 69
サポートされない言語. 69
言語制限 70
RECORD I/O 70
STREAM I/O 71
構造式 71
配列式 71
DEFAULT ステートメント 72
自動変数の範囲 72
組み込み関数. 72
DEFINED BIT 集合体. 73
OPTIONS(REENTRANT) 73
iSUB 定義. 73
LABEL 配列 73
DBCS 73
GRAPHIC および POSITION 73
マクロ・プリプロセッサー 74
オプション制限 74
サポートされないオプション 75
コンパイラーに対するその他のインターフェースに関する制約事項 75
バッチ・コンパイル 75
アセンブラーからのコンパイラーの起動. . . . 76
TSO 環境でのコンパイル. 76
INCLUDE データ・セット名の指定 77
SYSLIN データ・セットの指定. 77
コンパイル時の時間およびスペース所要量 77
AMODE(24) の制約事項 78
EXTERNAL 名の制限 78
リスト表示の相違点 79
制御ブロックの相違点. 79
ISAM サポートの相違点 79

第 11 章 新しいコンパイラーのオプションについて 81
互換性におけるデフォルト・オプションの効果について. 81
BACKREG(5) 81
BIFPREC(15) 82
CMPAT(V2) 83
EXTRN(FULL) 83
LIMITS(EXTNAME(7)) 83
NORENT および WRITABLE 84
SYSTEM 85
互換性をさらに向上させるためのデフォルト以外のオプションの選択 85
COMMON 85
DFT(NOBIN1ARG) 85
DEFAULT(LINKAGE(SYSTEM)) 86

DFT(OVERLAP) 86
NOREDUCE 86
NORESEXP 86
RULES(LAXCTL) 87
RULES(NOLAXINOUT NOLAXSEMI) 87
NOWRITABLE 87
パフォーマンスを向上させるためのオプションの選択 88
ARCH 88
BIFPREC(31) 88
DEFAULT(NONASGN) 89
DEFAULT(CONNECTED) 89
DEFAULT(REORDER) 89
DEFAULT(NOOVERLAP) 90
OPTIMIZE(2)/OPTIMIZE(3) 90
REDUCE 90
NORENT 91
RULES(NOLAXCTL) 91
品質を向上させるためのオプションの選択 92
RULES(NOLAXDCL) 92
RULES(NOLAXIF) 93
RULES(NOLAXLINK) 94
RULES(NOLAXMARGINS) 94
RULES(LAXSTRZ) 94
RULES(NOMULTICLOSE) 95
テスト用のオプションの選択 95
CHECK(CONFORMANCE) 96
GONUMBER 96
PREFIX 96
TEST 96

第 12 章 新しいコンパイラーのメッセージについて 97

IBM1044: 1 バイトの FIXED BIN 97
IBM1053: スケール付き FIXED BIN の評価 . . . 97
IBM1065: 不正確な浮動小数点定数 98
IBM1091: FIXED BIN 精度の警告 98
IBM1099: 混合した FIXED 98
IBM1181: 誤ってコーディングされた DO ループ 100
IBM1206: BIT 演算子の誤用 100
IBM1208: 完全には初期化されていない配列 . . . 101
IBM1215: 不完全な宣言 102
IBM1216: 間違った構造体宣言 102
IBM1220: 無意味な比較 103
IBM1927: SIZE 条件 103
IBM1948: 制限された式評価 104
IBM2063: 無効な ALLOCATE 104
IBM2402: ストレージ・オーバーレイ 104
IBM2409: 関数内での RETURN; 105
IBM2410: 関数内に RETURN がない 105
IBM2412: RETURNS オプションの欠落 106
IBM2421: ENDFILE での CLOSE 106
IBM2610: 精度の解釈 106
IBM2611, IBM2612: 重複する WHEN 107
IBM2617: PL/I 以外のラベルの引き渡し 108
IBM2621: ON ERROR SYSTEM の欠落 108
IBM2622: 不完全にコーディングされた DO ループに対する警告 108
IBM2626: 長さゼロを持つ SUBSTR 109
IBM2628: 大きな BYVLAUE パラメーター . . . 109
IBM2801: スケール付き FIXED BIN の導入 . . . 109
IBM2804: 完全に最適化されていない比較 110
IBM2810: スケール付き FIXED BIN の変換 . . . 110
IBM2811: DO 制御変数としての PICTURE の使用 111
IBM2812: 不完全な TRANSLATE および VERIFY 111
PLIXOPT メッセージ 111
コンパイラー・ユーザー出口の使用 112

第 13 章 作業コードを変更する必要がある場合について 113

間違ったコード 113
宣言の順序を当てにする 113
無効な FIXED DECIMAL データを使用する 114
無効な SUBSTR 参照を使用する 114
異なる EXTERNAL 宣言を使用する 115
間違った PLITABS 宣言を使用する 115
変数を初期化する 115
AUTOMATIC の初期設定 115
BASED の初期設定 116
CONTROLLED の初期設定 116
STATIC の初期設定 116
使用されない宣言の保持 117
使用されない INTERNAL STATIC の保持 . . 117
例外が発生するようになった間違ったコード . . . 117
SIZE が無効になっている場合の FIXEDOVERFLOW 117
無効な割り振り 119
無限ループが発生するようになった間違ったコード 119
偶数精度の PICTURE ループ制御変数 120
異なる結果を生成する代入 121
代入元と代入先の重複 121
float 間の代入 122
異なる結果を生成するその他のステートメント . . 123
印刷不能な文字が含まれた STREAM 入出力 124
初期化されていない EXTERNAL STATIC . . . 124
不完全に宣言された FILE 124
仮引数と配置 125
仮引数と CONTROLLED 125
ポインター算術 126
パフォーマンスが劣るコード 126
FIXED DEC をループ制御変数として使用 . . . 126
FIXED BIN(15) をループ制御変数として使用 126
TOTAL を使用した入出力 126

第 14 章 作業コードを変更する必要がある可能性のある場合について 127

例外が発生するようになったコード 127
ERROR にプロモートされる ZERODIVIDE および OVERFLOW 127
使用不可の場合に発生する条件 127
無効な RETURN 128
GOTO の欠点 128

NOFOFL のスコープ 129
例外が発生しなくなったコード 129
FIXED BIN について発生する FIXEDOVERFLOW 129
数値変数にブランクを代入する際に生じる CONVERSION 130
過度に大きな集合体をマッピングした際に生じる ERROR 130
異なる方法でマップされるストレージ 130
1 バイトの FIXED BIN 130
異なる方法で処理される宣言 131
INITIAL 属性を持つ AREA 131
異なる方法で処理される変換 132
FLOAT から文字への変換 132
スケール付き FIXED BINARY からの変換 . . 132
異なる方法で処理される組み込み関数 133
スケール係数および FIXED BIN を持つ算術組み込み関数 133
DBCS 文字ストリングの変換用ストリング処理組み込み関数 134
MACRO プリプロセッサーの相違点. 135
SQL プリプロセッサーの相違点 136

第 15 章 新しいオブジェクトのリンク 137
プリリンカーと PDSE に関する考慮事項 137
AMODE(24) に関する考慮事項 137
PLICALLA または PLICALLB エントリーの使用 137
CHANGE カード 137

第 16 章 言語環境プログラムの新しいコンパイラーとの使用 139
適切なランタイム・オプションの使用 139
アセンブラーのメインプログラムからの PL/I の呼び出し. 140
結果が異なる可能性がある場合について 140
戻りコード 140
ランタイムにメッセージが発行される場合. . . 141
ランタイム・メッセージの意味 141
ランタイム・メッセージの出力先. 142
数学組み込み関数 142
ダンプ. 143
ストレージ報告書 143
前提条件言語環境プログラム PTF 143

第 17 章 CPU とストレージの使用効率を向上させるためのチューニング. . . . 145
CPU 使用効率の向上. 145
ストレージ使用効率の向上 146
サブシステム環境でのパフォーマンス向上. . . . 147

第 18 章 既存の PL/I アプリケーションへの Enterprise PL/I プログラムの追加. 149
オブジェクト・モジュールおよびロード・モジュールに関する考慮事項 149
SYSPRINT の共用. 150
ランタイム・オプションの考慮事項 152
条件処理に関する考慮事項 152
PL/I ソース・プログラムの実行単位への区分化 152

第 19 章 旧リリースの Enterprise PL/I から Enterprise PL/I V5R1 へのマイグレーション. 153
Enterprise PL/I V4R5 からのマイグレーション 153
Enterprise PL/I V4R4 からのマイグレーション 155
Enterprise PL/I V4R3 からのマイグレーション 156
Enterprise PL/I V4R2 からのマイグレーション 157
Enterprise PL/I V4R1 からのマイグレーション 159
Enterprise PL/I バージョン 3 (すべてのリリース) からのマイグレーション. 162
V5R1 で導入されたメッセージ 164
V4R5 で導入されたメッセージ 165
V4R4 で導入されたメッセージ 167
V4R3 で導入されたメッセージ 168
V4R2 で導入されたメッセージ 170
V4R1 で導入されたコンパイラー・メッセージ . . 173
V3R9 で導入されたコンパイラー・メッセージ . . 173
V3R8 で導入されたコンパイラー・メッセージ . . 174
V3R7 で導入されたコンパイラー・メッセージ . . 175
V3R6 で導入されたコンパイラー・メッセージ . . 176
V3R5 で導入されたコンパイラー・メッセージ . . 177
V3R4 で導入されたコンパイラー・メッセージ . . 177
オブジェクトの互換性 179
ランタイムの変更 181

第 5 部 サブシステムおよびその他の言語に関する考慮事項 183

第 20 章 PL/I アプリケーションに関するアセンブラーの考慮事項 185
PL/I メイン・プロシージャーを模倣したアセンブラー・プログラムに関する考慮事項 185
アセンブラーおよび言語環境プログラムに準拠したアセンブラーからの PL/I の呼び出し 185
条件処理およびアセンブラー・プログラム. . . . 186
アセンブラー・ユーザー出口の使用に関する考慮事項 186

第 21 章 PL/I アプリケーションに関する CICS の考慮事項. 189
CICS に関する一般的な考慮事項 189
CICS システム定義 (CSD) ファイルの更新 . . 189
マクロ・レベル・インターフェース 190
CICS 環境で実行されるプログラム用のコンパイラー・オプション. 190
CICS アプリケーションのリンクとランタイムに関する考慮事項 190
エラー処理 190
PL/I MAIN プロシージャーの FETCH . . . 191
ランタイム出力. 191

CICS 環境で PL/I によって使用される異常終了コード. 191
統合 CICS プリプロセッサーへのマイグレーション 191

第 22 章 PL/I アプリケーションに関する IMS の考慮事項 193
IMS へのインターフェース. 193
SYSTEM(IMS) コンパイル時オプション 193
IMS での PLICALLA サポート 193
サポートされている PSB 言語オプション. . . . 194
ストレージの使用に関する考慮事項 194
IMS 環境での調整条件処理. 194
ライブラリー保存 (LRR) によるパフォーマンスの向上 195

第 23 章 PL/I アプリケーションに関する DB2 の考慮事項 197
DB2 に関する一般的な考慮事項 197
統合 SQL プリプロセッサーへのマイグレーション 197
プログラミングとコンパイルに関する考慮事項 197
FOR BIT DATA への代入に関する注記 . . . 199
前提条件 DB2 APAR. 199

第 6 部 付録 201

付録 A. 変換とマイグレーションの支援機能 203
OS PL/I ルーチン置換ツール. 203
OS PL/I バージョン 1 リリース 5.1 メイン・ロード・モジュール ZAP. 204
OS PL/I 共用ライブラリー置換ツール. 205
OS PL/I オブジェクト・モジュール再リンク・ツール - APAR PN69803 205
ILC アプリケーション 205
PLISRTx アプリケーション. 206
EDGE Portfolio Analyzer 207
ベンダー製品 207

付録 B. コンパイラー・エレメントの比較 209

付録 C. コンパイラーの制限の比較. . . 211

付録 D. バッチ処理のサンプル 215

付録 E. デバッグ・ツールの比較. . . . 217
デバッグ・ツール間の相違点 217

特記事項. 219
プログラミング・インターフェース情報 220
商標 220

参考文献. 221
PL/I の資料. 221
関連資料 221

索引 225

表

1. Enterprise PL/I 付属資料の使用方法 xiii
2. z/OS 言語環境プログラム 付属資料の使用方法 xiv
3. PL/I コンパイラーの IDR 値 19
4. 新規 DD 名の仕様 34
5. SPIE および STAE オプションの TRAP オプションとの対応 38
6. OS PL/I バージョン 2 リリース 3の ERROR ON ユニットと ERROR 条件のメッセージ . . 40
7. 言語環境プログラムの ERROR ON ユニットと ERROR 条件のメッセージ 40
8. PLICALLB 引数リスト・サポートにおける相違点. 43
9. 言語環境プログラムでの戻りコードの振る舞い 47
10. 言語環境プログラムによるオブジェクト・モジュールおよびロード・モジュールのサポートの有無についての要約. 56
11. IMS/ESA バージョン 4 リリース 1 以降の PSB LANG オプション 64
12. IMS/ESA バージョン 4 リリース 1 以降の PSB LANG オプション 194
13. PL/I のエレメント名 209
14. 言語エレメントの制限 211
15. PLITEST コマンドおよびこれらに対応する Debug Tool コマンド. 217

© Copyright IBM Corp. 1999, 2016

図

1. CESE の出力データ・キュー. 63
2. CESE の出力データ・キュー 191

© Copyright IBM Corp. 1999, 2016

本書について

本書には、言語環境プログラム以前のランタイム・ライブラリーを IBM Language Environment for z/OS に移行するとき、およびソース・プログラムを IBM Enterprise PL/I for z/OS バージョン 5 リリース 1 にアップグレードするときに役立つ情報が記載されています。また本書では、PL/I (OS PL/I、PL/I for MVS & VM、および VisualAge PL/I) の旧リリースと Enterprise PL/I とのサポートの相違が原因で生じる問題の解決策を示しています。

重要: 本書では、Enterprise PL/I V5R1M0 および z/OS V2R1 言語環境プログラム、またはそれ以降を使用する場合のマイグレーションに関する考慮事項について説明します。本書で説明するマイグレーション用の拡張機能を利用するには、これら 2 つの製品がインストールされている必要があります。 Enterprise PL/I という用語は、特に指示のない限り バージョン 5 リリース 1 を指します。言語環境プログラムという用語は、特に指示のない限り V2R1 以降の z/OS 言語環境プログラムまたはそれ以降を指します。

本書は、PL/I プロダクトのマイグレーションを行うシステム・プログラマー、アプリケーション・プログラマー、および IBM サポート担当員を対象としています。本書を使用するにあたって、前提条件として以下の知識が必要になります。

- ご使用のオペレーティング・システムに関する全般的な知識
- PL/I 言語およびオプションに関する若干の知識
- PL/I がランタイム環境として言語環境プログラムを使用する方法に関する若干の知識

資料の使用

Enterprise PL/I に付属の資料は、PL/I でのプログラミングに役立つことを目的としています。言語環境プログラムに付属の資料は、Enterprise PL/I で生成されたアプリケーションのランタイム環境の管理を支援することを目的としています。資料はそれぞれ、異なる作業を支援するものです。

下の表は、Enterprise PL/I および言語環境プログラムの資料の使用方法を示しています。 使用するコンパイラーおよびランタイム環境についての情報を知ることができます。これらの資料および関連資料の正式名称および資料番号については、 221 ページの『参考文献』を参照してください。

PL/I 情報

表 *1. Enterprise PL/I* 付属資料の使用方法

目的...	使用...
Enterprise PL/I の評価	Fact Sheet
保証情報の理解	Licensed Programming Specifications
Enterprise PL/I の計画とインストール	Enterprise PL/I プログラム・ディレクトリー

© Copyright IBM Corp. 1999, 2016

表 1. *Enterprise PL/I* 付属資料の使用方法 (続き)

目的...	使用...
コンパイラーおよびランタイム変更作業の理解と、プログラムの Enterprise PL/I と言語環境プログラムへの適合	コンパイラーおよびランタイム 移行ガイド
プログラムの準備とテスト、およびコンパイラー・オプションについての詳細情報の入手	プログラミング・ガイド
PL/I の構文および言語エレメントの仕様についての詳細情報の入手	言語解説書
コンパイラーの問題診断および IBM への連絡	診断ガイド
コンパイル時メッセージについての詳細情報の入手	メッセージおよびコード

言語環境プログラム情報

表 2. *z/OS* 言語環境プログラム 付属資料の使用方法

目的...	使用...
言語環境プログラムの評価	概念
言語環境プログラムの計画	概念 ランタイム マイグレーション・ガイド
言語環境プログラムを z/OS にインストール	z/OS Program Directory
言語環境プログラムを z/OS でカスタマイズ	カスタマイズ
言語環境プログラムのプログラム・モデルおよび概念の理解	概念 プログラミング・ガイド
言語環境プログラムランタイム・オプションおよび呼び出し可能サービスの構文の検索	プログラミング・リファレンス
言語環境プログラムで実行されるアプリケーションの開発	プログラミング・ガイド、および使用している言語のプログラミング・ガイド
言語環境プログラムで実行されるアプリケーションのデバッグ、ランタイム・メッセージに関する詳細情報の入手、言語環境プログラムでの問題の診断	デバッグのガイドとランタイム・メッセージ
言語間通信 (ILC) アプリケーションの開発	ILC アプリケーションの作成
言語環境プログラムへのアプリケーションの移行	「ランタイム・アプリケーション マイグレーション・ガイド」、および言語環境プログラムで使用できる各言語の移行ガイド

ご意見の送付方法

本書または PL/I の他のマニュアルについてご意見がありましたら、IBM 発行のマニュアルに関する情報の Web ページ (http://www.ibm.com/jp/manuals/) よりお送りください。今後の参考にさせていただきます。(URL は、変更になる場合があります)

アクセシビリティー

アクセシビリティー機能は、運動障害または視覚障害など身体に障害を持つユーザーがソフトウェア・プロダクトを快適に使用できるようにサポートします。 z/OS のアクセシビリティー機能は、Enterprise PL/I のアクセスを支援します。

アクセシビリティー機能

z/OS には、以下の主要なアクセシビリティー機能が用意されています。

- スクリーン・リーダーおよびスクリーン拡大ソフトウェアで一般的に使用されるインターフェース
- キーボードのみによるナビゲーション
- 色、コントラスト、フォント・サイズなどの表示属性をカスタマイズする機能

z/OS では、US Section 508 (http://www.access-board.gov/guidelines-and-standards/communications-and-it/about-the-section-508-standards/section-508-standards) および Web Content Accessibility Guidelines (WCAG) 2.0 (http://www.w3.org/TR/WCAG20/) に確実に準拠するために、最新の W3C 標準である WAI-ARIA 1.0 (http://www.w3.org/TR/wai-aria/) を使用します。アクセシビリティー機能を利用するには、最新リリースのスクリーン・リーダーを、この製品でサポートされる最新の Web ブラウザーと併用してください。

IBM® Knowledge Center の Enterprise PL/I オンライン製品資料は、アクセシビリティーに対応しています。 IBM Knowledge Center のアクセシビリティー機能については、http://www.ibm.com/support/knowledgecenter/doc/kc_help.html#accessibility に説明があります。

キーボード・ナビゲーション

ユーザーは、TSO/E または ISPF を使用して z/OS ユーザー・インターフェースにアクセスできます。

ユーザーはまた、IBM Rational® Developer for System z® を使用して、z/OS サービスにアクセスすることもできます。

これらのインターフェースへのアクセスに関する情報については、以下の資料を参照してください。

- *z/OS TSO/E Primer* (http://publib.boulder.ibm.com/cgi-bin/bookmgr/BOOKS/ikj4p120)
- *z/OS TSO/E User's Guide* (http://publib.boulder.ibm.com/cgi-bin/bookmgr/BOOKS/ikj4c240/APPENDIX1.3)
- *z/OS ISPF User's Guide Volume I* (http://publib.boulder.ibm.com/cgi-bin/bookmgr/BOOKS/ispzug70)
- IBM Rational Developer for System z Knowledge Center (http://www.ibm.com/support/knowledgecenter/SSQ2R2/rdz_welcome.html?lang=en)

上記の資料には、キーボード・ショートカットまたはファンクション・キー (PF キー) の使用方法を含む TSO/E および ISPF の使用方法が記載されています。 それ

ぞれの資料では、PF キーのデフォルトの設定値とそれらの機能の変更方法についても説明しています。

インターフェース情報

Enterprise PL/I のオンライン製品資料は、IBM Knowledge Center で入手でき、標準の Web ブラウザーで表示できます。

PDF ファイルでのアクセシビリティー・サポートは限定的です。 PDF 資料では、オプションのフォント拡大機能およびハイコントラスト表示設定を使用でき、キーボードのみでナビゲートできます。

ご使用のスクリーン・リーダーが構文図やソース・コード例を正確に読み上げたり、ピリオドやコンマといった PICTURE 記号を含むテキストを正確に読み上げたりできるようにするには、ご使用のスクリーン・リーダーがすべての句読点を読み上げるように設定する必要があります。

支援技術製品は、z/OS に備わっているユーザー・インターフェースと連動します。特定のガイダンス情報については、z/OS インターフェースのアクセスに使用する支援テクノロジー製品の資料を参照してください。

アクセシビリティー関連情報

標準の IBM ヘルプ・デスクとサポート Web サイトに加え、IBM は、聴覚が不自由なお客様が営業やサポート・サービスにアクセスするために使用できる TTY 電話サービスを立ち上げました。

TTY サービス
800-IBM-3383 (800-426-3383)
(北アメリカ内)

IBM およびアクセシビリティー

IBM のアクセシビリティーに対する取り組みについては、IBM Accessibility (www.ibm.com/able) を参照してください。

第 1 部 概説

© Copyright IBM Corp. 1999, 2016

第 1 章 再コンパイルの必要性

理想的には、プログラムは IBM Enterprise PL/I for z/OS を使用してコンパイルし、サポートされるランタイム・ライブラリー (言語環境プログラム) を使用して実行する必要があります。この理想的な状態に到達するために、最初にランタイム、次にコンパイラーの順で段階的にマイグレーションします。この章では、アプリケーション (ランタイムまたはソース) のマイグレーションがいつ必要か、なぜ必要かについて説明します。

用語説明

本書を使用するときには、以下の用語に注意してください。

Enterprise PL/I
: 一般に、IBM Enterprise PL/I for z/OS バージョン 5 リリース 1 を指します。

PL/I
: 一般に、以下の製品を指します。
- OS PL/I
- PL/I for MVS & VM
- VisualAge PL/I
- Enterprise PL/I

新しい *PL/I* コンパイラー
: 以下のコンパイラー製品を指します。
- VisualAge PL/I
- Enterprise PL/I

従来の *PL/I* コンパイラー
: 以下のコンパイラー製品を指します。
- OS PL/I V3R2 およびそれ以前
- PL/I for MVS & VM

マイグレーションに関する重要な注記

まず始めに、従来の PL/I コンパイラーと新しい PL/I コンパイラーは、まったく違うものであるということを理解することが重要です。新しい PL/I コンパイラーは PL/I で書かれており、従来の PL/I コンパイラーで使用されたいくつかの技法は使用していません。実際、これらはあまりに異なるので、言語環境プログラムの観点からは別々の言語として捉えられており、それぞれ独自のシグニチャー CSECT を持っています。

今までは、「旧版」PL/I コンパイラーから別の「従来の」PL/I コンパイラーにマイグレーションすることは、さほど難しいことではありませんでした。しかし、「新しい」Enterprise PL/I コンパイラーの登場により、マイグレーション・プロセスは以前よりはるかに複雑になりました。 新しい Enterprise PL/I コンパイラーにマイグレーションする際は、スムーズに移行するためにプロジェクトの十分な調査、計画、および実行を行う必要があります。

© Copyright IBM Corp. 1999, 2016

マイグレーションについての基本事項

マイグレーション・プロセスには、ランタイムのマイグレーション (新しいランタイムへのアプリケーションのマイグレーション) およびコンパイラーのマイグレーション (新しいコンパイラーによる、ソース・プログラムのコンパイル) が含まれます。

マイグレーション・プロセスの一部として、インベントリーの評価およびテストも行う必要があります。ランタイムとソースを同時にマイグレーションする必要はありません。

マイグレーション・プロセスについての詳細は、 11 ページの『一般的な移行作業』を参照してください。

インベントリーの評価およびテスト計画の実施については、 17 ページの『アプリケーションのインベントリーの作成』を参照してください。

ランタイムのマイグレーション - 言語環境プログラムへの移行

すべての PL/I プログラムは、実行するためにランタイム・ライブラリー・ルーチンが必要です。

実行時に、複数の PL/I タイム・ライブラリーをアプリケーションで使用可能にしてはいけません。例えば LNKLST には、言語環境プログラム用の SCEERUN などの PL/I ランタイム・ライブラリーが 1 つだけ含まれるようにする必要があります。複数のライブラリーが存在すると、ライブラリーが見つからないというエラーや、連結内に使用されないロード・ライブラリーが含まれるという事態が発生します。それに加えて、連結内に複数のランタイム・ライブラリーが存在する構成は、IBM ではサポートしない無効な構成になります。

まだ言語環境プログラムに移行しておらず、OS PL/I V2R3 などの、言語環境プログラム以前の PL/I コンパイラーを使用している場合には、 15 ページの『第 3 章 言語環境プログラムへの移行の計画』を参照してください。

すでに 言語環境プログラムに移行しており、新しい IBM Enterprise PL/I for z/OS コンパイラーにマイグレーションする予定である場合は、 25 ページの『第 4 章 新しいコンパイラーへの移行の計画』でコンパイラーのマイグレーションについて参照してください。

コンパイラーのマイグレーション

すべてのソースについて、新しい Enterprise PL/I コンパイラーで再コンパイルすることを強くお勧めします (すでにすべてのソースを VisualAge PL/I で再コンパイルしている場合を除く)。 Enterprise PL/I コンパイラーは、従来の PL/I コンパイラーとは完全に異なるコンパイラーであるため、ソースを再コンパイルすることが、従来の PL/I オブジェクトおよびロード・モジュールを新しい PL/I オブジェクトおよびロード・モジュールと混合使用するときに課せられる制限事項を回避する最良の手段です。

コンパイラーのマイグレーションは、すべてを一度に行うか、または個々の実行単位ごとに行うことができます。 PL/I ソースを個々の実行単位に分割する方法については、 152 ページの『PL/I ソース・プログラムの実行単位への区分化』を参照してください。

旧版 PL/I モジュールと Enterprise PL/I モジュールを混在させることは、制限された環境下でのみ可能です。詳しくは、 149 ページの『オブジェクト・モジュールおよびロード・モジュールに関する考慮事項』を参照してください。

OS PL/I から Enterprise PL/I に移行する場合、コードを若干変更する必要が生じるケースがあります。詳しくは、 113 ページの『第 13 章 作業コードを変更する必要がある場合について』および 127 ページの『第 14 章 作業コードを変更する必要がある可能性のある場合について』を参照してください。

すでに 言語環境プログラムに移行しており、新しい IBM Enterprise PL/I for z/OS コンパイラーにマイグレーションする予定である場合は、 25 ページの『第 4 章 新しいコンパイラーへの移行の計画』で新しいコンパイラーへのマイグレーションについて参照してください。

PL/I for MVS & VM コンパイラーにマイグレーションする場合は、「*IBM PL/I for MVS & VM* コンパイラーおよびランタイム 移行ガイド」を参照してください。

マイグレーション・ロードマップ

このトピックには、マイグレーションの実現性に関する簡単な要約があります。

OS PL/I または PL/I for MVS & VM からのマイグレーション

- 現在、言語環境プログラムにマイグレーションしていない場合、まず言語環境プログラムにマイグレーションしてから、Enterprise PL/I for z/OS にマイグレーションすることができます。この場合、 15 ページの『第 3 章 言語環境プログラムへの移行の計画』から始めて、 31 ページの『第 3 部 既存アプリケーションの言語環境プログラムへの移行』に進んでください。
- OS PL/I からマイグレーションする場合、まず PL/I for MVS & VM にマイグレーションするようお勧めします。この場合、「*IBM PL/I for MVS & VM* コンパイラーおよびランタイム 移行ガイド」の指示に従う必要があります。
- 既に言語環境プログラムにマイグレーションしている場合、Enterprise PL/I for z/OS にマイグレーションできます。この場合、 7 ページの『第 2 章 新しいコンパイラーおよびランタイムの紹介』から始めて、 25 ページの『第 4 章 新しいコンパイラーへの移行の計画』および 67 ページの『第 4 部 新しいコンパイラーへの移行』に進んでください。

VisualAge PL/I または旧リリースの Enterprise PL/I からのマイグレーション

67 ページの『第 4 部 新しいコンパイラーへの移行』を参照してください。特に 153 ページの『第 19 章 旧リリースの Enterprise PL/I から Enterprise PL/I V5R1 へのマイグレーション』に注意してください。

サブシステムについて詳しくは、 183 ページの『第 5 部 サブシステムおよびその他の言語に関する考慮事項』を参照してください。

OS PL/I および PL/I for MVS & VM に対するサービス・サポート

OS PL/I コンパイラーでコンパイルされたプログラムが PL/I ライブラリー・ルーチンの 言語環境プログラム・ランタイム・ライブラリーのバージョンを使用している場合、IBM では、これらのプログラムの実行に対するサービス・サポートの提供を継続します。

注: CICS TS (Transaction Server) バージョン 2 リリース 2 より後の CICS TS リリースでは、OS PL/I モジュールをサポートしていません。 CICS TS V2R2 より後の CICS を使用するには、OS PL/I から 言語環境プログラム 対応の PL/I コンパイラーに移行する必要があります。

このサポートとその制約事項について詳しくは、 53 ページの『第 7 章 オブジェクト・モジュールおよびロード・モジュールに関する考慮事項』を参照してください。

第 2 章 新しいコンパイラーおよびランタイムの紹介

この章では、Enterprise PL/I コンパイラー (IBM Enterprise PL/I for z/OS) および共通ランタイム (言語環境プログラム) の概要を示し、本書全体を通じて使用される用語について説明します。

製品間の関係 - コンパイラー、ランタイム、Debug Tool

IBM Enterprise PL/I for z/OS は、z Systems プラットフォームに対応する、IBM の戦略的 PL/I コンパイラーです。言語環境プログラムは、言語ランタイム環境を提供します。 Debug Tool は、大きな改良が加えられたデバッグ機能を提供します。

IBM Enterprise PL/I for z/OS
Enterprise PL/I は、OS PL/I、PL/I for MVS & VM、および VisualAge PL/I の機能から構成されており、また Unicode のサポート、XML の構文解析機能、C と Java のインターオペラビリティーの向上、統合 CICS プリプロセッサー、および統合 SQL プリプロセッサーなどの追加機能を有します。

言語環境プログラム
言語環境プログラムにより、COBOL、PL/I、C、および FORTRAN 用の単一言語ランタイム環境が提供されます。既存アプリケーションのサポートに加えて、言語環境プログラムは共通の条件処理、向上した言語間通信 (ILC)、再使用可能なライブラリー、および言語間アプリケーションのより効率的なアプリケーション開発も提供します。アプリケーション開発は、共通の規約、共通のランタイム機能、および共用される一連の呼び出し可能サービスを使用することで単純化されます。言語環境プログラムは、Enterprise PL/I プログラムを実行するために必要となります。

Debug Tool
Debug Tool を使用すると、Enterprise PL/I プログラムと、他の言語環境プログラム準拠の言語プログラム (COBOL や C/C++ など) をデバッグできます。 Debug Tool は、以前の PL/I デバッグ・ツールと比べて、大きな改良が加えられたデバッグ機能を提供します。

PL/I コンパイラーの一般情報

Enterprise PL/I アプリケーションをコンパイルする際には、言語環境プログラムにアクセスする必要があります。アプリケーションをコンパイルする際に既存の JCL を使用する場合は、STEPLIB ステートメントまたは JOBLIB ステートメントに SCEERUN (言語環境プログラムのランタイム・ライブラリー) を必ず組み込むか、またはその SCEERUN が LNKLST に組み込まれていることを確認してください。

PL/I アプリケーションのコンパイルには、 IBMZC カタログ式プロシージャーを使用できます。

コンパイル・ステップには、次のステートメントが含まれている必要があります。

© Copyright IBM Corp. 1999, 2016

```
//PLI      EXEC PGM=IBMZPLI,REGION=4000K
//STEPLIB  DD  DSN=&LNGPRFX;.SIBMZCMP,DISP=SHR
//         DD  DSN=&LIBPRFX;.SCEERUN,DISP=SHR
```

コンパイル時の SCEERUN の使用法を理解するには、 Enterprise PL/I に付属するカタログ式プロシージャーに関する情報が役立ちます。 詳しくは、「*Enterprise PL/I for z/OS* プログラミング・ガイド」で『PL/I のカタログ式プロシージャーの使用』を参照してください。

言語環境プログラムを使用して Enterprise PL/I アプリケーションのリンク・エディットを行う際に、既存の JCL を使用する場合は、SYSLIB ステートメントに SCEELKED (言語環境プログラムのリンク時ライブラリー) を必ず組み込んでください。

他のプログラムに対する言語環境プログラムのランタイム・サポート

Enterprise PL/I は、ランタイム環境として言語環境プログラムを使用します。言語環境プログラムにより、COBOL、PL/I、C、および FORTRAN 用の単一言語ランタイム環境が提供されます。

言語環境プログラムは、次の言語コンパイラー用の共通ランタイム環境です。

C/370
C/C++
COBOL for MVS & VM
COBOL for OS/390 & VM
Fortran
PL/I for MVS & VM
Enterprise PL/I

言語環境プログラムには、ランタイム・オプションおよび呼び出し可能サービスの共通セットが用意されています。また、各 ILC 呼び出しの言語固有の初期化および終了をなくすことによって、高水準言語 (HLL) とアセンブラーの間の言語間通信 (ILC) も改善しています。 言語環境プログラムは、既存のアプリケーションのために互換性サポートを提供しますが、このサポートにはいくつかの制約があります。

新しいコンパイラーおよびランタイムの利点

新しい IBM Enterprise PL/I for z/OS コンパイラーは、数多くの新しい機能および利点を備えています。このトピックには、役立つ拡張機能の概要が記載されています。

- フェッチの向上:
  - フェッチされたルーチンは、他のルーチンをフェッチできる。
  - フェッチされたルーチンは、MAIN と同じ入出力を実行できる。
  - フェッチされたルーチンは、それ自身の CONTROLLED を持つことができる。
- DECIMAL および PICTURE の精度は 31 桁。
- さまざまな制限の緩和:
  - 内部名および外部名に最大 100 文字まで使用可能。

  - FILE および CONTROLLED 変数の数にコンパイラーの制限はない。
  - PROCEDURE ごとに 4095 個までのパラメーターが許可される。
- 390 の新しい命令をサポート (AHI、ALCR など)。
- 書き込み可能な再入可能静的属性および DLL のサポート。
- 容易になった C/C++ との互換性およびインターオペラビリティー。
- 整数のサポートの向上:
  - 符号付き FIXED BIN の最大精度は 63 桁
  - UNSIGNED 属性をサポート (最大精度 64 桁)
  - 符号付き FIXED BIN(7) は 1 バイトにマップ (UNSIGNED FIXED BIN(8) と同様)
- 以下の項目を含む、いくつもの新しい強力な言語機能:
  - PACKAGE (第 2 の ENTRY に対する ANSI の代替機能)
  - DO FOREVER (`DO WHILE( 1 = 1 );` に対する優れた代替)
  - " (二重引用符) によるストリングの区切り
  - 定数を読みやすくするためのアンダースコアーの使用 (例えば "0011_0101"b)
  - 複合代入 (例えば *x += 1;*)
  - RESIGNAL (より強力な例外処理用)
- 以下の項目を含む、いくつもの新しい強力な属性:
  - ABNORMAL (C の volatile と同様)
  - NONASSIGNABLE (C の const と同様)
  - BYVALUE
  - LIMITED ENTRY (C の関数ポインター用)
  - ORDINAL (強く型付けされた enum 用)
  - RESERVED (C に類似した static 用)
  - UNION
  - UNSIGNED
  - VALUE (名前付き定数用)
  - VARYINGZ (C スタイルのヌル終了ストリング用)
- 以下の項目を含む、100 を超える新しい組み込み関数:
  - HEX および HEXIMAGE (デバッグ用)
  - PROCNAME および SOURCELINE (トレース用)
  - PLIMOVE、PLIFILL、および COMPARE (memcpy、memset、および memcmp と同様)
  - IAND、IOR、IEOR、および NOT (ビット単位の整数演算用)
  - COPY (ANSI で定義された REPEAT の改良版)
- 以下の項目を含む、フル z/OS UNIX システム・サービス・サポート:
  - HFS 内のソース、オブジェクト、およびリスト・ファイル
  - HFS ファイルに対する入出力
- 向上したマクロ機能:
  - デック・ファイルはソースの大/小文字を保持する。
  - マクロ変数は配列にもできる。
  - いくつもの組み込み関数がさらにサポートされる。
  - ステートメント ANSWER、ITERATE、LEAVE、SELECT (オープン・コードおよびマクロで)、および REPLACE がサポートされる。
  - キーワード WHILE、UNTIL、および LOOP が %DO ステートメントでサポートされる。

- マルチスレッド化のサポート
- UTF-16 Unicode のサポート
- IEEE 浮動小数点のサポート
- SAX スタイルの XML 構文解析
- XML 生成
- 統合された CICS プリプロセッサー
- 統合された SQL プリプロセッサー

詳細については、「*PL/I* 言語解説書」および「*Enterprise PL/I for z/OS* プログラミング・ガイド」を参照してください。

新しいコンパイラーおよびランタイムの主な変更点

Enterprise PL/I を使用した場合、例えばコンパイラー・オプションの廃止または変更、デフォルトのコンパイラー・オプションの相違、および新旧のロード・モジュールの結合に関する制限などのいくつかの点で、既存の PL/I アプリケーションは影響を受けます。

次に示す考慮事項のリストは、一部のお客様にとって重要であった事項の中から、代表的な項目だけを抽出したものです。 個々の特定のお客様に重要な情報が示されているとは限りません。詳細については、本書で後述します。

- Enterprise PL/I は、現在サポートされている言語環境プログラムのリリースのみをサポートします。
- Enterprise PL/I は VM をサポートしていません。
- Enterprise PL/I はマルチタスキングをサポートしていません (ただし、マルチスレッド化はサポートしています)。
- 誤ったコードや無効なコード (例えば、未初期化変数を使用するコード) は、従来どおりに実行されない場合があります。 この問題は重要でないように見えるかもしれませんが、マイグレーションを行ったほとんどのお客様にとって重大な問題でした。
- コンパイラーの動作の互換性を最大限に高めたり、コンパイラーの最高のパフォーマンスを確保したりするために、デフォルト以外のオプションをいくつか指定する必要が生じる場合があります。
- パフォーマンスを最適化するために、プログラムのチューニングが必要になる場合があります。 特に、ランタイム・オプション RPTSTG(ON) を使用すると、パフォーマンスをチューニングしている間は役立ちますが、このオプションを残したままにしておくと、実動プログラムで大きな負担になります。
- すべての PL/I ソースを再コンパイルすることをお勧めします。 再コンパイルをしない場合には、旧版 PL/I オブジェクトと混合させる Enterprise PL/I コードをコンパイルする際に、コンパイラー・オプションを慎重に選択する必要があります。 また、FILE、CONTROLLED 変数、および条件の使用方法に従って、ソースを複数の区画に分割する必要があります。 詳しくは、 149 ページの『オブジェクト・モジュールおよびロード・モジュールに関する考慮事項』を参照してください。

一般的な移行作業

ほとんどの場合、お客様のニーズに応じて、一般的な移行作業を 1 つ以上行う必要があります。

- 戦略を計画する。
- 言語環境プログラム・ランタイム・ライブラリーに移行する。
- ソースを Enterprise PL/I で再コンパイルする。
- Enterprise PL/I プログラムを既存のアプリケーションに追加する。

戦略の計画

言語環境プログラム・ランタイム・ライブラリーに移行したり、Enterprise PL/I を使用してソース・プログラムを再コンパイルする前に、移行の戦略を立てます。 周到な戦略を立てることは、新しいコンパイラーおよびランタイムにスムーズに移行するために役立ちます。

移行の戦略としては、まず最初に言語環境プログラムに移行し、次に Enterprise PL/I を使用して、既存アプリケーションを必要に応じて順次再コンパイルしていくことになります。 本書では、新しいランタイムに移行するための戦略と、PL/I ソースを再コンパイルするための戦略を個別に提供します。

現在は言語環境プログラムを使用していないが、移行を計画する方法についての情報が必要な場合は、 15 ページの『第 3 章 言語環境プログラムへの移行の計画』を参照してください。

すでに言語環境プログラムに移行しており、新しいコンパイラーへの移行に関する情報が必要な場合は、 25 ページの『第 4 章 新しいコンパイラーへの移行の計画』を参照してください。

言語環境プログラム・ランタイムへの移行

既存のロード・モジュールを、言語環境プログラムで実行して、言語環境プログラム以前のライブラリーを使用した場合と同じ結果を得ることができます。

互換性に関する重要な情報については、 33 ページの『第 5 章 言語環境プログラムの下での既存アプリケーションの実行』を参照してください。

旧版 PL/I ランタイムで実行されているアプリケーションの移行に関する情報については、 37 ページの『第 6 章 マイグレーション前の考慮事項』を参照してください。

ほとんどの場合、言語環境プログラムを使用して既存のアプリケーションをリンク・エディットするか、または Enterprise PL/I を使用してプログラムを再コンパイルする必要があります。 言語環境プログラムを使用してリンク・エディットする必要のあるプログラムを判別するには、 59 ページの『第 8 章 リンク・エディットに関する考慮事項』を参照してください。

Enterprise PL/I によるソースの再コンパイル

新しい Enterprise PL/I コンパイラーでは、数多くの強力な新機能を利用できます。この新しいコンパイラーと、以前の PL/I コンパイラーとの間には、いくつかの相違点もあります。

Enterprise PL/I コンパイラーと、以前の PL/I コンパイラーの相違点については、 69 ページの『第 10 章 新しいコンパイラーの制限について』を参照してください。

新しいコンパイラー・オプションについての詳細は、 81 ページの『第 11 章 新しいコンパイラーのオプションについて』を参照してください。

変更を加えた後に Enterprise PL/I を使用して再コンパイルする必要のあるプログラムを判別するには、 113 ページの『第 13 章 作業コードを変更する必要がある場合について』および 127 ページの『第 14 章 作業コードを変更する必要がある可能性のある場合について』 を参照してください。

Enterprise PL/I プログラムを既存アプリケーションに追加

新しい Enterprise PL/I プログラムを作成 (または Enterprise PL/I を使用して既存のプログラムを再コンパイル) し、それらを既存アプリケーションとともに言語環境プログラムで実行できます。

Enterprise PL/I プログラムを既存アプリケーションに追加する場合は、新旧の PL/I モジュールを混在させる場合の制限に注意する必要があります。詳しくは、 149 ページの『第 18 章 既存の PL/I アプリケーションへの Enterprise PL/I プログラムの追加』を参照してください。

第 2 部 移行の戦略

© Copyright IBM Corp. 1999, 2016

第 3 章 言語環境プログラムへの移行の計画

この章では、ランタイム環境を言語環境プログラムに移行するための一般的戦略について説明します。

以下の作業が必要です。作業は、おおむね次の順序で行ってください。

- 言語環境プログラム・ランタイム・ライブラリーへの移行を準備する。
- アプリケーションのインベントリーを作成する。
- 段階的に言語環境プログラムに移行する方法を決定する。
- リグレッション・テスト・プロシージャーをセットアップする。
- 実動使用に切り替える。

すでに言語環境プログラムに移行している場合は、この章をお読みになる必要はありません。 25 ページの『第 4 章 新しいコンパイラーへの移行の計画』に進んでください。

重要: Enterprise PL/I プログラムは、z/OS バージョン 1 リリース 7 以降の言語環境プログラムのエレメントを使用した場合にのみ実行できます。

言語環境プログラム・ランタイム・ライブラリーへの移行を準備

言語環境プログラムへの移行を準備するにあたって、言語環境プログラムをインストールし、ストレージ要件を評価し、プログラマーを言語環境プログラムについて習熟させる必要があります。

これらのタスクは同時に行うことができます。

言語環境プログラムのインストール

このトピックには、言語環境プログラムをインストールするときに参照できる資料がリストされています。

言語環境プログラムを z/OS にインストール

z/OS (言語環境プログラム・エレメントも含む) をインストールするには、「*z/OS Program Directory*」または「*ServerPac: Installing Your Order*」を参照してください。

言語環境プログラムを OS/390 にインストール

OS/390 (言語環境プログラム・エレメントも含む) をインストールするには、「*OS/390 Program Directory*」または「*ServerPac: Installing Your Order*」を参照してください。

© Copyright IBM Corp. 1999, 2016

重要: 言語環境プログラム・ランタイムの結果が言語環境プログラム以前の結果と互換性を保つようにするために、デフォルトのランタイム・オプションの変更が必要になる場合があります。詳しくは、 37 ページの『ランタイム・オプションにおける相違点』を参照してください。

ストレージ要件の評価

言語環境プログラムのストレージ要件は、言語環境プログラム以前の PL/I ライブラリーの場合よりも大きくなります。

DASD ストレージ要件

移行時には、言語環境プログラム・ランタイム・ライブラリー用の DASD ストレージと、言語環境プログラム以前のランタイム・ライブラリー用の DASD ストレージが必要です。言語環境プログラムへの移行が完了したら、以前の PL/I ランタイム・ライブラリー用に予約したストレージは解放することができます。

言語環境プログラムが必要とする DASD ストレージの容量を判断するには、次の資料を参照してください。

- z/OS: *z/OS Program Directory*
- OS/390: *OS/390 Program Directory*

仮想記憶域の要件

言語環境プログラムを使用して PL/I プログラムを実行するための仮想記憶域の要件は、OS PL/I ランタイムの場合よりも増大します。CICS アプリケーションと非 CICS アプリケーションのどちらの場合も、増加する量は次のような多くの要因に左右されます。

- 言語環境プログラムのランタイム・ストレージ・オプション (STACK、LIBSTACK、HEAP、ANYHEAP、および BELOWHEAP) に使用された値
- 言語環境プログラムのランタイム・オプション ALL31 の値
- LPA (リンク・パック域) または ELPA (拡張リンク・パック域) に置かれているランタイム・ルーチン

注: 言語環境プログラム RPTSTG(ON) ランタイム・オプションによって生成された情報は、調整フェーズでストレージ・オプションを調整するために役立ちます。詳しくは、「*z/OS* 言語環境プログラム プログラミング・リファレンス」を参照してください。 このオプションは、パフォーマンスを大きく低下させるため、PL/I アプリケーションを実動状態にする前に、RPTSTG(OFF) に再設定してください。

言語環境プログラムについてのプログラマーの教育

言語環境プログラムに移行する前に、アプリケーション・プログラマーが言語環境プログラムの機能や、言語環境プログラム以前のランタイムと言語環境プログラム・ランタイムの相違点について十分理解するようにしてください。

言語環境プログラムに習熟していれば、プログラマーは言語環境プログラムへの移行に対して、さらに準備を整えることができます。例えば、アプリケーションのインベントリーの作成を支援できます。

IBM を通じて利用できる Enterprise PL/I および言語環境プログラムの教育については、1-800-IBM-TEACH にお問い合わせください。また、言語環境プログラムの資料、ユーザー・グループ (SHARE など)、および言語環境プログラムのホーム・ページ (http://www.ibm.com/s390/le) から直接情報を入手することもできます。

アプリケーションのインベントリーの作成

言語環境プログラム・ランタイムへの移行を計画するときは、言語環境プログラムで実行する予定のアプリケーションをすべて網羅したインベントリーを作成する必要があります。

このインベントリーには、次のものを含める必要があります。

- ベンダー製のツール、パッケージ、および製品
- PL/I アプリケーション

Edge Portfolio Analyzer は、既存のロード・モジュールについてインベントリーを作成するために役立ちます。

関連情報:

207 ページの『EDGE Portfolio Analyzer』

ベンダー製のツール、パッケージ、および製品

ランタイムの言語環境プログラムへの移行を開始する前に、ベンダー製のツール、パッケージ、および製品が言語環境プログラムで動作するように設計されているかどうかを確認する必要があります。

以下の項目について確認してください。

- すべてのパッケージが言語環境プログラムで動作すること (特にそれらのソース・コードが手元にない場合)。
- ソースが手元にあるパッケージのソース・コードが、Enterprise PL/I コンパイラーでコンパイルできること。
- コード生成プログラムが、Enterprise PL/I コンパイラーでコンパイルできるソース・コードを生成すること。
- 独自の ESPIE または ESTAE を発行する開発ツールおよびデバッガーが、言語環境プログラムと調和して動作すること。

言語環境プログラムで使用可能なベンダー製品のリストの入手方法については、207 ページの『ベンダー製品』を参照してください。

PL/I アプリケーション

PL/I アプリケーションのインベントリーを作成する際には、言語環境プログラムへの移行に影響を与えるプログラム属性についての情報を収集する必要があります。この情報には、テストの方法と対象、および言語環境プログラムでパフォーマンスに影響を与える要因などが含まれます。

インベントリーについて、以下の項目を判別しておく必要があります。

言語環境プログラムへのアプリケーションの移行に関して:

- どのプログラムが OS PL/I でコンパイルされているか、およびどのプログラムが PL/I for MVS & VM でコンパイルされているか。
- どのプログラムが PL/I 共用ライブラリーにリンクされているか。
- 使用されているランタイム・オプション (および、それらがどのように指定されているか)。
- どの PL/I プログラムがアセンブラー・プログラムを呼び出すか、あるいはアセンブラー・プログラムから呼び出されるか。
- どの PL/I プログラムがマルチタスキングを行うか。
- どの PL/I プログラムが言語間通信 (COBOL、C、または FORTRAN) を行うか。
- どの PL/I プログラムが CICS、IMS、DB2、またはその他のサブシステムで使用されているか。
- 予想される異常終了の頻度およびタイプ。

レグレッション・テストに関して:

- 必要であり使用可能なテスト・ケース

パフォーマンス測定に関して:

- 使用するストレージの容量。
- 再使用可能モジュール/共通モジュールの実行頻度。
- プログラムの実行時間 (CPU 時間と経過時間の両方)。

既存の PL/I ロード・モジュール

言語環境プログラムへの移行を計画する上で、ご使用のライブラリーにどのバージョンの PL/I ロード・モジュールが含まれているかを把握しておくことは重要です。 Edge Portfolio Analyzer は、既存のロード・モジュールについてインベントリーを作成するために役立ちます。

AMBLIST ユーティリティーもまた、ご使用のロード・モジュールについての情報を提供してくれるツールです。 AMBLIST は IBM により提供され、通常 *SYS1.LINKLIST* にあります。 LISTIDR 制御ステートメントを使用することで、選択した CSECT 識別レコード (IDR) のリストを得ることができます。 IDR のフィールドの 1 つには、CSECT のコンパイルに使用された変換プログラムの名前、もしくは PL/I の場合はコンパイラーの名前が含まれています。次の例は、AMBLIST によるサンプル出力です。

```
-------------------------------------------------------------------------
CSECT       TRANSLATOR       VR.MD               YR/DY
MYPLI       5655-H31         32.00               2003/171
MYPLI2      5655-B22         22.01               2001/073
D1          566896201        02.01               1972/271
UNRES       566896201        02.01               1992/034
-------------------------------------------------------------------------
```

TRANSLATOR 列中のテキストを使用して、どの PL/I コンパイラーがモジュールを作成したか判別することができます。各種 PL/I コンパイラーの Translator フィールド値については、 19 ページの表 3 を参照してください。

表 3. *PL/I* コンパイラーの *IDR* 値

PL/I コンパイラー・バージョン	変換プログラム識別レコード
OS PL/I V1 リリース 5.1	5734-PL1
OS PL/I V2.3	5668-910
PL/I for MVS & VM	5688-235
VisualAge PL/I for OS/390 V2R2	5655-B22
Enterprise PL/I for z/OS バージョン 3	5655-H31
Enterprise PL/I for z/OS バージョン 4	5655-W67

段階的に言語環境プログラムに移行する方法の決定

言語環境プログラムを実動モードで使用する準備ができたら、以下の項目について決定してください。

- 複数言語の変換処理の方法
- アプリケーションがライブラリーにアクセスする方法

複数言語の変換処理

ILC を使用する PL/I アプリケーションが存在する場合には、関係する各言語を変換した後、それらのアプリケーションを言語環境プログラム・ランタイムに移行します。

例えば、PL/I-COBOL アプリケーションを言語環境プログラムに移行するには、その前に PL/I のみ、および COBOL のみのアプリケーションを言語環境プログラムに移行しておきます。

注**:** 1 つの言語について、2 つの異なるライブラリーを LNKLST/LPALST にインストールすることはできません。 例えば、言語環境プログラムを PL/I コンポーネントとともに LNKLST/LPALST にインストールする場合、OS PL/I ライブラリーまたは PL/I for MVS & VM ライブラリーは LNKLST/LPALST にインストールしないでください。

デフォルトでは、言語環境プログラムを LNKLST にインストールすると、すべての PL/I アプリケーションは言語環境プログラムで実行されます。

アプリケーションがライブラリーにアクセスする方法の決定

言語環境プログラムを実動状態に移行するための一般的方法としては、言語環境プログラムを LNKLST/LPALST に追加する方法と、STEPLIB アプローチを使用する方法の 2 種類があります。

LNKLST/LPALST

言語環境プログラムを LNKLST/LPALST に追加すると、言語環境プログラムはすべてのアプリケーションから使用可能になります。

言語環境プログラムを LNKLST/LPALST に追加する前に、すべてのアプリケーションが言語環境プログラムで正しく機能するようにするためには、言語環境プログラムを LNKLST/LPALST に一時的にインストールするか、または STEPLIB を使用します。

実行時に、複数の PL/I ランタイム・ライブラリーをアプリケーションで使用可能にしてはいけません。例えば LNKLST には、言語環境プログラム用の SCEERUN などの PL/I ランタイム・ライブラリーが 1 つだけ含まれるようにする必要があります。複数のライブラリーが存在すると、ライブラリーが見つからないというエラーや、連結内に使用されないロード・ライブラリーが含まれるという事態が発生します。 言語環境プログラムを LNKLST/LPALST に追加する場合は、他のすべての PL/I ランタイム・ライブラリーを除去してください。

一時的に LNKLST/LPALST にインストールまたは STEPLIB を使用するときの提案

- 言語環境プログラムを、最初にテスト用または開発用マシンの LNKLST/LPALST にインストールする。
- MVS システム・コマンド SETPROG を使用して LNKLST または LPA を一時的に変更する (システムの IPL を実行する必要はありません)。 SETPROG コマンドの使用については、「*z/OS MVS* システム・コマンド」 (SA88-8593) または「*OS/390 MVS* システム・コマンド」 (GC88-6592) を参照してください。
- 週末に IPL を実行し、言語環境プログラムを LNKLST/LPALST にインストールする。週末の間に、アプリケーションが言語環境プログラムで実行できることを検証する。

注: z/OS および OS/390 の多くのエレメントが言語環境プログラム・ランタイム・ライブラリーに依存していますが、z/OS と OS/390 はどちらも、言語環境プログラムが LNKLST にインストールされていることを必要としません。 (ただし、言語環境プログラムは z/OS および OS/390 と同じゾーンにインストールする必要があります。) 言語環境プログラムを LNKLST に入れない場合、言語環境プログラムを必要とする、個別の z/OS PROC または OS/390 PROC で、言語環境プログラムにアクセスする STEPLIB を使用する必要があります。どのエレメントが言語環境プログラムを必要とするかについては、「*z/OS Program Directory for z/OS バージョン 1 リリース 1*」または「*OS/390 Program Directory for OS/390 バージョン 2 リリース 10*」を参照してください。

STEPLIB

STEPLIB アプローチを使用することにより、言語環境プログラムを段階的に導入できます。 このようにして、一度に 1 つの領域 (CICS または IMS)、バッチ (アプリケーションのグループ)、またはユーザー (TSO) を段階的に取り入れます。

STEPLIB を使用すると、JCL を変更することになりますが、段階的に移行するほうが、すべてのアプリケーションを一度に移行するよりも簡単になる可能性があります。 さらに、STEPLIB を使用するときは、プログラムの実行速度は LNKLST/LPALST を通してランタイム・ライブラリーにアクセスするときよりも遅くなり、より多くの仮想記憶域が使用されることに注意してください。

注: 複数のプロセッサーがチャネル間接続で相互にリンクされている場合は、システム全体を 1 つのプロセッサーとして扱い、サブシステムごとに移行する必要があります。 CEEDUMP のデフォルトの割り振りがユーザーのショップのニーズに合わない場合は、初期セットアップ時に JCL を変更して STEPLIB に 言語環境プログラム・ランタイムを指定することに加えて、CEEDUMP DD も指定する必要が生じる場合があります。 (CEEDUMP は、言語環境プログラムがダンプ出力を書き込む DD 名です。)

STEPLIB と IMS プログラムに関する問題:

STEPLIB を IMS/DC オンライン上で使用して言語環境プログラム・ランタイムにアクセスすると、プリロードした言語環境プログラム・ライブラリー・ルーチンは、読み取り専用ストレージにはロードされません。

アプリケーションにエラーがあり、アプリケーション以外のストレージを上書きした場合、プリロードしたランタイム・ルーチンは破壊され、使用すると異常終了を発生させる可能性があります。 再入可能としてマークされたこれらのプリロード済みルーチンは、LPA または LNKLST/LPALST からロードされた場合を除き、リフレッシュ時にリフレッシュされません。 そのため、異常終了は繰り返し発生します。

注: これは、MVS (OS/390)、IMS、および STEPLIB に関して 20 年前から存在する問題ですが、言語環境プログラムに段階的に移行するために提案した STEPLIB アプローチに関連して、ここで説明しました。

この問題を回避するには、次のいずれかの方法を使用します。
- 言語環境プログラムを LNKLST/LPALST にインストールする。
- ランタイム・ルーチンをプリロードしない。 (これによりパフォーマンスが低下します。)

影響を最小限にとどめる方法
- 言語環境プログラムの証明を可能な限り短くする。 (証明を早く受けるほど、LNKLST/LPALST に早くインストールできます。)
- 複数の異なるアプリケーションが同じ領域で異常終了するかどうかを監視する。これにより、リカバリー手順を行う必要があるかどうかがわかります。

リカバリーの方法

複数の異なるアプリケーションが同じ領域で異常終了していたら、下に示す IMS コマンドを使用し、その領域を停止して再始動します。
1. 次のコマンドを発行して、領域番号を判別します。
   ```
   '/DISPLAY ACTIVE'
   ```
2. 次のコマンドを発行して、領域を停止します。
   ```
   '/STOP REGION region#'
   ```
3. 次のコマンドを発行して、領域を再始動します。
   ```
   '/START REGION region-name'
   ```

STEPLIB の例:

次の例は、STEPLIB 方式を用いて段階的に言語環境プログラムにしていく方法を示しています。中央の開発センター (すべてのコンパイルおよびリンクが 1 つの場所で行われる) と個別の実動場所がある組織を想定してください。

これは非常に無難なアプローチですが、実動アプリケーションが絶対に停止しないことを要求する多くのお客様によって使用されています。

1. 中央の開発センターで言語環境プログラムと Enterprise PL/I を証明する。
   a. 現在のランタイム上でキャプチャーしたデータを使用してテストを実行し、すべての結果を保存する。

b. 言語環境プログラムを STEPLIB 環境にインストールする。 これにより、変更されていないジョブは現在のランタイムで実行されますが、一部のユーザーは、STEPLIB JCL を使用して言語環境プログラム・ランタイム・ ライブラリーにアクセスすることにより、言語環境プログラム・ランタイムを使用できます。

c. STEPLIB 環境を使用し、言語環境プログラム・ランタイム上でキャプチャーしたデータを使用してテストを実行し、その結果を現在のランタイムの場合と比較する。 証明サイクルの全体を通じて並列テストを実行し、アプリケーションを言語環境プログラムで実行した場合に、現在のランタイムで実行した場合と同じ結果が出るようにします。

d. テスト・アプリケーションを Enterprise PL/I でコンパイルする。 STEPLIB を使用して言語環境プログラム・ランタイム・ライブラリーにアクセスし、証明テストを再実行します。

2. 言語環境プログラムを中央の開発センターのシステムにインストールし、テストを実行する。
   a. 既存アプリケーションの移行されていないバージョンについて、STEPLIB を使用して現在のランタイムにアクセスして、並列テストを実行する。
   b. すべての新規アプリケーションを、実稼働実行用にリリースする前に、言語環境プログラム・ランタイム環境で実行する。
3. 言語環境プログラム・ランタイムをバックアウトする必要が生じた場合に備えて、現在のランタイムのインストール手順を保管することによって、バックアウト戦略を準備しておく。
4. 言語環境プログラム・ランタイムを 1 カ所の実動場所にインストールする。
   a. 既存アプリケーションの移行されていないバージョンについて、STEPLIB 環境で現在のランタイムを使用して、並列テストを継続する。
   b. この実動場所で言語環境プログラム・ランタイムを 1 カ月間実行する。
5. 言語環境プログラム・ランタイムをすべての実動場所にインストールする。
   a. 既存アプリケーションの移行されていないバージョンについて、STEPLIB 環境で現在のランタイムを使用して、並列テストを継続する。
   b. すべての実動場所で言語環境プログラム・ランタイムを 1 カ月間実行する。
   c. 1 カ月経過したら、現在のランタイム・ライブラリーの内容をすべて削除する。

可能な範囲で、最も大きな作業単位の移行を試行してください。オンライン領域、アプリケーション、あるいは実行単位の全体を一度に移行することで、1 つのアプリケーションや実行単位に含まれるプログラム間の相互作用を確実にテストできます。

レグレッション・テスト手順のセットアップ

コーディングの技法には非常に多くの組み合わせがあるため、アプリケーションが言語環境プログラムでも実行でき、期待される結果が得られるかどうかを判別する唯一の方法は、レグレッション・テストの手順をセットアップすることです。

アプリケーションをテスト環境に移し、言語環境プログラムで実行した場合にも、期待される結果が得られるようにします。

リグレッション・テストは、以下の項目を識別するために役立ちます。

- 127 ページの『第 14 章 作業コードを変更する必要がある可能性のある場合について』に示した、必要とされるソース・コードの変更。
- 現在のランタイムと言語環境プログラム・ランタイムのストレージ使用量の相違。
- 現在のランタイムと言語環境プログラム・ランタイムの CPU 時間の相違。

テストの際には、既存のアプリケーションを現在のランタイムと言語環境プログラム・ランタイムの両方で同時に実行し、結果が同じであることを検証します。 既存アプリケーションのパフォーマンスを測定し、言語環境プログラムのパフォーマンスと比較します。

プログラムが正常に実行されたら、プログラムを個別にテストし、また実行単位内の別のプログラムとともにテストします。 プログラムを各種のデータでテストすることで、プログラムのすべての処理機能を実行できるため、実行の予期しない相違が生じないようにすることができます。

プログラムの出力を分析し、結果が正しくない場合は、デバッグ・ツールまたは言語環境プログラムのダンプ出力を使用して、エラーを見つけ出し、それらのエラーを修正します。必要なその他の変更も行い、その後に再実行します。必要な場合には、デバッグを継続します。

注**:** ほとんどのアプリケーションは、言語環境プログラムで実行した場合でも、既存のランタイムでの場合と同じ結果を生じますが、コーディングのスタイル、リソースの使用状況、パフォーマンス、異常終了時の動作、あるいは言語環境プログラムでの IBM の規則をより厳格に順守していることが原因で、異なる結果が得られる場合があります。既存のコードが異なる動作をする状態については、 127 ページの『第 14 章 作業コードを変更する必要がある可能性のある場合について』を参照してください。

パフォーマンス測定の実行

アプリケーションがテスト環境内の言語環境プログラム下で実行された後、特に時間や応答が重視されるアプリケーションについて、パフォーマンスを測定してください。

言語環境プログラムと現在のランタイム環境のランタイム・パフォーマンスを比較し、パフォーマンスを向上させる必要のあるアプリケーションが見つかったら、プログラムを調整し、パフォーマンスを向上させるために利用できる方法を調べます。例えば、言語環境プログラムのランタイム・オプションを使用して、ストレージの値を変更できます。

実動使用への切り換え

テストの結果、アプリケーション全体 (IMS 領域または TSO で複数のアプリケーションを実行している場合は、アプリケーションのグループ) で期待される結果が生じることが確認できたら、ユニット全体を実動使用に移行します。 ただし、予期しないエラーに対する準備が必要です。

以下のようにして、リカバリーを準備できます。

- z/OS および OS/390 の場合は、最後のプロダクティビティー・チェックポイントから、旧版バージョンを代替として実行する。
- DB2、CICS、および IMS の場合は、最後のコミット地点に戻り、マイグレーションされていない PL/I プログラムを使用して、その場所から処理を継続する。(DB2 の場合は、SQL ROLLBACK WORK ステートメントを使用してください。)
- バッチ・アプリケーションの場合は、ショップにあるバックアップおよびリストア機能を使用してリカバリーする。

既存のアプリケーションを言語環境プログラム・ランタイム環境で実動使用に移行したら、アプリケーションを短期間モニターして、正常に動作していることを確認します。 そうすることで、以前のランタイムと同じだけの信頼性のある実行が可能になります。

第 4 章 新しいコンパイラーへの移行の計画

この章では、ソース・プログラムを Enterprise PL/I に移行するための一般的戦略について説明します。

以下の作業が必要です。作業は、おおむね次の順序で行ってください。

1. ソースを新しいコンパイラーに移行する準備をする。
2. アプリケーションのインベントリーを作成する。
3. アプリケーション・プログラムを更新する。

従来の PL/I コンパイラーに対するサービス・サポートはいずれ終了されるため、最終的にすべての PL/I ソース・プログラムを新しいコンパイラーに移行する必要があります。これは直ちに必要となるわけではありませんが、将来の時点で旧版コンパイラーとサポート対象フィックスは利用できなくなります。 その時点で、「非常に短期間で」マイグレーションする必要に迫られることになりますが、それが大変不都合な時期に行わなければならない場合もあります。

ソース・プログラムを新しいコンパイラーに移行する前に、アプリケーションを言語環境プログラムに移行しておく必要があります。

新しいコンパイラーへのソースの移行の準備

ソースを新しい Enterprise PL/I コンパイラーに移行する準備をする際には、Enterprise PL/I をインストールし、ストレージ要件を評価し、プログラマーを新しいコンパイラーについて習熟させる必要があります。

これらのタスクは同時に行うことができます。

Enterprise PL/I のインストール

Enterprise PL/I をインストールしていない場合、このコンパイラーをインストールしてください。

z/OS または OS/390 の場合は、使用する製品の「プログラム・ディレクトリー」を参照してください。

ストレージ要件の評価

Enterprise PL/I オブジェクト・プログラムは 31 ビットまたは 64 ビット・アドレッシング・モードで実行でき、16 MB ラインより上に常駐できるため、16 MB ラインより下のストレージを解放できます。解放されたストレージは、16 MB ラインより下に常駐する必要のあるプログラムやデータのために使用できます。

コンパイラーのマイグレーションの際には、Enterprise PL/I コンパイラー用だけでなく、現在の PL/I コンパイラー用としても DASD ストレージが必要になります。 コンパイラーのマイグレーションが完了し、OS PL/I、PL/I for MVS &

© Copyright IBM Corp. 1999, 2016

VM、または VisualAge PL/I プログラムをすべて Enterprise PL/I に移行したら、現在の PL/I コンパイラー用に予約されているストレージを解放できます。

同じソース・コードから生成したロード・モジュールでも、Enterprise PL/I でコンパイルした場合には、OS PL/I または PL/I for MVS & VM でコンパイルした場合よりもサイズが大きくなる可能性があります。

新しいコンパイラーの機能についてのプログラマーの教育

移行作業の早い段階で、アプリケーション・プログラマーが Enterprise PL/I の機能や、Enterprise PL/I、言語環境プログラム、デバッグ・ツールあるいはショップで使用されているその他のアプリケーション生産性向上ツールの間の関係および相互依存性について、十分理解するようにしてください。

さらにプログラマーは、言語環境プログラム・ランタイム・オプション、条件処理、および呼び出し可能サービスについても十分に習熟する必要があります。

ご使用の環境に応じた正しいコンパイラー・オプションを選択することは、非常に重要な作業です。 選択するオプションは、パフォーマンスを最適にするか、以前の PL/I バージョンとの互換性を優先するかによって、大きく異なります。 コンパイラー・オプションの選択について詳しくは、 81 ページの『第 11 章 新しいコンパイラーのオプションについて』を参照してください。

IBM を通じて利用できる Enterprise PL/I および言語環境プログラムの教育については、1-800-IBM-TEACH にお問い合わせください。また、言語環境プログラムの資料や SHARE などのテクニカル・コンファレンス、あるいは IBM Technical Interchange から直接情報を入手することもできます。

プログラマーが Enterprise PL/I の機能に習熟したら、プログラムのインベントリーの作成を支援することができます。

関連情報:

『アプリケーションのインベントリーの作成』

アプリケーションのインベントリーの作成

PL/I ソース・プログラムの Enterprise PL/I への移行を計画する際には、Enterprise PL/I でコンパイルする予定のプログラムを含んだ、アプリケーションの包括的なインベントリーを作成する必要があります。

アプリケーションのインベントリーを作成することで、必要な作業を詳細に把握することができます。 以下のアプリケーションについて、インベントリーを作成する必要があります。

- ベンダー製のツール、パッケージ、および製品
- PL/I アプリケーション

Edge Portfolio Analyzer は、既存のロード・モジュールについてインベントリーを作成するために役立ちます。

関連情報:

207 ページの『EDGE Portfolio Analyzer』

ベンダー製ツール、パッケージ、および製品のインベントリーの作成

ソースの移行を開始する前に、ベンダー製のツール、パッケージ、および製品が、Enterprise PL/I で動作するように設計されているかどうかを確認する必要があります。

- PL/I コード生成プログラムが、Enterprise PL/I でコンパイルできる PL/I プログラムを生成するかどうかを確認します。
- PL/I パッケージを Enterprise PL/I でコンパイルできるかどうかを確認します。

PL/I アプリケーションのインベントリーの作成

PL/I アプリケーションの評価にあたって、さまざまな要因を考慮する必要があります。

PL/I アプリケーション内の各プログラムについて、少なくとも以下の情報をインベントリーに含めます。

OS PL/I、**PL/I for MVS & VM**、および **VisualAge PL/I** の場合**:**

- 担当のプログラマー
- 使用したコンパイラー
- 使用したコンパイラー・オプション、特に CMPAT
- 使用したプリコンパイラー・オプション
- PL/I モジュール
- PL/I プログラムで使用されている INCLUDE ライブラリー・メンバー
- 呼び出し先、または FETCH 先のサブプログラム
- 呼び出し側、または FETCH する側のプログラム
- 実行の頻度
- 必要であり使用可能なテスト・ケース

従来の PL/I モジュールを Enterprise PL/I モジュールと混用する予定の場合は、以下の情報もインベントリーに含めます。

- CONTROLLED 変数の使用
- FILE 変数および定数の使用

上記の情報に基づく PL/I ソース・プログラムの実行単位への区分化については、152 ページの『PL/I ソース・プログラムの実行単位への区分化』を参照してください。

アプリケーションの優先順位付け

インベントリーが完成したら、そのインベントリーを使用して、移行作業に優先順位を付けます。

1. 完成したインベントリーの各項目に複雑さの等級を割り当て、個々のプログラムやアプリケーションについて、総合的な複雑さの等級を決定する。

2. 各プログラムまたはアプリケーションについて、移行の優先順位を決定する。

移行の優先順位の決定

インベントリーに含まれる各プログラムの複雑さの等級を決定したら、その情報に基づいて、新しい Enterprise PL/I コンパイラーに移行するプログラムや、それらのプログラムを移行する順序を決定することができます。

移行の優先順位を決定する際には、次の点を考慮する必要があります。

- 16MB ラインより下で使用できるストレージの限度一杯であるアプリケーションは、Enterprise PL/I に移行する第一の候補となる。 z/OS または OS/390 アーキテクチャーでは、仮想記憶域制約解放を利用できます。
- 言語環境プログラムでは実行できないプログラムは、変換する必要がある。

移行する必要のある各プログラムの優先順位と、それらのプログラムを移行するために必要な作業を決定したら、アプリケーションおよびプログラムを変換する順序を決定します。

移行カテゴリーと非移行カテゴリーのセットアップ

確立した移行の優先順位を使用し、プログラムの重要性と実行の頻度を考慮することにより、ほとんどのプログラムを、Enterprise PL/I に移行する順序でリストすることができます。

以下のような、移行をまったく行いたくないプログラムがある場合もあります。

- ソース・コードが手元になく、再コンパイルを必要とせず、言語環境プログラム環境で正常に実行されるプログラム。
- 組織にとって重要度が低く、言語環境プログラム環境で正常に実行され、移行の際に多大な労力を必要とするプログラム。
- 実動から段階的に除去していくプログラム。

ただし、既存のモジュールを、新しい Enterprise PL/I コンパイラーに移行されたプログラムと混合して実行する際には、制限がある場合があることに注意してください。 149 ページの『第 18 章 既存の PL/I アプリケーションへの Enterprise PL/I プログラムの追加』を参照してください。

アプリケーション・プログラムの更新

以下に示すアプリケーション・プログラミング作業は、ソースを移行する際には必ず行う必要があります。プログラム更新の規模を決定する必要があります。

例えば、通常の保守とともにプログラムの更新を行うことも可能ですし、プログラムを機能上のグループに分割し、ソースをグループ単位で更新することもできます。 「ビッグ・バン」プロセスにより、すべてのプログラム更新を同時に行った例もあります。いずれの進行方法に決定した場合でも、これらの作業は、おおむね次に示す順序で行う必要があります。

移行したモジュールに問題が生じた場合に備えて、既存のソースをバックアップ (比較対象のベンチマークやリカバリー用のバージョン) として保存しておきます。

1. ジョブおよびモジュールの資料を更新する。

すべての更新を正しく文書化しておくことは、きわめて重要です。 PL/I 自体は、インテリジェントに自己文書化を行います。 ただし、指定したコンパイラー・オプションや、それらを指定した理由については、ログに記録しておく必要があります。 また、システムに関する特別な考慮事項についても、すべて記録してください。 これは反復プロセスであり、移行のためのプログラミング作業全体を通じて行う必要があります。

2. 使用可能なソース・コードを更新する。

   ソース・コードを手動で、または独自に開発したツールを使用して更新します。どのようなときにソース・コードを変更する必要があり、どのようなときに変更が必須でないかについては、 113 ページの『第 13 章 作業コードを変更する必要がある場合について』および 127 ページの『第 14 章 作業コードを変更する必要がある可能性のある場合について』を参照してください。

3. コンパイルし、リンク・エディットして、実行する。

   ソースの更新が終わったプログラムは、新しく作成された Enterprise PL/I プログラムと同様に処理することができます。 (言語環境プログラム・ランタイムをインストールしておく必要があります。) コンパイル処理中に新規のメッセージが表示され、その詳細を調べる場合は、 97 ページの『第 12 章 新しいコンパイラーのメッセージについて』を参照してください。

4. デバッグする。

   プログラムの出力を分析し、結果が正しくない場合は、デバッグ・ツールまたは言語環境プログラムのダンプ出力を使用して、エラーを見つけ出します。

5. 変換されたプログラムをテストする。

   ソースを Enterprise PL/I に移行したら、レグレッション・テストの手順をセットアップします。 リグレッション・テストは、以下の項目を識別するために役立ちます。

   - 変更する必要のあるコード
   - ファイル属性のミスマッチ
   - ストレージ初期設定の問題
   - パフォーマンスの相違点
   - AMODE 問題

   レグレッション・テストの手順を確立し、プログラムが正常に実行されたら、さまざまなデータを使用して、次のようにプログラムをテストします。

   - ローカル: 各プログラムを個別に。
   - グローバル: 1 つの実行単位内にある複数のプログラムを、相互に対話する状態で。

   この方法により、プログラムのすべての処理機能を実行できるため、実行の予期しない相違が生じないようにすることができます。

   レグレッション・テストは非常に重要です。従来の PL/I コンパイラーから Enterprise PL/I への移行は、類似した言語ではあるものの、別の言語への移行として捉える必要があるため、テストもそのように計画してください。

6. 必要があれば繰り返す。

   必要なその他の修正も行い、その後で再コンパイル、再リンク、および再実行します。必要な場合には、デバッグを継続します。

7. 実動モードに切り換える。

   テストの結果、アプリケーション全体で期待される結果が生じることが確認できたら、ユニット全体を実動モードに移行します。 (ここでは、実動システムがすでに言語環境プログラム・ランタイムを使用していることを想定しています。そうではない場合、 20 ページの『STEPLIB』を使用して、言語環境プログラム・ランタイムにアクセスしてください。)

   予期しないエラーが発生した場合は、次のようにリカバリーを行います。

   - z/OS または OS/390 の場合は、最後のプロダクティビティー・チェックポイントから、旧版バージョンを代替として実行する。
   - DB2 および IMS の場合は、最後のコミット地点に戻り、マイグレーションされていない PL/I プログラムを使用して、その場所から処理を継続する。(DB2 の場合は、SQL ROLLBACK WORK ステートメントを使用してください。)
   - 非 CICS アプリケーションの場合は、ショップにあるバックアップおよびリストア機能を使用してリカバリーする。

8. 実動モードで実行する。

   実動モードに切り換えた後、アプリケーションを短期間モニターして、期待される結果が得られていることを確認します。これが終わったら、ソースの移行作業は完了です。

第 3 部 既存アプリケーションの言語環境プログラムへの移行

重要: この章は、OS PL/I からのマイグレーションを予定しており、かつ現時点では言語環境プログラムを使用していないユーザーを対象としています。 現在 PL/I for MVS & VM、VisualAge PL/I、または Enterprise PL/I を使用している場合には、直接 67 ページの『第 4 部 新しいコンパイラーへの移行』に進んでください。

© Copyright IBM Corp. 1999, 2016

第 5 章 言語環境プログラムの下での既存アプリケーションの実行

アプリケーションの特性によっては、アプリケーションに変更を加え、いくつかの言語環境プログラム・カスタマイズ作業を行って、現行アプリケーションが言語環境プログラムのもとで稼働するようにする必要があります。

- 既存のアプリケーションを起動します。
- 既存のアプリケーションをリンク・エディットします。

ランタイムを OS PL/I または PL/I for MVS & VM から移行している場合、互換性を確保するには、その他の要因も当てはまります。詳しくは、37 ページの『第 6 章 マイグレーション前の考慮事項』を参照してください。

既存のアプリケーションの起動

言語環境プログラムにアクセスするには、アプリケーションを起動するために使用している手順を変更する必要があります。非 CICS アプリケーションに必要な手順は、CICS アプリケーションの場合の手順とは異なります。

注: プログラム名には AFH、CEE、EDC、IBM、IGZ、ILB、または FOR で始まる名前は使用できません。 これらのプレフィックスは、言語環境プログラム・ライブラリー・ルーチンのモジュール名として予約されています。

CICS 以外のアプリケーションに関する考慮事項

このセクションには、CICS 以外のアプリケーションに必要な起動プロシージャーにおける変更に関する詳細があります。

言語環境プログラムを使用してプログラムを作成および実行する方法についての詳細は、「*z/OS* 言語環境プログラム プログラミング・ガイド」を参照してください。

正しいライブラリーの指定

言語環境プログラムの下で実行する場合、既存のアプリケーションを起動するには、現行ライブラリーを言語環境プログラム SCEERUN ライブラリーに置き換える必要があります。

代替の DDNAME の指定 (オプション)

言語環境プログラムを使用している場合、言語環境プログラムの出力の宛先を指示するには、MSGFILE ランタイム・オプションの DD 名を、目的の DD 名に変更します。

34 ページの表 4 に、言語環境プログラムの出力のデフォルトの DD 名をリストします。

© Copyright IBM Corp. 1999, 2016

表 4. 新規 *DD* 名の仕様

出力	デフォルトの **DD** 名
メッセージ	SYSOUT
ランタイム・オプション報告書 (RPTOPTS)	SYSOUT
ストレージ報告書 (RPTSTG)	SYSOUT
ダンプ	CEEDUMP

上の表の DD 名は、すべて動的に割り振られます。

言語環境プログラムで使用されているデフォルトがショップのニーズに合わない場合を除いて、言語環境プログラムのメッセージ、報告書、またはダンプの DD 名を定義するために、JCL、CLIST、または Rexx EXEC を変更する必要はありません。言語環境プログラムのデフォルト宛先は次のとおりです。

- z/OS および OS/390 の場合 : SYSOUT=*

CICS アプリケーションに関する考慮事項

CICS 上で言語環境プログラムを実行するには、いくつかの必須ステップを実行する必要があります。

CICS 上で実行する PL/I アプリケーションを言語環境プログラムの下で起動する方法、および言語環境プログラム・ランタイム・ライブラリー SCEERUN を指定する方法についての詳細は、以下の資料を参照してください。

- z/OS の場合は「*z/OS Language Environment* カスタマイズ」
- OS/390 の場合は「*OS/390* 言語環境プログラム カスタマイズ」

CICS 上で言語環境プログラムを使用した場合の出力の相違点

CICS の下では、言語環境プログラムの出力は CESE という名前の一時データ・キューに入れられます。 ファイルに書き込まれた各レコードには、端末 ID、トランザクション ID、日付、および時刻を含んだヘッダーがあります。 一時データ・キュー (CESE) は、次のタイプの言語環境プログラム出力を受け取ります。

- メッセージ
- ランタイム・オプション報告書 (RPTOPTS)
- ストレージ報告書 (RPTSTG)
- ダンプ
- PL/I ストリーム出力

既存のアプリケーションのリンク・エディット

言語環境プログラムを使用してリンク・エディットする必要のある、またはリンク・エディットすることが有益となる既存アプリケーションが判別できたら、次に正しいライブラリー名を指定する必要があります。

言語環境プログラムのリンク・エディット・ライブラリーは、非 CICS アプリケーションの場合と、CICS アプリケーションの場合とで共通です。

z/OS および **OS/390**

SYSLIB 連結に言語環境プログラム SCEELKED を組み込みます。

注**:** NCAL リンケージ・エディター・オプションを使用してリンク・エディットする場合は、SCEELKED のすべての必須ランタイム・ルーチンをロード・モジュールに組み込んでください。 そのようにしない場合は、予測不能なエラーが発生します (一般的にはプログラム・チェック)。

SCEELKED ライブラリーには、IBM の命名規則に従っていないため、ユーザーのサブプログラム名と競合する可能性のある名前がいくつか存在します。 例えば、DUMP という名前の、静的に呼び出されるサブルーチンがあり、リンク・エディット時に SCEELKED が連結内で専用サブルーチン・ライブラリーよりも先に置かれていた場合、DUMP に対する参照は SCEELKED で解決されます。 この例では、FORTRAN ルーチンの AFHUDUMS がリンク・エディットによって組み込まれるため、誤った結果が得られたり、機能が失われたり、あるいは結果的にパフォーマンスが低下したりする可能性があります。 (もう一つの共通名は ABORT で、これは C ランタイム・ライブラリーのルーチンである EDC4$05C のエントリー・ポイントです。)

これらの問題を回避するには、次の 2 つの方法があります。

- SCEELKED データ・セット内の名前を、ユーザーの専用サブルーチンの名前と比較してチェックする。 重複が見つかった場合は、SCEELKED データ・セット内の名前と同じにならないように、専用サブルーチンの名前を変更します。
- 別の方法として、専用サブルーチン・ライブラリーを SYSLIB 連結内で SCEELKED の前に置く。 ただし、この方法を行うと、アプリケーションに Fortran または C/C++ のプログラムが含まれていた場合に、言語環境プログラムで使用できる機能が使用できなくなる可能性があります。 ユーザーのサブルーチンの名前を変更することで言語環境プログラム・サブルーチンとの競合を回避する方法のほうが、専用サブルーチン・ライブラリーを SCEELKED の前に置く方法よりも優れています。

言語環境プログラムを使用してリンク・エディットする必要のあるアプリケーションを判別するには、59 ページの『第 8 章 リンク・エディットに関する考慮事項』を参照してください。

第 6 章 マイグレーション前の考慮事項

この章では、OS PL/I ランタイムと言語環境プログラムの機能の相違点について説明します。アプリケーションを言語環境プログラムにマイグレーションする前にこれらの相違点について考慮してください。

言語環境プログラムは、現在では z/OS および OS/390 の一部であるため、PL/I for MVS & VM、VisualAge PL/I、あるいは Enterprise PL/I などの言語環境プログラム対応 PL/I コンパイラーをインストールする前に、言語環境プログラムへのアプリケーションのマイグレーションを開始することができます。

ランタイム・オプションにおける相違点

言語環境プログラムのランタイム・オプションは、PL/I のランタイム・オプションを置き換えます。 PL/I のランタイム・オプションのほとんどに、同じ機能を提供する同等な言語環境プログラムのランタイム・オプションがあります。ここでは、ランタイム・オプションの使用法の相違点について説明します。

次の相違点を考慮して、アプリケーションを修正する必要があります。

削除されたランタイム・オプション

- OS PL/I の COUNT オプションは無視されます。
- OS PL/I の FLOW オプションは無視されます。
- OS PL/I の HEAP オプションは常に有効です。 つまり、BASED 変数と CONTROLLED 変数にストレージを割り振る際に、ストレージは常に HEAP ストレージから割り振られます。 ストレージは、PL/I 初期ストレージ域 (ISA) からは割り振られません。 HEAP(0) はサポートされず、使用しても無視されます。

置き換えられたランタイム・オプション

- 言語環境プログラムの NATLANG オプションは、OS PL/I の LANGUAGE オプションを置き換えます。
- 言語環境プログラムの RPTSTG オプションは、OS PL/I の REPORT オプションを置き換えます。
- 言語環境プログラムの TRAP オプションは、OS PL/I の SPIE と STAE の両オプションを置き換えます。 次の表は、OS PL/I の SPIE オプションおよび STAE オプションが、言語環境プログラムの TRAP オプションにマッピングされる方法を示しています。

© Copyright IBM Corp. 1999, 2016

表 5. *SPIE* および *STAE* オプションの *TRAP* オプションとの対応

OS PL/I	言語環境プログラム	処置
SPIE|NOSPIE	TRAP(ON|OFF)	SPIE|NOSPIE が入力で指定されていれば、TRAP は、SPIE の場合はオプション TRAP(ON)、NOSPIE の場合はオプション TRAP(OFF) に従って設定されます。
STAE|NOSTAE	TRAP(ON|OFF)	STAE|NOSTAE が入力で指定されていれば、TRAP は、STAE の場合はオプション TRAP(ON)、NOSTAE の場合はオプション TRAP(OFF) に従って設定されます。
SPIE STAE または SPIE NOSTAE または STAE NOSPIE NOSPIE NOSTAE	TRAP(ON) TRAP(OFF)	SPIE|NOSPIE と STAE|NOSTAE の両方が共に入力で指定されている場合、TRAP は両方のオプションに従って次のように設定されます。両方のオプションが否定の場合は TRAP(OFF)、それ以外の場合は TRAP(ON)。アプリケーションが正常に実行されるためには、TRAP(ON) が有効である必要があります。

注**:** 独自の条件管理を実行するアプリケーションは、言語環境プログラムの条件管理と矛盾する場合がよくあります。 言語環境プログラムの条件処理について詳しくは「*z/OS* 言語環境プログラム プログラミング・ガイド」を参照してください。

- 言語環境プログラムの STACK オプションは、OS PL/I の ISASIZE と ISAINC の両オプションを置き換えます。 ISASIZE および ISAINC を含むソース・コードを変更および再コンパイルする必要はありません。 また、PLIXOPT ストリングを含むオブジェクト・モジュールまたはロード・モジュール、あるいはその両方は、言語環境プログラムの環境では従来のままの ISASIZE と ISAINC を使用して実行されます。

  STACK(,,ANY) は、エディットされたストリーム入出力を含まない、言語環境プログラムと再リンクされた OS PL/I アプリケーションに対して使用できます。

  STACK(,,ANY) を使用するには、アプリケーションを AMODE(31) で実行する必要があります。

  CICS 環境では、ALL31(ON) および STACK(,,ANY) はデフォルトです。ただし、言語環境プログラムに再リンクされていない OS PL/I アプリケーションには STACK(,,BELOW) が必須であるため、インストール時にデフォルトを STACK(,,BELOW) に変更するか、または再リンクされていないすべての OS PL/I アプリケーションに対して明示的に STACK(,,BELOW) を指定する必要があります。

新しいランタイム・オプション

- 言語環境プログラムの ABTERMENC オプションは、異常終了時にアプリケーションが受け取る戻りコードや異常終了コードのタイプを制御します。 ABTERMENC(RETCODE) を指定すると、アプリケーションはランタイム戻りコードを受け取ることができ、OS PL/I と同等の動作になります。

- 言語環境プログラムの ERRCOUNT オプションは、ランタイムに処理される条件の数を制限します。 ERRCOUNT(0) を指定するとこの数は無制限になり、OS PL/I と同等の動作になります。
- 言語環境プログラムの DEPTHCONDLMT オプションは、条件をネストできる度合いを制限します。 互換性を維持するには、深さに制限がないことを示す DEPTHCONDLMT(0) を指定します。
- 言語環境プログラムの XUFLOW オプションは、アンダーフローが発生したときに UNDERFLOW 条件をオンにするかどうかを決定します。 XUFLOW(AUTO) を指定すると、UNDERFLOW 条件のオン/オフに関して PL/I のセマンティクスが維持されます。
- 言語環境プログラムの ALL31 オプションは、ライブラリー・ルーチン間での AMODE の切り替えを制御します。 すべてのアプリケーションが AMODE(31) である場合は、ALL31(0N) を設定する必要があります。

ランタイム・オプションを MVS GO ステップで渡す場合は、メイン・プロシージャーのパラメーター・ストリングと区別するため、ランタイム・オプションのストリングの末尾をスラッシュ (/) にする必要があります。スラッシュを省略すると、そのストリングはメイン・プロシージャーのパラメーターとして渡されます。

次に示すランタイム・オプションは、OS PL/I との互換性を提供するために必要になります。
- ABTERMENC(RETCODE)
- ERRCOUNT(0)
- DEPTHCONDLMT(0)
- STORAGE(,,CLEAR)
- TRAP(ON)
- XUFLOW(AUTO | ON)

STORAGE オプションの CLEAR サブオプションを使用する前に、APAR PK02614 用の適切な PTF をインストールしておく必要があることに注意してください。 またこのオプションを使用すると、Enterprise PL/I コード全体が効率的ではなくなる可能性があります。変数の初期化については、本書の後続部分を参照してください。

ランタイム・オプションについて詳しくは、「*z/OS* 言語環境プログラム プログラミング・リファレンス」を参照してください。

OS PL/I アプリケーションの場合、PLIXOPT ストリングに指定したオプションは、アプリケーション固有のオプションとして処理されます。 言語環境プログラムの CEEUOPT を指定すると、CEEUOPT は無視されます。

メイン・ロード・モジュールに ILC が含まれている場合、PLIXOPT ストリングは無視されます。この場合、アプリケーション固有オプションに対しては、CEEUOPT を指定する必要があります。

条件処理における相違点

タイミングの相違点

PL/I の条件処理セマンティクスは、言語環境プログラムでもサポートされます。ただし、ERROR ON ユニットに関する ERROR 条件のランタイム・メッセージを出すタイミングが異なります。

以下のように異なっています。

- ERROR 条件のランタイム・メッセージが発行されるのは、ERROR ON ユニットが設定されていない場合、または ERROR ON ユニットがブロックから抜ける GOTO によって条件から回復していない場合だけです。 ERROR ON ユニットから抜ける GOTO を使用することで、PL/I ERROR 条件のメッセージを回避できます。

メッセージの発行と ERROR 条件の発生を暗黙処置とする PL/I 条件の場合、メッセージを発行するタイミングは変更されません。

表 6 に、ERROR ON ユニットに関連して、OS PL/I の環境で ERROR 条件のランタイム・メッセージが発行されるタイミングを示します。

表 *6. OS PL/I* バージョン *2* リリース *3*の *ERROR ON* ユニットと *ERROR* 条件のメッセージ

条件	**ON** ユニットなし	**ERROR ON** ユニット	**ERROR ON** ユニット **GOTO**
ERROR 条件が発生[1]	メッセージ	メッセージは ON ユニットの前	メッセージは ON ユニットの前
ZERODIVIDE 条件が発生[2]	メッセージ	メッセージは ON ユニットの前	メッセージは ON ユニットの前

注:

1 負数の平方根やデータ例外などの処理。

2 ZERODIVIDE ON ユニットは使用しません。このため、暗黙処置が行われます。メッセージが表示され、ERROR 条件が発生します。

表 7 に、ERROR ON ユニットに関連して、言語環境プログラムの環境で ERROR 条件のランタイム・メッセージが発行されるタイミングを示します。

表 *7.* 言語環境プログラムの *ERROR ON* ユニットと *ERROR* 条件のメッセージ

条件	**ON** ユニットなし	**ERROR ON** ユニット **GOTO** なし	**ERROR ON** ユニット **GOTO**
ERROR 条件が発生[1]	メッセージ	メッセージは ON ユニットの後	メッセージなし
ZERODIVIDE 条件が発生[2]	メッセージ	メッセージは ON ユニットの前	メッセージは ON ユニットの前

表 7. 言語環境プログラムの *ERROR ON* ユニットと *ERROR* 条件のメッセージ (続き)

条件	**ON** ユニットなし	**ERROR ON** ユニット **GOTO** なし	**ERROR ON** ユニット **GOTO**

注:

1 負数の平方根やデータ例外などの処理。

2 ZERODIVIDE ON ユニットは使用しません。このため、暗黙処置が行われます。メッセージが表示され、ERROR 条件が発生します。

ERROR ON ユニットが制御を受け取るまで、ON ERROR SNAP によって生成される SNAP トレースバック・メッセージが継続して出されます。 SNAP トレースバック・メッセージは、通常の ERROR メッセージとは異なります。

一部の高精度固定小数点除算の実行における相違点

Enterprise PL/I V5R1 の時点で、一部の高精度固定小数点除算は、10 進浮動小数点 (DFP) 機能を使用して行われます。このため、いくつかの ZERODIVIDE 例外が INVALIDOP として報告される可能性があります。

未処理条件の相違点

OS PL/I アプリケーションが、OS PL/I アセンブラー・ユーザー出口 IBMBXITA または異常終了出口 IBMBEER を使用して、OS PL/I ランタイム環境で未処理条件の異常終了を強制していた場合、言語環境プログラムの環境では、ABTERMENC(ABEND) オプションまたは CEEBXITA ユーザー出口を使用して異常終了を強制できます。

言語環境プログラムのもとで異常終了を強制するには、以下の方法のいずれかを使用します。

- 言語環境プログラム ABTERMENC(ABEND) オプションを使用してアプリケーションを実行する。 ランタイム・オプションを使用して、ユーザー独自の異常終了コードを指定することはできません。
- 言語環境プログラムのアセンブラー・ユーザー出口 CEEBXITA を使用して、ユーザー独自の異常終了コードによる異常終了を強制する。

IBMBXITA および IBMBEER の相違点

言語環境プログラムは、OS PL/I IBMBXITA と IBMBEER のサポートを制限付きで提供しています。

詳しくは、 186 ページの『アセンブラー・ユーザー出口の使用に関する考慮事項』を参照してください。

ABEND U4039 の相違点

重大度 2 以上の UNHANDLED 条件が発生すると、異常終了 U4039 が生成されます。また、SYSUDUMP または SYSABEND DD 名が存在する場合は、オプションでシステム・ダンプも生成されます。

ABTERMENC(RETCODE) が有効な場合でも、アプリケーションは異常終了コードを出して終了します。 U4039 異常終了が発生しないようにしたい場合は、言語環境プログラムの提供する機能を使用して異常終了を抑止できます。

U4039 異常終了を抑止または変更する方法については、「*z/OS Language Environment Installation and Customization under OS/390*」の『Abnormal Termination Exit』または「*z/OS* 言語環境プログラム カスタマイズ」を参照してください。

重大度の相違点

一部の PL/I 条件の重大度は、言語環境プログラム環境では異なっています。

重大度については、「*PL/I* 言語解説書」を参照してください。

PLICALLA と PLICALLB のサポートの相違点

言語環境プログラム環境では、このセクションで説明するインターフェースは使用しないようお勧めします。

PLICALLA に関する考慮事項

言語環境プログラムは、PLICALLA エントリー・ポイントを使用する OS PL/I アプリケーションをサポートします。 プログラムを言語環境プログラム環境で再リンクすることもできます。

言語環境プログラム環境でアプリケーションを再リンクする方法の詳細は、 203 ページの『OS PL/I ルーチン置換ツール』を参照してください。

PLICALLA は、FETCH/CALL された PL/I メイン・ロード・モジュールの 1 次エントリー・ポイントとして使用できます。ただし、呼び出しルーチンは、サブルーチンに渡されるユーザー引数だけを渡す必要があります。ランタイム・オプションを渡した場合、これらはユーザー引数として処理されます。

新しく開発する PL/I アプリケーションで、メイン・プロシージャーがサブルーチンと同様にユーザー引数を受け取るようにするには、次のいずれかを行います。

- 次のようにして、IMS から直接制御を受け取る。
  - ロード・モジュールの 1 次エントリー・ポイントとして CEESTART または PLISTART を使用する。
  - SYSTEM(IMS) コンパイル時オプションを指定する。
- FETCH または CALL ステートメントを使用して、次のようにアセンブラー・プログラムまたはプロシージャーから制御を受け取る。
  - ロード・モジュールの 1 次エントリー・ポイントとして CEESTART または PLISTART を使用する。
  - NOEXECOPS オプションおよび SYSTEM(MVS) コンパイル時オプションを指定する。
  - アセンブラー・コードがパラメーターを渡すために使用した機構に必要で適切な、BYADDR オプションまたは BYVALUE オプションを指定する。

次の環境では、言語環境プログラムによる PLICALLA のサポートは利用できません。

CICS 環境
事前初期設定済みの環境
ネストされた別プログラム環境 (PL/I FETCHable メインを除く)

PLICALLB に関する考慮事項

言語環境プログラムは、PLICALLB エントリー・ポイントを使用する PL/I アプリケーションをサポートします。

次の表に、OS PL/I と言語環境プログラムとの間での PLICALLB パラメーターのマッピングを示します。

表 *8. PLICALLB* 引数リスト・サポートにおける相違点

OS PL/I	言語環境プログラム
非マルチタスキング・プログラム、またはマルチタスキング・プログラム内の大タスクの、ISA ストレージの長さのアドレス	STACK(*init_size*) にマップされます。
ISA ストレージのアドレス	初期 STACK セグメントとして使用されます。
各サブタスクに対する ISA ストレージの長さのアドレス	NONIPTSTACK(*init_size*) にマップされます。
並行サブタスクの最大数のアドレス	PLITASKCOUNT(*max_thread*) にマップされます。
次のランタイム・オプションを指定できる、オプション・ワードのアドレス：	次のようにサポートされます。
REPORT	REPORT は RPTSTG にマップされます。
SPIE|STAE	SPIE|STAE は TRAP にマップされます。
COUNT	COUNT は無視されます。
FLOW	FLOW は無視されます。
HEAP サブオプション	HEAP(,,KEEP|FREE)|(,,ANY|BELOW)
TASKHEAP サブオプション	THREADHEAP(,,KEEP|FREE)|(,,ANY|BELOW)
非マルチタスキング・プログラム、またはマルチタスキング・プログラム内の大タスクの、HEAP ストレージの長さのアドレス	HEAP(*init_size*) にマップされます。
HEAP ストレージのアドレス	初期 HEAP セグメントとして使用されます。
非マルチタスキング・プログラム、またはマルチタスキング・プログラム内の大タスクの、HEAP 増分のアドレス	HEAP(,*incr_size*) にマップされます。
サブタスクの HEAP のアドレス	THREADHEAP(,*increment*) にマップされます。
非マルチタスキング・プログラム、またはマルチタスキング・プログラム内の大タスクの、ISA 増分のアドレス	STACK(,*incr_size*) にマップされます。
各サブタスクの ISA 増分のアドレス (非タスキング・アプリケーションの場合はオプション)	NONIPTSTACK(,*incr_size*) にマップされます。

PLICALLB エントリー・ポイントを介して上記の引数リストを渡す際に、リストの引数はアドレスを指しているか、または 0 でなければなりません。引数の高位ビットが ON であることは、引数リストの最後を表します。 R1 には引数リストのアドレスが格納されている必要があります。

言語環境プログラムの環境では、PLICALLB エントリー・ポイントを介して渡されるランタイム・オプションは、アプリケーションの起動時に指定されるオプションとして処理され、CEEUOPT オプションまたは PLIXOPT オプションより優先順位が高くなります。アセンブラー・ユーザー出口を使用して、PLICALLB の呼び出しによって渡されるランタイム・オプションを変更することはできません。

要約すると、渡されるランタイム・オプションは、言語環境プログラムのオプションの指定方法において、次の優先順位を持ちます (高いほうから順)。
1. インストール時に定義され、オーバーライド不可能属性を持つオプション
2. PLICALLB エントリー・ポイントを介して指定されたオプション
3. PLIXOPT ストリングまたは CEEUOPT 内で指定されたオプション
4. インストール時に定義された、オプションのデフォルト

PL/I メインルーチンに渡されるユーザー引数の優先順位は、次のとおりです (最高から最低の順)。
1. アセンブラー・ユーザー出口の CXIT_PARM または AUE_PARM からの出力
2. PLICALLB エントリーを介して渡されるユーザー引数

注: アセンブラー・ユーザー出口の CXIT_PARM または AUE_PARM への入力は、PLICALLB パラメーター・リストの最初の引数、つまりユーザー引数アドレスのベクトルのアドレスです。

言語環境プログラムは、16M ラインより上のストレージの使用を推奨しています。OS PL/I との互換性のために、言語環境プログラムはユーザー提供の ISA ストレージと HEAP ストレージを、STACK と HEAP にマップします。 ただし、このマッピングを使用する場合でも、言語環境プログラムは GETMAIN をいくつか実行する必要があります。ユーザー提供の ISA/HEAP ストレージは通常 16M ラインより下にあるので、言語環境プログラムの環境では、16M ラインより下のストレージがすぐに消費される可能性があります。言語環境プログラムによるストレージの管理方法は、「*z/OS* 言語環境プログラム プログラミング・ガイド」に記載されています。

言語環境プログラムによるストレージの管理方法は、OS PL/I とは異なります。OS PL/I がサポートする ISA と HEAP よりも多くのカテゴリーにストレージを分割します。その結果、ユーザーが提供した OS PL/I の ISA または HEAP ストレージを言語環境プログラムの STACK または HEAP ストレージにマップするためには、ランタイムに GETMAIN を必要とします。 さらに、言語環境プログラムでは、ユーザーの提供した ISA または HEAP ストレージの長さが 8 バイトの倍数となっており、アドレスがダブルワードの境界に来ていることを確認するために診断が行われます。

また、言語環境プログラムは、ユーザーが提供した ISA または HEAP ストレージの位置が、STACK または HEAP ランタイム・オプションでの位置指定に一致するようにします。ユーザー提供の HEAP ストレージは、以下の条件すべてがあてはまる場合には無視されます。

1. ユーザーが提供したヒープ・ストレージが 16M ラインより上にある。
2. HEAP オプションの ANYWHERE サブオプションが有効である。
3. メインプログラムが AMODE(24) にある。

言語環境プログラムは、16M ラインより下のストレージを、HEAP オプションで指定された init_sz24 および incr_sz24 サブオプションを使用して割り振ります。

次の環境では、言語環境プログラムによる PLICALLB のサポートは利用できません。

CICS
IMS
事前初期設定済みの環境
ネストされた別プログラム環境

PLICALLB エントリー・ポイントでサポートされるメインルーチンは 1 つのみです。

事前初期設定サポートにおける相違点

Enterprise PL/I は、旧版事前初期設定の方式をサポートしないため、アプリケーションの再設計を考慮しなければならない場合があります。

言語環境プログラムの事前初期設定サービスは、再設計したアプリケーションで Enterprise PL/I と共に使用する必要があります。ただし、事前初期設定済みのプログラムを、それらのプログラムが再設計されるまでの間に限って言語環境プログラム環境で実行する場合のために、このセクションではマイグレーションの前に考慮しておく必要のある相違点について説明します。

PL/I の事前初期設定済みプログラム・インターフェースは、サポートされますが、以下の点が変更されています。

- PL/I 事前初期設定済みプログラム・インターフェースは、REINITIALIZE 要求修飾コードをサポートしません。 この機能を使用すると、診断の結果 4093-136 異常終了コードが出力されます。
- CALL 要求で指定されたルーチンが、アセンブラー・ドライバーと静的にリンクされておらず、かつ ILC を含んでいる場合には、ILC 環境が、同じ ILC を INIT 要求で指定されたルーチンに組み込むことによって初期化されるようにする必要があります。
- TERM 要求は、OS PL/I ランタイムとは異なり、1000 の戻りコードを戻しません。
- OS PL/I によって定義されたサービス・ベクトルの戻りコードおよび理由コードの一部が変更されています。 言語環境プログラムの事前初期設定サービスで定義されたサービス・ベクトルの戻りコードおよび理由コードは、「*z/OS* 言語環境プログラム プログラミング・リファレンス」の説明に従って使用する必要があります。

言語環境プログラムの事前初期設定サービスは、同じ TCB で複数の事前初期設定環境をサポートします。 同じ TCB での複数の事前初期設定環境は、OS PL/I で

はサポートされません。 このサービスの動作方法を理解するには、「*z/OS* 言語環境プログラム プログラミング・ガイド」の『事前初期設定サービスの使用』を参照してください。

PLISRTx のサポートの相違点

PLISRTx 呼び出しが含まれている OS PL/I アプリケーションは、Language Environment for OS/390 & VM リリース 1.4 以降でサポートされます。ただし、ランタイムとして言語環境プログラムのリリース 1.3 を使用している場合は、アプリケーションを再リンクする必要があります。

使用しているリリースにかかわらず、言語環境プログラムを使用してロード・モジュールを再リンクすることをお勧めします。それには、次の理由があります。

- 再リンクすることにより、ライブラリー・ルーチンは言語環境プログラム提供の DFSORT インターフェースにアクセスできるようになるため、より統合された言語およびソート環境が実現できる。
- 再リンクすることにより、ライブラリー・ルーチンは 24 ビット DFSORT パラメーター・リストを拡張 31 ビット DFSORT パラメーター・リストに置き換えることができる。

OS PL/I PLISRTx アプリケーションを再リンクするには、次のいずれかの方法を使用します。

- 205 ページの『OS PL/I オブジェクト・モジュール再リンク・ツール - APAR PN69803』を使用してオブジェクト・モジュールを再リンクする。
- 203 ページの『OS PL/I ルーチン置換ツール』を使用してライブラリー・ルーチンを置き換える。
- オブジェクト・モジュールを直接言語環境プログラムに再リンクする。

マルチタスキング・サポートにおける相違点

Enterprise PL/I では、マルチタスキングはサポートされていません。

マルチスレッド化を使用するようにアプリケーションを変更するか、または PL/I for MVS & VM コンパイラーを使用する必要があります。 Enterprise PL/I マルチスレッド化コードには、POSIX(ON) ランタイム・オプションを使用しなければならないことにも注意してください。

OS PL/I 共用ライブラリーのサポートの相違点

Enterprise PL/I では、旧版 OS PL/I 共用ライブラリーはサポートされません。

DATE/TIME 組み込み関数における相違点

DATETIME および TIME 組み込み関数は、すべての環境で ms (ミリ秒) の値を戻すようになりました。

これらの組み込み関数の構文および説明は、「*PL/I* 言語解説書」に記載されています。

ユーザー戻りコードの相違点

言語環境プログラムは、PLIRETC、PLIRETV、および OPTIONS(RETCODE) に対して、FIXED BIN(31) 4 バイト・ユーザー戻りコード値をサポートします。 このサポートにより、最大値が 999 という制約事項が除去されます。 4 バイト・ユーザー戻りコード値を利用するには、OS PL/I アプリケーションを 言語環境プログラムに再リンクする必要があります。

次の表は、PL/I ユーザー戻りコードがどのようにサポートされるかを示しています。

表 *9.* 言語環境プログラムでの戻りコードの振る舞い

関数	**OS PL/I** ロード・モジュール	言語環境プログラムにリンクされた **OS PL/I** オブジェクト・モジュール	**Enterprise PL/I** ロード・モジュール
PLIRETC 組み込み関数	999 までに制限された 2 バイト値	999 までに制限されない 4 バイト値	999 までに制限されない 4 バイト値
PLIRETV 組み込み関数	2 バイト値	4 バイト値の下位 2 バイト	4 バイト値
RETCODE オプション	R15 の下位 2 バイト	R15 の下位 2 バイト	2 バイト値

PLIRETC の場合、再リンクされた OS PL/I ロード・モジュールは、4 バイト・ユーザー戻りコード値を設定できます。

言語環境プログラムでは、PLISRTx 呼び出しから戻るときに、PL/I ユーザー戻りコードは常にゼロにリセットされます。このことは、OS PL/I ランタイムには当てはまりません。

ランタイム・メッセージにおける相違点

ランタイム・メッセージの形式と内容が異なります。 ランタイム・メッセージを分析するアプリケーションを使用する場合は、相違点を考慮してアプリケーションを変更する必要があります。

次のリストで、相違点のいくつかについて説明します。

- メッセージ接頭語のメッセージ番号は、3 桁でなく 4 桁になり、IBM*nnnnx* の形式になります。ただし *nnnn* はメッセージ番号を表し、*x* はメッセージの重大度を表します。
- メッセージ接頭語のメッセージ重大度は、I、W、E、S、または C のいずれかです。
- 一部の英大/小文字混合のメッセージ・テキスト、および日本語メッセージが拡張されました。 大文字の英語メッセージのメッセージ・テキストについては変更されていません。

詳しくは、言語環境プログラム デバッグのガイドおよびランタイム・メッセージを参照してください。

言語環境プログラムの環境では、ランタイム・メッセージはランタイム・オプション MSGFILE に指定された MSGFILE 宛先に送られます。デフォルトの MSGFILE 宛先は SYSOUT です。 ユーザー出力は、現在も SYSPRINT に出力されます。Enterprise PL/I では、ランタイム APAR PQ78307 用の PTF を適用した後のみに MSGFILE(SYSPRINT) はサポートされます。MSGFILE オプションについて詳しくは、*z/OS* 言語環境プログラム プログラミング・ガイドを参照してください。

PLIDUMP の相違点

PLIDUMP は、言語環境プログラム形式のダンプを生成するようになりました。PLIDUMP の使用法とダンプ出力が異なっています。

以下に、PLIDUMP の使用方法と、生成される出力の相違点をリストします。 コンパイル単位 は外部プロシージャーの 1 次エントリー・ポイントを示し、コンパイル単位名 は外部プロシージャーの名前を示します。

- ダンプ出力ファイルの DD 名は、CEEDUMP、PLIDUMP、または PL1DUMP のいずれかです。 これらのファイルのいずれかを定義しない場合、言語環境プログラムはデフォルトの CEEDUMP ファイルを作成してダンプ出力を格納します。 ダンプ出力ファイルの LRECL は、ダンプ・レコードのラッピングを防ぐために、OS PL/I が必要とする 121 バイトではなく、少なくとも 133 バイトにする必要があります。
- PLIDUMP の 16 進 (H) オプションを使用する場合は、MVS に対して DD 名 CEESNAP を指定するか、または VM に対してファイル名 CEESNAP を指定する必要があります。指定しない場合、H オプションは無視されます。このデータ・セットには、スナップ・ダンプ出力が含まれます。

  MVS の環境で 16 進 (H) オプションを指定した場合、スナップからの出力には、すべてのシステム制御プログラム情報が含まれます (SDATA=ALL)。 OS PL/I は、部分的な情報だけを提供します (SDATA=CB、Q、および TRT)。
- ILC を使用した場合、ダンプ出力には他の言語 (例えば C/C++ または COBOL) に関連した情報が含まれます。
- ID 文字ストリングは、OS PL/I がサポートする 90 バイトではなく、60 バイトに制限されます。
- トレースバック・セクションは、それぞれのエントリー・ポイント名に関連したコンパイル単位名をリストします。 エントリー・ポイントが 2 次エントリー・ポイントである場合、実際のエントリー・ポイントに関連した 1 次エントリー・ポイント名はリストされません。

  トレースバック・セクションには、コンパイル単位 のアドレスに対するオフセット、および実際のエントリー・ポイントのアドレスに対するオフセットも含まれます。
- ランタイム・メッセージは、独立したセクションにあり、トレースバック・セクションの一部ではなくなりました。
- PLIDUMP の BLOCK (B) オプションを指定すると、条件処理ルーチンの保管域がダンプのブロック・セクションに出力されます。 PLIDUMP の BLOCK オプションを指定しない場合、条件処理ルーチンの保管域はダンプに出力されません。

- プログラムが TEST コンパイル時オプション付きでコンパイルされ、開始ブロックにラベルがある場合、その開始ブロックは `Label:BEGIN block.` として識別されます。 それ以外の場合、その開始ブロックは `%BLOCK`*nn* として識別されます。ここで *nn* は、その開始ブロックのブロック数です。
- コンパイラー生成 ILC サブルーチンがトレースバック・セクションに表示されるようになりました。 これらのサブルーチンは、コンパイル単位名 として識別され、サフィックス ILC と連結されます。
- 言語環境プログラムで定義済みの Program Prologue Area (PPA) を持つ PL/I ライブラリー・ルーチンは、ダンプ内の名前で識別されます。 ライブラリー・ルーチンが言語環境プログラムの PPA を持たない場合、これらのルーチンは `Library(PL/I)` として識別されます。
- STATIC ストレージの 16 進ダンプが言語環境プログラム定様式ダンプに含まれています。ダンプが行われたときにコンパイル単位からのルーチンが 1 つより多くスタックに存在する場合は、静的ストレージは、そのコンパイル単位に対して 1 回のみダンプされます。
- PL/I ルーチンを模擬するための規則に準拠したアセンブラー・ルーチンは、ダンプ出力内では CSECT 名によって識別されます。
- PLIDUMP は、各国語サポートの標準に準拠するようになりました。
- PLIDUMP は、言語環境プログラムの複数の別プログラムにわたる情報を提供できます。例えば、ある別プログラム内で実行されているアプリケーションがメイン・プロシージャーの FETCH (もう 1 つの別プログラムを作成するアクション) を行った場合、PLIDUMP には両方のプロシージャーに関する情報が入ります。

ストレージ報告書における相違点

ランタイム・ストレージ報告書の形式、内容、および宛先が変更されました。

言語環境プログラムは、OS PL/I と同等なストレージ情報を提供します。ストレージ報告書についての詳細は、「*z/OS* 言語環境プログラム プログラミング・リファレンス」で説明されています。

ランタイム・ストレージ報告書の見出しを指定するために、PLIXHD 宣言は使用されなくなりました。 代わりに、言語環境プログラムの呼び出し可能サービス CEE3RPH を使用して見出しを指定してください。 CEE3RPH を使用しない場合、見出しには実行のメイン・プロシージャー名、日付、および時刻が含まれます。

言語間通信 (ILC) のサポートの相違点

OS PL/I およびその他の言語環境プログラム以前の言語プログラムを含んだ ILC アプリケーションのサポートには、いくつかの制約事項があります。

この制約事項は、次の 3 つのグループに分けられます。

- 完全にサポートされるロード・モジュール

  OS PL/I、および言語環境プログラム以前の C/370 のプログラムを含んだロード・モジュールは、言語環境プログラム環境でサポートされます。
- 再リンクする必要のあるロード・モジュール

OS PL/I、および VS COBOL II リリース 3 (またはそれ以降) のプログラムを含んだロード・モジュールは、言語環境プログラムと再リンクする必要があります。

OS PL/I バージョン 2 リリース 3 ではマイグレーション・エイドとして APAR PN69803 および PN69804 が提供されるため、OS PL/I バージョン 2 リリース 3 環境下で再リンクを実行できます。アプリケーションが OS PL/I バージョン 2 リリース 3 環境で PN69803 および PN69804 と再リンクされている限り、このアプリケーションは 言語環境プログラム環境でサポートされます。 マイグレーション・エイドについての詳細は、 205 ページの『OS PL/I オブジェクト・モジュール再リンク・ツール - APAR PN69803』を参照してください。

- サポートされない ILC

  OS PL/I と下記の言語との間の ILC は、サポートされていません。

  – Fortran (言語環境プログラム リリース 5 より前)
  – OS/VS COBOL
  – VS COBOL II バージョン 1 リリース 2 またはそれ以前のリリース

詳しくは、「*Language Environment for OS/390 & VM ILC (*言語間通信*)* アプリケーションの作成」または「*z/OS Language Environment* 言語間通信 アプリケーションの作成」を参照してください。

ILC を使用する特定のアプリケーションの動作は、以下のように異なる場合があります。

- 条件処理の振る舞いが異なる場合があります。 条件処理の違いが生じる主な原因は、INTER オプションが無視されるようになったことと、PL/I 以外のルーチン内で発生した条件を、PL/I の条件処理機能が INTER の指定の有無に関係なく処理できるようになったことです。
- OS PL/I の環境では、ILC を使用するアプリケーション内で、関連した言語 (PL/I を含む) の環境の初期化と終了が複数回行われる可能性がありました。 言語環境プログラムの環境では、ランタイム環境は 1 つしか存在せず、言語固有の初期化と終了は 1 回だけ行われます。振る舞いの変化は、ファイルのオープンとクローズ、割り振られたストレージのリリース、および設定された ON ユニットの起動などに現れます。

  注**:** ランタイム環境を管理するためのコードを独自に設計した場合は、マイグレーション作業の一環としてそのコードを除去する必要があります。この専用 コードは言語環境プログラムと互換性がなく、ランタイム環境と競合します。

ILC が言語環境プログラムのランタイム環境でどのように動作するかについて詳しくは、「*Language Environment for OS/390 & VM ILC (*言語間通信*)* アプリケーションの作成」または「*z/OS Language Environment* 言語間通信 アプリケーションの作成」を参照してください。

アセンブラー・サポートにおける相違点

言語環境プログラムでは、PL/I ルーチンを呼び出すアセンブラー・プログラムは、言語環境プログラムによって定義されている呼び出し規則 (例えば、レジスター 13 が保管域を指していること、保管域の反復チェーニングが正しく行われていること、保管域の最初のワードが 0 であることなどが必要) に従う必要があります。

詳しくは、「*z/OS* 言語環境プログラム プログラミング・ガイド」を参照してください。

OS PL/I メインプログラムがアセンブラー・プログラムから呼び出される場合、アセンブラー・プログラムを変換して、言語環境プログラム準拠のアセンブラーを使用するには、OS PL/I プログラムを OPTIONS(MAIN) を指定せずに再コンパイルするか、または、エントリー・ポイント受取制御が PL/I プログラムの本当のエントリー・ポイントになるようにする必要があります。 どちらの場合にも、呼び出し先の PL/I プログラムはサブルーチンとして扱われます。 両方の場合とも、呼び出し先のプログラムは同じ言語環境プログラムの別プログラムの下で実行されます。この別プログラムでは、アセンブラー・プログラムがメインプログラムで、呼び出し先の PL/I プログラムがサブルーチンです。

言語環境プログラムに準拠するアセンブラーのメインプログラムは、言語環境プログラム PL/I 固有のランタイム環境が初期化されるようにするために OS PL/I サブルーチンを呼び出す場合、言語環境プログラムの PL/I for MVS & VM のシグニチャー CSECT である CEESG010 を明示的にインクルードする必要があります。 言語環境プログラム準拠のアセンブラーが OS PL/I サブルーチンに制御を渡すには、次の 3 つの方法があります。

1. 静的にリンクされた PL/I サブルーチンに分岐する。
2. 言語環境プログラムのマクロである CEELOAD を使用して、別個にリンクされた PL/I サブルーチンに分岐する。
3. LOAD や BALR などのアセンブラー命令を使用して、別個にリンクした PL/I サブルーチンに分岐する。

アセンブラーからの LINK の条件処理動作が明確に定義されました。 詳しくは、「*z/OS* 言語環境プログラム プログラミング・ガイド」を参照してください。

メイン・パラメーター・リストを検出するアセンブラー・プログラム

PL/I メインプログラムに渡されるパラメーター・リストを検出するために、保管域の反復チェーニングを使用する PL/I から呼び出されるアセンブラー・プログラムは、言語環境プログラムで実行しても、もう作動しません。これは、アセンブラー・プログラムと、PL/I メインプログラムを呼び出したプログラムの保管域との間の保管域の数が変更されたためです。

PL/I メインプログラムに渡されるパラメーター・リストを検出するのに必要なアセンブラー・プログラムは、言語環境プログラム EDB 内の CEEEDB_R13_PARENT フィールドを使用して、PL/I メインプログラムを呼び出したプログラムの保管域アドレスを取得できます。

第 7 章 オブジェクト・モジュールおよびロード・モジュールに関する考慮事項

この章では、言語環境プログラムで OS PL/I のオブジェクト・モジュールおよびロード・モジュールの互換性に影響を与える要因について説明します。

この章の説明には、OS PL/I バージョン 1 および OS PL/I バージョン 2 のオブジェクト・モジュールとロード・モジュールが含まれます。

1 つのロード・モジュール内のすべてのライブラリー・ルーチンは、ランタイム・ライブラリーの同じリリースのものである必要があります。 例えば、言語環境プログラム・スタブ、OS PL/I 共用ライブラリー・スタブ、および OS PL/I 常駐ライブラリー・ルーチンは、同じロード・モジュール内に存在することはできません。

ライブラリーを言語環境プログラムのランタイム環境にマイグレーションするために利用できるツールを探すには、 203 ページの『付録 A. 変換とマイグレーションの支援機能』を参照してください。

OS PL/I バージョン 1 のオブジェクト・モジュールおよびロード・モジュールの互換性

言語環境プログラムは、OS PL/I バージョン 1 のオブジェクト・モジュールとロード・モジュールを、いくつかの制限付きでサポートします。 バージョン 1 のオブジェクト・モジュールおよびロード・モジュールのほとんどは、このセクションで説明する規則を守る限り、使用を継続できます。

ロード・モジュールが、OS PL/I バージョン 1 のオブジェクト・モジュールを含んでいるが、OS PL/I バージョン 2 の常駐ライブラリーにリンクされている場合、そのロード・モジュールは OS PL/I バージョン 2 のロード・モジュールと見なされ、OS PL/I バージョン 2 の規則が適用されます。しかし、そのロード・モジュールが OS PL/I バージョン 1 リリース 1.0 から 2.3 までのオブジェクト・モジュールを含んでいる場合、そのオブジェクト・モジュールを再コンパイルする必要があります。

ロード・モジュールが OS PL/I の異常終了出口である IBMBEER を含んでいる場合、その異常終了出口は言語環境プログラムによって無視されます。 このトピックについての詳細は、 186 ページの『アセンブラー・ユーザー出口の使用に関する考慮事項』を参照してください。

OS PL/I バージョン 1 リリース 5.1

オブジェクト・モジュール

オブジェクト・モジュールはサポートされます。

© Copyright IBM Corp. 1999, 2016

共有ライブラリーを使用しないロード・モジュール

- 非 CICS、非マルチタスキングの MVS 用メイン・ロード・モジュール

  OS PL/I のブートストラップ・ルーチンである IBMBPIRA は、常にユーザー・ロード・モジュールとリンクされ、また、言語環境プログラムとは互換性のない、高速な初期化と終了などの機能を含んでいます。 サンプルの ZAP である IBMRZAPM は、言語環境プログラムの SCEESAMP で提供されており、それらの非互換機能を非活動化するために役立ちます。 サンプルの ZAP については、 204 ページの『OS PL/I バージョン 1 リリース 5.1 メイン・ロード・モジュール ZAP』で説明されています。

  ZAP されたロード・モジュールは、言語環境プログラムだけでなく、OS PL/I V1.5.1 および V2 でも継続して動作しますが、元のロード・モジュールに高速な初期化と終了機能が含まれている場合には、性能低下が発生する可能性があります。

  ロード・モジュールに対して ZAP を実行しない場合、以下のいずれかの手順に従う必要があります。

  – オブジェクト・モジュールを言語環境プログラムまたは OS PL/I バージョン 2 と再リンクする。
  – 203 ページの『OS PL/I ルーチン置換ツール』を使用し、ロード・モジュール内のライブラリー・ルーチンを言語環境プログラム・スタブで置き換える。

- CICS 環境以外のマルチタスキングの MVS 用メイン・ロード・モジュール

  ロード・モジュールはサポートされます。

- CICS 環境のメイン・ロード・モジュール

  ロード・モジュールはサポートされます。

- VM 環境のメイン・ロード・モジュール

  OS PL/I の VM 固有ブートストラップ・ルーチンである DMSIBM は、言語環境プログラムとは互換性のない機能を含んでいます。サンプルの ZAP である IBMRZAPV は、言語環境プログラムの SCEESAMP で提供されており、非互換機能を非活動化するために役立ちます。 サンプルの ZAP については、 204 ページの『OS PL/I バージョン 1 リリース 5.1 メイン・ロード・モジュール ZAP』で説明されています。

  ZAP されたロード・モジュールは、言語環境プログラム環境でのみサポートされます。 OS PL/I バージョン 1 または バージョン 2 環境では動作しません。ロード・モジュールに対して ZAP を実行しない場合、以下のいずれかの手順に従う必要があります。

  – オブジェクト・モジュールを言語環境プログラムまたは OS PL/I バージョン 2 と再リンクする。
  – 203 ページの『OS PL/I ルーチン置換ツール』を使用し、ロード・モジュール内のライブラリー・ルーチンを言語環境プログラム・スタブで置き換える。

- フェッチされたサブルーチン・ロード・モジュール

ロード・モジュールはサポートされます。

共有ライブラリーを使用するロード・モジュール

ロード・モジュールは、OS PL/I V1R5.1 共用ライブラリーがすべての PLRSHR オプションを指定して作成され、かつ、共用ライブラリーが、マルチタスキング共用ライブラリーを含めて、言語環境プログラム・スタブと置き換えられている限りでサポートされます。共用ライブラリーの置き換えは、言語環境プログラムのインストール中に 1 回だけ行う必要があります。

共用ライブラリーがすべての PLRSHR オプションを指定して作成されていないか、または共用ライブラリーが言語環境プログラム・スタブと置き換えられていない場合、オブジェクト・モジュールを言語環境プログラムまたは OS PL/I バージョン 2 と再リンクするか、またはロード・モジュール内の共用ライブラリー・スタブを言語環境プログラム・スタブと置き換える必要があります。オブジェクト・モジュールを再リンクするか、またはロード・モジュールを置き換えた後には、OS PL/I 共用ライブラリー機能を使用することはありません。

Enterprise PL/I は共用ライブラリーをサポートしないことに注意してください。Enterprise PL/I にマイグレーションする予定である場合は、共用ライブラリーの使用を停止する必要があります。 言語環境プログラム環境では、PL/I はフルサイズの常駐モジュールの代わりにスタブを使用するため、共用ライブラリーを使用する必要はありません。

OS PL/I バージョン 1 リリース 5

OS PL/I バージョン 1 リリース 5 は、MVS アプリケーションに対するサポートだけを提供します。 VM および CICS は、リリース 5.0 ではサポートされません。

オブジェクト・モジュール

オブジェクト・モジュールはサポートされます。

ロード・モジュール

ロード・モジュールは、共用ライブラリーを使用するかどうかに関係なく、サポートされません。オブジェクト・モジュールを言語環境プログラムまたは OS PL/I バージョン 2 と再リンクするか、あるいは 203 ページの『OS PL/I ルーチン置換ツール』 を使用して、ロード・モジュール内のライブラリー・ルーチンを言語環境プログラム・スタブと置き換える必要があります。

OS PL/I バージョン 1 リリース 3.0 ～ リリース 4.0

オブジェクト・モジュール

- MVS 環境

  オブジェクト・モジュールは、CICS マクロ言語を除き、サポートされます。

- VM 環境

  オブジェクト・モジュールはサポートされます。

ロード・モジュール

ロード・モジュールは、共用ライブラリーを使用するかどうかに関係なく、サポートされません。オブジェクト・モジュールを言語環境プログラムまたは OS PL/I バージョン 2 と再リンクするか、あるいは 203 ページの『OS PL/I ルーチン置換ツール』 を使用して、ロード・モジュール内のライブラリー・ルーチンを言語環境プログラム・スタブと置き換える必要があります。

バージョン 1 リリース 3.0 より前の OS PL/I

リリース 3.0 より前に作成されたオブジェクト・モジュールまたはロード・モジュールはサポートされないため、言語環境プログラムでサポートされる PL/I コンパイラーまたは OS PL/I バージョン 2 を使用して、アプリケーションを再コンパイルする必要があります。

OS PL/I バージョン 2 のオブジェクト・モジュールおよびロード・モジュールの互換性

ほとんどの場合、OS PL/I バージョン 2 で作成されたオブジェクト・モジュールおよびロード・モジュールは、再リンクする必要はありません。

このマイグレーション・ガイドの前半のいくつかのセクションでは、OS PL/I の機能を詳細に説明しています。 これらの機能の一部は確かに再リンクを必要としますが、そのいくつかはサポートされなくなっています。

言語環境プログラムは、PL/I アセンブラー・ユーザー出口である IBMxXITA を含んだ OS PL/I アプリケーションをサポートします。 このトピックについての詳細は、 186 ページの『アセンブラー・ユーザー出口の使用に関する考慮事項』を参照してください。

OS PL/I オブジェクト・モジュールおよびロード・モジュールのサポートの要約

このトピックでは、このセクションで説明されている PL/I オブジェクト・モジュールおよびロード・モジュールのサポートが要約されています。

サポートに対する例外は脚注に示されており、関連するセクション内で説明されています。

表 10. 言語環境プログラムによるオブジェクト・モジュールおよびロード・モジュールのサポートの有無についての要約

サポートの説明	**V2**	**V1R5.1**	**V1R5.0**	**V1R3.0-V1R4.0**	**V1R3.0** より前
メイン・ロード・モジュール	有[3]	有[1, 3]	無	無	無
フェッチされたサブルーチン・ロード・モジュール	有[3]	有[3]	無	無	無
オブジェクト・モジュール	有	有	有	有[2]	無

表 *10.* 言語環境プログラムによるオブジェクト・モジュールおよびロード・モジュールのサポートの有無についての要約 *(*続き*)*

サポートの説明	**V2**	**V1R5.1**	**V1R5.0**	**V1R3.0-V1R4.0**	**V1R3.0** より前
例外:					
1. MVS 非 CICS 非マルチタスキング・ロード・モジュールおよび VM ロード・モジュールは、特定の処置が取られない限りサポートされません。これらのモジュールのサポートを使用可能にするために必要なアクションについては、 54 ページの『共有ライブラリーを使用しないロード・モジュール』を参照してください。					
2. CICS マクロ言語は、 55 ページの『OS PL/I バージョン 1 リリース 3.0 ～ リリース 4.0』の 55 ページの『オブジェクト・ モジュール』で説明した通り、サポートされません。					
3. 共用ライブラリーは、すべての PLRSHR オプションを指定して作成し、かつ言語環境プログラム・スタブと置き換える必要があります。そのために必要な処置については、 46 ページの『OS PL/I 共用ライブラリーのサポートの相違点』を検討してください。					

第 8 章 リンク・エディットに関する考慮事項

この章では、OS PL/I によって作成されたオブジェクト・モジュールをリンク・エディットする際に考慮する必要のある要因について説明します。

説明するトピックには、シンボル・テーブルと数学ルーチンが含まれています。

SCEERUN

OS PL/I アプリケーションを言語環境プログラム環境で実行し、既存の JCL を使用する場合は、すでに SCEERUN を含んでいる TASKLIB または LINKLIB を使用する場合を除き、STEPLIB ステートメントまたは JOBLIB ステートメントに SCEERUN を必ず組み込んでください。

シンボル・テーブルに関する考慮事項

PL/I の異なるリリースで作成されたオブジェクト・モジュールをリンク・エディットする場合、またそのオブジェクト・モジュールに外部変数のシンボル・テーブルが含まれている場合、結果として生成されるロード・モジュールに現れるシンボル・テーブルは、PL/I の最新のリリースによって作成されたものである必要があります。

プログラムに、外部変数に対する下記の PL/I 機能が 1 つ以上含まれている場合、コンパイラーは外部シンボル・テーブル制御セクション (CSECT) を含んだオブジェクト・モジュールを作成します。

- GET DATA ステートメント
- PUT DATA ステートメント
- TEST(SYM) コンパイル時オプション

プログラムが、外部変数に関するこれらの機能を 1 つ以上使用している場合には、ロード・モジュールに正しいシンボル・テーブルが現れるようにする必要があります。 PL/I の最新のリリースによって作成されたオブジェクト・モジュールは、リンク・エディット・ジョブ・ストリーム内の他のすべてのオブジェクト・モジュールより前に置いてください。 複数のオブジェクト・モジュールによって同じ名前のシンボル・テーブル CSECT が作成された場合、リンケージ・エディターは、最初に検出したシンボル・テーブル CSECT を保持し、他のシンボル・テーブルを破棄します。

例えば、OS PL/I バージョン 1 リリース 5.1 によって作成されたオブジェクト・モジュールを、OS PL/I バージョン 2 リリース 3 によって作成されたオブジェクト・モジュールと共にリンク・エディットするとします。 OS PL/I バージョン 2 リリース 3 によって作成されたオブジェクト・モジュールは、リンク・エディット・ジョブ・ストリーム内では、OS PL/I バージョン 1 リリース 5.1 によって作成されたオブジェクト・モジュールより前に置いてください。 それにより、両方の

© Copyright IBM Corp. 1999, 2016

オブジェクト・モジュールがシンボル・テーブルを作成した場合、リンケージ・エディターは OS PL/I バージョン 2 リリース 3 によって作成されたシンボル・テーブルを保持します。

NCAL リンケージ・エディター・オプション

言語環境プログラム環境では、NCAL リンケージ・エディター・オプションは、サブルーチン・オブジェクト・モジュールを今後の使用のためにリンク・エディットする際に、引き続き必要となります。

ロード・モジュールには、言語環境プログラム・スタブも OS PL/I 常駐ライブラリー・ルーチンも含めることはできません。

ENTRY カード

OS PL/I コンパイラーでコンパイルされた MAIN プログラムのエントリー・ポイントは PLISTART ですが、MAIN プログラムが PL/I for MVS & VM コンパイラーまたはそれ以降のコンパイラーでコンパイルされた場合はエントリー・ポイントは CEESTART です。

バッチ・アプリケーションのビルド時に ENTRY カードを使用する必要がある場合は、OS PL/I コンパイラーでコンパイルしたプログラムには CEESTART を使用しないでください。

OS PL/I 数学ルーチンの使用

言語環境プログラムは、指数用のルーチンを含む、数学ルーチンのセットを提供しています。最も一般的に使用されるルーチンの場合、言語環境プログラムは OS PL/I よりも正確な結果を出します。

言語環境プログラムの一部のルーチンは、パフォーマンスの点でも OS PL/I より優れています。 数学ルーチンは、言語環境プログラムで提供されるものを使用してください。

言語環境プログラムでは、言語環境プログラムへのマイグレーションを容易にするために、OS PL/I 数学ルーチンも提供しています。ただし、OS PL/I 数学ルーチンは互換性を維持するためだけに提供されており、将来には提供が中止されます。

アプリケーションが言語環境プログラム環境で OS PL/I 数学ルーチンを使用しなければならない場合は、オブジェクト・モジュールをリンク・エディットする際に、SIBMMATH を SCEELKED の前に置いてください。

Enterprise PL/I for z/OS には、OS PL/I にあった数学ルーチンは用意されていません。

第 9 章 サブシステムに関する考慮事項

この章では、CICS、IMS、および DB2 の環境で稼働するアプリケーションをマイグレーションする際に知っておく必要がある、サブシステム固有の考慮事項について説明します。

CICS に関する考慮事項

言語環境プログラムは、非 CICS に対する場合と同じレベルで、OS PL/I のオブジェクト・モジュールおよびロード・モジュールのサポートを提供します。

詳しくは、 53 ページの『第 7 章 オブジェクト・モジュールおよびロード・モジュールに関する考慮事項』を参照してください。

CICS バージョン 3 リリース 3 環境を使用している場合は、CICS APAR PN38032 をインストールしておく必要があります。PN38032 がインストールされていない場合、言語環境プログラムの使用を試行したアプリケーションは、APLE 異常終了を受け取ります。

CICS ストレージ保護機能は、CICS 3.3 で導入されました。この機能により、アプリケーション・プログラムや特に CICS 領域全体のデータ保全性とセキュリティーが強化されます。 この新機能が原因で、正常に実行されていた OS PL/I アプリケーションが、ASRA(0C4) 異常終了と CICS メッセージ DFHSR0622 を出して失敗する可能性があります。

上記の問題が OS PL/I アプリケーション・プログラムで発生した場合は、以下の 2 つの方法のいずれかを実行することで、問題を修正できる可能性があります。

1. CICS システム初期化パラメーター RENTPGM=NOPROTECT を設定する。このパラメーターは、ユーザー・キーによるユーザー・プログラムの保護を設定します。 RENTPGM のデフォルトは PROTECT です。
2. OS PL/I アプリケーション・プログラムを、APAR PN38032 がインストールされている言語環境プログラム環境で再リンクする。

OS PL/I CICS アプリケーションでストリーム出力機能、特に `PUT DATA;` ステートメントを使用している場合には、上記のエラーが発生する可能性があります。 PL/I ストリーム出力機能は、デバッグだけを目的としています。 パフォーマンス上の理由から、実動プログラムではこの機能を使用しないようにお勧めします。

CICS システム定義 (CSD) ファイルの更新

言語環境プログラムの環境で CICS 領域を起動する場合は、言語環境プログラム CEECCSD にリストされているモジュール名が、CSD 内で定義されていることを確認する必要があります。

CEECCSD は SCEESAMP 内にあります。 CICS 第 4 版の自動インストール機能を使用する場合は、CSD 内で言語環境プログラムのモジュールを手動で定義する必要はありません。

© Copyright IBM Corp. 1999, 2016

エラー処理

診断メッセージは、ERROR ON ユニットがプログラム内で設定されていないか、または ERROR ON ユニットがブロックから抜ける GOTO を使用することで条件から回復していない場合に限って発行されます。

CICS 環境でのユーザー作成条件ハンドラーの制限

いくつかの EXEC CICS コマンドは、CEEHDLR によって設定したユーザー作成条件ハンドラー内およびユーザー作成条件ハンドラーから呼び出されたルーチン内では使用できません。

以下の EXEC CICS コマンドは許可されていません。

- EXEC CICS ABEND
- EXEC CICS HANDLE AID
- EXEC CICS HANDLE ABEND
- EXEC CICS HANDLE CONDITION
- EXEC CICS IGNORE CONDITION
- EXEC CICS POP HANDLE
- EXEC CICS PUSH HANDLE

他のすべての EXEC CICS コマンドは、ユーザー作成条件ハンドラー内でも使用できます。 しかし、それらは NOHANDLE オプション、RESP オプション、または、RESP2 オプションを使用してコーディングされていなければなりません。これにより、CICS サービスの障害のために新たな条件が発生することを防ぎます。

マクロ・レベル・インターフェース

CICS マクロ・レベル・インターフェースはサポートされません。

PL/I MAIN プロシージャーの FETCH

CICS は、PL/I による PL/I MAIN プロシージャーの FETCH をサポートしません。

STACK ランタイム・オプション

このトピックには、STACK ランタイム・オプションのための言語環境プログラム・サポートの概要が記載されています。

言語環境プログラムは、ランタイム・オプション STACK(,,ANY) を使用する PL/I for MVS & VM アプリケーションをサポートします。

また、言語環境プログラムは、言語環境プログラムと再リンクされている OS PL/I アプリケーションについても、そのアプリケーションが次の条件を満たしている限り、STACK(,,ANY) をサポートします。

- アプリケーションが、編集されたストリーム入出力を含んでいない (例えば、PUT ステートメント内で EDIT が使用されなかった)。
- アプリケーションが AMODE(31) を指定している。

ランタイム出力

プログラムを DISPLAY(STD) を指定してコンパイルすると、すべてのランタイムの出力は、CICS 一時データ・キュー CESE に送信されます。

プログラムを DISPLAY(WTO) を指定してコンパイルすると、DISPLAY の出力は、CICS JESLOG に経路指定されます。 すべての他のランタイムの出力は、CICS 一時データ・キュー CESE に送信されます。

言語環境プログラムは、CICS 環境における MSGFILE オプションを無視します。出力データ・キューのフォーマットは、図 1に示されています。

ASA	端末 ID	トランザクション ID	B	日時 YYYYMMDDHHMMSS	B	データ

図 *1. CESE* の出力データ・キュー

また、PL/I 一時キュー CPLI および CPLD は、使用されなくなりました。 このため、CICS 宛先管理テーブル (DCT) 内で CPLI と CPLD のエントリーを指定する必要はありません。

CICS 環境で PL/I によって使用される異常終了コード

OS PL/I バージョン 2 の環境で発行されていた APL*x* 異常終了コードは、発行されなくなりました。 その代わり、言語環境プログラムの定義による異常終了コードが発行されます。

言語環境プログラムの異常終了コードについて詳しくは、「*z/OS* 言語環境プログラム ランタイム・メッセージ」を参照してください。

IMS に関する考慮事項

言語環境プログラムは IMS に対し、非 IMS に対する場合と同じレベルで OS PL/I のオブジェクトおよびロード・モジュールのサポートを提供します。

詳しくは、 53 ページの『第 7 章 オブジェクト・モジュールおよびロード・モジュールに関する考慮事項』を参照してください。

IMS へのインターフェース

言語環境プログラムは、PL/I ルーチンからの PLITDLI、ASMTDLI、および EXEC DLI の各インターフェースをサポートしています。

SYSTEM(IMS) コンパイル時オプション

OS PL/I バージョン 2 で使用可能な SYSTEM(IMS) オプションは、PL/I IMS アプリケーション専用にサポートされていました。

IMS アプリケーションのメイン・プロシージャーでは、パラメーターに POINTER データ型を使用する必要があります。

IMS での PLICALLA サポート

OS PL/I PLICALLA エントリー・ポイントは、言語環境プログラム環境でサポートされますが、これは IMS 用の推奨インターフェースではなく、サポートは一定の時点で打ち切られる可能性があります。

代わりに、SYSTEM(IMS) コンパイル時オプションと、PLISTART エントリー・ポイントまたは CEESTART エントリー・ポイントを使用してください。

言語環境プログラムは、OS PL/I PLICALLA アプリケーションに対しても、同様のサポートを提供します。ただし、メインのロード・モジュールを PL/I for MVS & VM を使用して再コンパイルし、かつ PLICALLA の使用を継続する場合には、追加の規則に従う必要があります。詳しくは、42 ページの『PLICALLA に関する考慮事項』を参照してください。

サポートされている PSB 言語オプション

言語環境プログラムは、IMS のサポート対象のリリースで、次の PSBGEN LANG オプションを使用する PL/I アプリケーションをサポートします。

IMS/ESA バージョン 4

表 11 は、IMS/ESA バージョン 4 リリース 1 以降の PSB LANG オプションのサポートを示しています。

表 *11. IMS/ESA バージョン 4 リリース 1 以降の PSB LANG オプション*

SYSTEM オプション	エントリー・ポイント	**LANG=**
IMS	CEESTART、PLISTART	PLI、または PASCAL を除くその他の値
IMS	PLICALLA[1]	PLI
省略	CEESTART、PLISTART	無効
省略	PLICALLA[1]	PLI

注:

1. 互換性のみのためにサポートされます。

ストレージの使用に関する考慮事項

IMS/ESA 第 3 版リリース 1 では、IMS インターフェースに渡されるパラメーターは、16M ラインより下のエリアに制限されなくなりました。

下記の方法を使用すると、ほとんどの場合、パラメーターは 16M ラインより上に配置されます。

- HEAP ランタイム・オプションの ANYWHERE サブオプションを使用する。これは、CONTROLLED または BASED 属性を持つ変数に適用されます。その理由は、これらの変数がストレージをヒープから取得するためです。
- STACK ランタイム・オプションの ANYWHERE サブオプションを使用する。OS PL/I アプリケーションを言語環境プログラムと再リンクし、かつアプリケー

ションが編集されたストリーム入出力を使用しない場合、またはアプリケーションを PL/I for MVS & VM を使用して再コンパイルした場合、そのアプリケーションが AMODE(31) であれば、STACK(,,ANYWHERE) を使用できます。 この場合、自動ストレージ内の変数は、16M ラインより上に配置されます。

- パラメーターを静的ストレージに配置し、使用するロード・モジュールの属性を RMODE(ANY) にする。

IMS 環境での調整条件処理

言語環境プログラムおよび IMS の条件処理は調整されます。つまり、アプリケーションが IMS 環境で実行されている際に、プログラム割り込みや異常終了が発生すると、アプリケーション、または IMS のどちらで問題が発生したかが、言語環境プログラム条件マネージャーに通知されます。

問題が IMS 内で発生した場合、言語環境プログラム (および呼び出された HLL 固有の条件処理ルーチンと同様に) は条件を IMS に戻してパーコレートします。

言語環境プログラムのランタイム・オプション TRAP(ON) を使用すると、PL/I ルーチンから呼び出した PLITDLI および ASMTDLI の両インターフェースに対する調整条件処理が、継続して言語環境プログラムによってサポートされます。

IMS/ESA バージョン 3 (PTF UN4928 適用済み)、または IMS/ESA バージョン 4 を使用する場合、言語環境プログラムでは、CEETDLI、C ルーチンからの CTDLI、COBOL プログラムからの CBLTDLI、PL/I プログラムからの AIBTDLI、および非 PL/I プログラムからの ASMTDLI の調整条件処理もサポートされます。

IMS の外部のアプリケーション内でプログラム割り込みまたは異常終了が発生した場合、または重大度 2 以上のソフトウェア条件が IMS の外部で発生した場合、言語環境プログラムの条件マネージャーは、「*z/OS* 言語環境プログラム プログラミング・ガイド」に記述されている通常の条件処理を実行します。このケースで、IMS がデータベース・ロールバックを実行できるようにするためには、次のいずれかの処置を行う必要があります。

- アプリケーションが処理を続行できるように、エラーを完全に解決する。
- IMS に対してロールバック呼び出しを実行してから、アプリケーションを終了する。
- IMS ロールバックを引き起こすために、すべての異常終了をシステム異常終了に変換する ABTERMENC(ABEND) ランタイム・オプションを使用して、アプリケーションを異常終了させる。
- IMS ロールバックを引き起こすために、すべての異常終了をシステム異常終了に変換するように変更したアセンブラー・ユーザー出口 (CEEBXITA) を提供して、アプリケーションを異常終了させる。

  ユーザー提供のアセンブラー・ユーザー出口は、戻りコードと理由コードまたは CEEAUE_ABTERM ビットを検査し、該当する場合には CEEAUE_ABND フラグを ON にして異常終了を要求する必要があります。 詳しくは、「*z/OS* 言語環境プログラム プログラミング・ガイド」を参照してください。

ライブラリー保存 (LRR) によるパフォーマンスの向上

LRR を使用すると、パフォーマンスを高めることができます。

詳しくは、 145 ページの『CPU 使用効率の向上』を参照してください。

DB2 に関する考慮事項

DB2 の使用に関しては、 63 ページの『IMS に関する考慮事項』で説明した考慮事項のほかには、特別な考慮事項はありません。

第 4 部 新しいコンパイラーへの移行

© Copyright IBM Corp. 1999, 2016

第 10 章 新しいコンパイラーの制限について

新しいコンパイラーには、VM をサポートしないことの他にも、理解しておく必要のあるさまざまな制限があります。この章では、新旧のコンパイラーの相違点をリストし、説明します。

言語環境プログラム要件

Enterprise PL/I V4R2 は、言語環境プログラム z/OS 版 1.10 以降でのみサポートされます。

サポートされない言語

コンパイラーは、サポートされていないすべての言語にフラグを立てます。

マルチタスキング

旧版のコンパイラーでサポートされていたマルチタスキング言語は、新しいコンパイラーではサポートされません。

ただし、新しいコンパイラーはマルチスレッド化をサポートします。 しかし、マルチスレッド化機能を使用するには、コードは POSIX(ON) オプションを使用して実行しなければなりません。

マルチスレッド化ステートメントについて詳しくは「PL/I 言語解説書」を参照してください。

CHECK

PL/I for MVS & VM では、CHECK ステートメント、CHECK プレフィックス、および CHECK 条件のサポートを廃止しました。また、新しいコンパイラーでも、これらの構成はサポートされていません。

CHARSET(48) および **CHARSET(BCD)**

これらのオプションに対するサポートは、OS PL/I バージョン 2 で廃止されました。ただし、ソースを変換するツールが IBM から提供されています。

EGCS OS PL/I バージョン 1 では EGCS がサポートされていました。OS PL/I バージョン 2 は、その後継の GRAPHIC をサポートしているため、EGCS のサポートは廃止されました。 新しいコンパイラーでも、EGCS はサポートされていません。

Fortran

新しいコンパイラーは、Fortran パラメーターの再マップをサポートしません。特に、PL/I は、Fortran から PL/I へ渡された 2 次元配列を、あたかも置き換えられたように認識します。

無効なコード

新しいコンパイラーでは、旧コンパイラーで受け入れられていたことがあった場合でも、無効なコードはサポートされなくなりました。例えば、旧コンパイラーでは、(CHAR 組み込み関数の引数には計算タイプを指定する必要

© Copyright IBM Corp. 1999, 2016

があると旧コンパイラーでは説明していましたが) CHAR 組み込み関数を FILE VARIABLE に適用できました。新しいコンパイラーでは、そのような無効なコードに対しては、重大メッセージでフラグが立てられます。

言語制限

指示されている場合を除いて、コンパイラーは、制限されている言語の使用に対してフラグを立てます。

RECORD I/O

RECORD I/O がサポートされていますが、制約事項があります。

- 2.1 ギガバイトより大きい REGIONAL(1) ファイルはサポートされません。
- READ/WRITE ステートメントの EVENT 文節はサポートされません。
- UNLOCK ステートメントはサポートされません。
- 以下のファイル属性はサポートされません。
  - BACKWARDS
  - EXCLUSIVE
  - TRANSIENT
- 次に示す ENVIRONMENT 属性のオプションはサポートされませんが、それらの使用に対しては、LANGLVL(NOEXT) を指定した場合にのみフラグが立てられます。
  - ADDBUFF
  - ASCII
  - BUFFERS
  - BUFOFF
  - INDEXAREA
  - NCP
  - NOWRITE
  - REGIONAL(2)
  - REGIONAL(3)
  - SIS
  - SKIP
  - TOTAL
  - TP
  - TRKOFL

なお、ENVIRONMENT 属性の TOTAL オプションはサポートされていないため、TOTAL オプションを使用したファイルに対する入出力操作は一般に、旧コンパイラーの場合よりパフォーマンスが劣ります。

しかしながら、TOTAL オプションの旧実装では、ライブラリーの入出力制御ブロックのレイアウトおよび DFSMS 制御ブロックのレイアウトの両方について認識しているコンパイラー生成コードに依存していました。 すなわち、それらのレイアウトのいずれかが変更された場合、そのコードは破損することになります。したがって、ライブラリー・コードは変更できず、固定レベルの DFSMS 制御ブロックを使用し続ける (または使用しているように見せかける) 必要がありました。その結果の 1 つとして、旧ライブラリーでは、ユーザーのロード・モジュールのコンパイラー

生成コードでは、入出力バッファーが標準以下であることを認識しているため、標準以上のバッファーを使用できませんでした。 (また、Enterprise PL/I は、TOTAL オプションの実装を開始する場合、標準以上、または制限を超えるバッファーを使用できません。) この硬直的な相互依存性は、TOTAL オプションを使用しない場合でも障害を及ぼす不適切な設計です。

さらに、コードがすべてのライブラリー・コードを迂回し、エラー処理で少なくとも問題があると言えたため、TOTAL オプションには固有の危険がありました。

STREAM I/O

STREAM I/O がサポートされていますが、PUT/GET DATA ステートメントに適用される制限があります。

- 以下の条件がどちらも真であれば、DEFINED はサポートされません。
  - DEFINED 変数が BIT または GRAPHIC である。
  - DEFINED 変数が POSITION 属性を持っている。
- DEFINED は、その基本変数が配列スライス、または定義された変数とは異なる次元数を持った配列の場合、サポートされません。

構造式

すべての定数エクステントを指定するパラメーター記述がない限り、引数としての構造式はサポートされません。

しかし GENERIC 参照では、構造式はサポートされていません。 GENERIC 参照では、一致しないパラメーターおよび引数の構造体もサポートされていません。

配列式

配列式は、既知のサイズのスカラーの配列である場合を除いて、ユーザー関数への引数としては許可されません。 そのため、算術型のスカラーの配列をユーザー関数に渡すことはできますが、可変長ストリングの配列を渡すと、問題が発生する場合があります。

しかし GENERIC 参照では、配列式はサポートされていません。 GENERIC 参照では、一致しないパラメーターおよび引数の配列もサポートされていません。

次の例は、呼び出しでサポートされる数値配列式を示したものです。

```
dcl x entry, (y(10),z(10)) fixed bin(31);

call x(y + z);
```

次に示すプロトタイプ化されていない呼び出しは、サイズの不明なストリング式を必要とするので、この呼び出しに対してはフラグが立てられます。

```
dcl a1 entry;
dcl (b(10),c(10)) char(20) var;

call a1(b || c);
```

ただし、次に示すプロトタイプ化された呼び出しに対しては、フラグは立てられません。

```
dcl a2 entry(char(30) var);
dcl (b(10),c(10)) char(20) var;

call a2(b || c);
```

DEFAULT ステートメント

default を因数で指定することは、サポートされていません。

例えば、次のステートメントはサポートされません。

```
default ( range(a:h), range(p:z) ) fixed bin;
```

しかし、このステートメントを次のようにして、サポートされる同等のステートメントに変更することができます。

```
default range(a:h) fixed bin, range(p:z) fixed bin;
```

DEFAULT キーワードの後に "(" を使用することは、ANSI 規格の場合と同じ目的のために予約されています。この規格では、DEFAULT キーワードの後の属性内に、小括弧で囲んだ論理述部を置くことが許可されています。

自動変数の範囲

自動変数の範囲は、その自動変数を宣言したプロシージャー内でネストした関数、あるいは入り口変数によって設定することはできません。ただし、入り口変数を自動変数よりも前で宣言した場合は設定可能です。

組み込み関数

例外および制約事項付きでサポートされる、いくつかの組み込み関数があります。

- PLITEST 組み込み関数はサポートされません。
- DO ループで許可される疑似変数は、以下のものに制限されます。
  - IMAG
  - REAL
  - SUBSTR
  - UNSPEC
- POLY 組み込み関数には、次の制限があります。
  - 最初の引数は REAL FLOAT でなければなりません。
  - 2 番目の引数はスカラーでなければなりません。
- COMPLEX 疑似変数はサポートされません。
- RULES(NOLAXDCL) オプションで、そうした PL/I 組み込み関数がない場合、コンパイラーは、組み込み関数として DISPLAY などの名前の宣言にフラグを立てます。 より許容度の高い RULES(LAXDCL) オプションでも、そうした PL/I 組み込み関数がなく、コードが組み込み関数として (単なる宣言ではなく) 名前を使用しようとする場合、コンパイラーは、組み込み関数として DISPLAY などの名前の宣言にフラグを立てます。

DEFINED BIT 集合体

DEFINED 変数が、UNALIGNED NONVARYING BIT であるエレメントが入っている構造体または共用体である場合、DEFINED 変数内のすべての配列境界およびストリングの長さは、定数として指定する必要があります。

この制限に違反すると、コンパイラーは S レベル・メッセージ IBM1900I を出します。

OPTIONS(REENTRANT)

このオプションは PROCEDURE または BEGIN ステートメントの OPTIONS の一部ですが、無視されます。

z/OS プラットフォームでは、RENT コンパイラー・オプションを使用してコンパイルされたすべてのプログラムは再入可能で、他のプログラムは静的変数を変更していない場合 (NOWRITABLE コンパイラー・オプションの使用が必要となる場合もあります) には再入可能です。

iSUB 定義

iSUB 定義のサポートは、スカラー配列に制限されます。

LABEL 配列

Enterprise PL/I コンパイラーでは、ステートメント・ラベルの配列を宣言する必要はありません。

そうした配列を宣言する場合には、ストレージ・クラスなし (およびストレージ・クラスを意味するアクティブな DEFAULT ステートメントなし) で宣言するか、STATIC として宣言しなければなりません。従来の PL/I コンパイラーでは、こうした形で宣言するか、配列を AUTOMATIC として宣言する必要がありました。それで、どちらのコンパイラーでもコードが受け入れられるようにするには、配列を宣言しますが、AUTOMATIC または STATIC のどちらとしても宣言しないようにしなければなりません。

DBCS

DBCS は G 定数と M 定数、ID、およびコメントでのみ使用できます。

G リテラルは、前後を DBCS 引用符で囲み、その後に DBCS G または SBCS G を続けることによって指定できます。

GRAPHIC および POSITION

OS PL/I で、変数が GRAPHIC 属性および POSITION 属性を使用して宣言された場合、POSITION 値は基底付き変数へのバイトの数として解釈されます。Enterprise PL/I では、POSITION 値は基底付き変数への GRAPHIC 文字の数として解釈されます。

GRAPHIC と POSITION は BIT と POSITION に似ていて、この場合の POSITION 値は基底付き変数へのビットの数として解釈されます。

マクロ・プリプロセッサー

ストリング定数に続くサフィックスは、それが正しい PL/I サフィックスであるかどうかにかかわらず、マクロ・プリプロセッサーに置き換えられません。ただし、ストリング末尾の引用符とサフィックスの先頭文字の間に区切り文字を挿入した場合を除きます。

この変更は、OS PL/I V2R1 コンパイラーによって導入されたものであり、Enterprise PL/I コンパイラーと、PL/I for MVS & VM コンパイラーまたは OS PL/I V2Rx コンパイラーとの相違点ではないことに注意してください。 そのため、この制約事項にはフラグは立てられません。

例として、次のコードを考えてみます。

```
%DCL (GX, XX) CHAR;
%GX='||FX';
%XX='||ZZ';
DATA = 'STRING'GX;
DATA = 'STRING'XX;
DATA = 'STRING' GX;
DATA = 'STRING' XX;
```

OS PL/I V1 環境では、次のソースが生成されます。

```
DATA = 'STRING'||FX;
DATA = 'STRING'||ZZ;
DATA = 'STRING' ||FX;
DATA = 'STRING' ||ZZ;
```

Enterprise PL/I環境では、次のソースが生成されます。

```
DATA = 'STRING'GX;
DATA = 'STRING'XX;
DATA = 'STRING' ||FX;
DATA = 'STRING' ||ZZ;
```

オプション制限

一部のオプションまたはサブオプションは無視されます。

以下に示すコンパイラー・オプションには、制約事項があります。

- INCLUDE

  NOINCLUDE オプションはサポートされていません。旧版の INCLUDE オプションは、基本的には常に使用可能です。

- LANGLVL

  NOSPROG および SPROG サブオプションは、サポートされていません。SPROG は常に有効です。

  SAA コンパイラーは現在では使用されていないため、SAA および SAA2 サブオプションはサポートされません。

- LIST

  LIST オプションはサポートされますが、LIST オプションのサブオプションはサポートされません。新しいコンパイラーでは、1 つの列に、疑似アセンブラー・リストが常に表示されます。

- STMT

  STMT オプションはサポートされますが、現時点では、LIST、MAP、または OFFSET オプションによって生成された出力に対して効果はありません。
- SYSTEM

  CMS および CMSTPL オプションはサポートされません (VM がサポートされないため)。

サポートされないオプション

このトピックには、コンパイラーによってサポートされなくなったコンパイラー・オプションがリストされています。

- CONTROL
- COUNT

  COUNT オプションはサポートされません。また、PL/I for MVS & VM コンパイラーでもサポートされていません。
- DECK
- ESD

  ESD オプションはサポートされていませんが、LIST または MAP オプションが有効である場合には、外部シンボル辞書が生成されます。
- FLOW

  FLOW オプションはサポートされません。また、PL/I for MVS & VM コンパイラーでもサポートされていません。
- GOSTMT
- HGPR
- IMPRECISE
- LMESSAGE
- SEQUENCE
- SIZE
- SMESSAGE

コンパイラーに対するその他のインターフェースに関する制約事項

バッチ・コンパイル

PROCESS によって区切られたチャンク内では、コンパイルは実行されません。

この違いにより、以下の結果が生じます。

- 以降の PROCESS ステートメントのセットに対するオプションは無視されます。
- TEXT デックまたは .o が 1 つ作成されます。
- 一連のメッセージが入ったリスト・ファイルが 1 つ作成されます。
- 同名の外部変数は一致している必要があります。

次に、バッチ・コンパイルの例を示します。 この場合、*b* と *x* のミスマッチに関しては、新しいコンパイラーでのみフラグが立てられます。

```
*process opt(0);

 a: proc;
   dcl b ext entry(1,2 char(2), 2 char(2));
   dcl
     1 x ext,
       2 x1a char(2),
       2 x1b char(2);

   call b(x);
 end;

*process opt(2);

 b: proc(p);
   dcl p pointer;
   dcl
     1 x ext,
       2 x1a bit(16),
       2 x1b bit(16);

 end;
```

バッチ・コンパイルの動作方法と同じようにするには、 215 ページの『付録 D. バッチ処理のサンプル』に示されているようなプログラムを使用します。

アセンブラーからのコンパイラーの起動

新しいコンパイラーは、アセンブラーから IEL0AA を呼び出して起動することはできません。

新しい DD オプションを使用することで、使用するコンパイラーの代替 DD 名のリストを指定できます。 これにより、旧版のコンパイラーをアセンブラーから起動することで提供される主要な機能を利用でき、コンパイラーをアセンブラーから呼び出す必要性は減るはずです。

また、コンパイラーは、Enterprise PL/I プログラムから SYSTEM 組み込み関数を使用しても起動できることにも注意してください。

ただし、コンパイラーをアセンブラーから起動しなければならない場合は、アセンブラー・コードが以下の要件を満たしていれば、起動することができます。

- アセンブラー・コードで LE が使用可能になっている。
- CEEFETCH を使用して IBMZPLI をロードしている。
- IBMZPLI を呼び出すとき、レジスター 1 がコンパイル時に受け渡すオプションを含む可変長ストリングのアドレスを指している。

TSO 環境でのコンパイル

TSO 環境でのコンパイルはサポートされていません。

つまり、ISPF 4.5 オプションは、Enterprise PL/I では使用できません。このオプションを無効にして、別の目的で使用しなければならないと考えられます。

ただし、z/OS UNIX 環境では、**pli** コマンドを使用することで、コンパイラーを起動できます。コンパイラーの z/OS UNIX 環境での使用について詳しくは、「*Enterprise PL/I for z/OS* プログラミング・ガイド」を参照してください。

INCLUDE データ・セット名の指定

%INCLUDE ステートメントに対応する DD ステートメントには、インクルードするファイルを含む PDS (または PDSE) の名前を指定しますが、メンバー・ファイルの名前を指定してはなりません。

例えば、TEST DD ステートメントを使用してデータ・セット INCLUDE.PLI からファイル DEBUG をインクルードする場合、次の %INCLUDE ステートメントと、対応する DD ステートメントを使用できます。

%INCLUDE ステートメント	**DD** ステートメント
`%INCLUDE TEST(DEBUG);`	`TEST  DD DISP=SHR,DSN=INCLUDE.PLI`

新しいコンパイラーでは、次の DD ステートメントは受け入れません。

```
TEST  DD DISP=SHR,DSN=INCLUDE.PLI(DEBUG)
```

SYSLIN データ・セットの指定

コンパイラーからの 1 つ以上のオブジェクト・モジュールの形式での出力は、OBJECT コンパイル時オプションを指定した場合は SYSLIN データ・セットに保管されます。 このデータ・セットは DD ステートメントで定義されます。

SYSLIN DD では、PDS および PDSE のいずれでもなく、順次データ・セットを指定する必要があります。

コンパイル時の時間およびスペース所要量

コンパイラー SYSPRINT データ・セットの LRECL は 137 です。新しいコンパイラーは、コードを生成する際に、旧版よりかなり多くの時間を必要とし、かなり多くのストレージを使用する可能性があります。

このことは、OPT(2) または OPT(3) を指定した場合に特に当てはまります。これを指定した場合、コンパイルによっては 100M を超える領域を必要とし、またコンパイルに数分を要する可能性があります。オプション DFT(REORDER) を指定せずにオプション OPT(2) または OPT(3) を使用すると、この問題が発生する可能性が高いため、この使用法は避ける必要があります。

領域サイズがコンパイルを実行するには小さすぎると、コンパイルは多くの場合、次のメッセージを表示して終了します。

```
IBM1936I S Invocation of compiler backend ended abnormally.
```

また、この状態では、SYSOUT にコンパイラー・バックエンドからの次のメッセージが出力されます。

```
SEVERE ERROR IBM5002: Virtual storage exceeded.
```

メッセージがこの組み合わせで表示された場合は、プログラムをいくつかの小さなプログラムに分割するか、または領域サイズを大きくしてから再コンパイルする必要があります。

新しいコンパイラーは、必ず ALL31(ON) を指定して、さらには 16MB ラインより上から取得された HEAP と STACK を使用して実行します。

AMODE(24) の制約事項

AMODE(31) と RMODE(ANY) は、Enterprise PL/I アプリケーションのデフォルト設定です。アプリケーションを AMODE(24) で実行するには、いくつかのタスクを完了する必要があります。

1. すべての PL/I ソースを、コンパイラー・オプションに NORENT を指定してコンパイルする。
2. SCEELKED データ・セットの前に連結した SIBMAM24 データ・セットとリンクする。
3. 言語環境プログラムのランタイム・オプション ALL31(OFF) および STACK=(,,BELOW,,,) を指定して実行する。

注:

1. (Enterprise PL/I および旧版の PL/I の両方が関係するアプリケーションを含め) ILC アプリケーションでは、AMODE(24) はサポートされていません。 この制約事項に関する唯一の例外は、Enterprise PL/I とサポートされている高水準アセンブラーのリリース間における ILC です。
2. バインダーの SYSLIB 連結内に SIBMAM24 ライブラリーが含まれる場合には、モード切り替え機能のあるライブラリー・モジュールを利用できます。 しかし SIBMAM24 ライブラリーが含まれているだけで、結果のロード・モジュールが自動的に AMODE(24) になるわけではありません。
3. バインダーの SYSLIB 連結内において、SCEELKED データ・セットの前に SIBMAM24 ライブラリーにリンクすることなく、AMODE(24) で Enterprise PL/I プログラムを実行しようとすると、アプリケーションは無効になり、不明瞭な形の異常終了になります。 例えば、最初のブロックから抜ける GOTO はほとんどの場合、ライブラリー SETJMP ルーチン内で異常終了します。

EXTERNAL 名の制限

名前が下記のいずれかで始まる変数は、EXTERNAL として宣言しないでください (PLIXOPT または PLITDLI などの IBM 提供の関数の名前を除く)。

- @@
- CEE
- IBM
- PLI

新しいコンパイラーのコード生成プログラムは、一部のタスクを実行する際、特に OPT(0) を指定した場合に、C 関数を使用します。 そのため、C 関数を直接呼び出す場合を除き、次のいずれかの名前を持つ変数を EXTERNAL として宣言することは避けてください。

- LONGJMP

- MEMCCPY
- MEMCHR
- MEMCMP
- MEMCPY
- MEMMOVE
- MEMSET
- SETJMP
- STRLEN
- SYSTEM

PLIXHD 変数は、ストレージ報告書の見出しとして使用されなくなりました。 その結果、PLIXHD という ID は予約済みでなくなったため、他の変数を宣言および使用する場合と同様に、この ID を宣言および使用できます (この ID を EXTERNAL として宣言していない場合に限る)。

リスト表示の相違点

新しいコンパイラーでは、旧版コンパイラーの作成するリストとは大きく異なるリストを作成します。

以下のように、いくつかの相違点があります。

- リスト表示の LRECL は 137 です。
- ソースの第 1 行は先頭ページの第 1 行に反映されませんが、その行の最初の 43 文字 (DBCS オプションが有効な場合は最初の 25 文字) は、それ以降のページのヘッダー行に取り込まれます (疑似アセンブラーのリスト表示の一部を除きます)。

制御ブロックの相違点

新しいコンパイラーは、生成されたコード内で、従来のコンパイラーによって使用されるものとは若干異なる内部制御ブロックを使用します。

そうした制御ブロックのレイアウトや意義を把握している従来のコードがある場合には、そのようなコードはほとんど作動しない可能性が高いので、おそらく変更しなければなりません。 以下のような場合、こうした違いによってコード変更が必要になります。

- PL/I ラベル変数のレイアウトを把握していているアセンブラー・コードで、それをアセンブラーから PL/I コードに再びブランチするために使用する場合
- PL/I ファイル変数および関連するファイルの制御ブロックのレイアウトを把握しているアセンブラー・コードで、それをファイルの DCB を取得するために使用する場合

ISAM サポートの相違点

Enterprise PL/I コンパイラーは、ISAM データ・セットをサポートしていません。

第 11 章 新しいコンパイラーのオプションについて

このセクションでは、重要なコンパイラー・オプションをいくつか説明し、重要なデフォルトをいくつか説明した後、互換性、パフォーマンス、品質、およびテストを向上させるための選択について説明します。

このセクションでのすべての説明を無視し、互換性のみを向上させて最大化する場合には、以下の推奨に従ってください。

1. 次のデフォルト・オプションを使用する。
   - BACKREG(5)
   - BIFPREC(15)
   - CMPAT(V2)
   - EXTRN(FULL)
   - LIMITS(EXTNAME(7))
   - NORENT
2. 次のデフォルトでない追加オプションを指定する。
   - COMMON
   - DFT(NOBIN1ARG)
   - DFT(LINKAGE(SYSTEM))
   - DFT(OVERLAP)
   - NOREDUCE
   - NORESEXP
   - RULES(LAXCTL)
   - RULES(NOLAXINOUT NOLAXSEMI)
   - STATIC(FULL)
   - NOWRITABLE(PRV)

このセクションの残りの部分では、行った選択の結果を理解できるように、これらおよびその他のオプションについて詳細に説明します。

また、コンパイラーのインストール時にジョブ IBMZWIOP を実行するか、コンパイラーのインストール後にモジュール IBMZIOP に usermod を適用すると、コンパイラー・オプションの IBM デフォルトを変更することができます。

互換性におけるデフォルト・オプションの効果について

このセクションでは、コンパイラー・オプションのいくつかのデフォルト設定と、それらを使用する必要性について説明します。

BACKREG(5)

BACKREG オプションは、逆チェーン・レジスターを制御し、このレジスターは、ネストされたルーチンが呼び出されたときに、親ルーチンの自動ストレージのアドレスを受け渡すのに使用されます。

PL/I for MVS & VM、OS PL/I V2R3、およびそれ以前のコンパイラーとの最適な互換性を実現するには、BACKREG(5) を使用する必要があります。

© Copyright IBM Corp. 1999, 2016

ENTRY VARIABLE を共用するルーチンはすべて同じ BACKREG オプションでコンパイルしなければなりません。また、アプリケーションの中のコードはすべて同じ BACKREG オプションでコンパイルすることを強くお勧めします。

VisualAge PL/I でコンパイルしたコードは、事実上 BACKREG(11) オプションを使用したものであることに注意してください。 Enterprise PL/I V3R1 または V3R2 でコンパイルしたコードも、デフォルトで BACKREG(11) オプションを使用しています。

BIFPREC(15)

BIFPREC オプションは、さまざまな組み込み関数によって戻された FIXED BIN の結果の精度を制御します。

BIFPREC は、以下の組み込み関数に影響します。

- COUNT
- INDEX
- LENGTH
- LINENO
- ONCOUNT
- PAGENO
- SEARCH
- SEARCHR
- SIGN
- VERIFY
- VERIFYR

BIFPREC コンパイラー・オプションの影響が最も明らかに見えるのは、上記の組み込み関数の結果の 1 つが、パラメーター・リストなしに宣言された外部関数に受け渡されるときです。例えば、次のような部分コードがあるとします。

```
dcl parm char(40) var;
dcl funky ext entry( pointer, fixed bin(15) );
dcl beans ext entry;
call beans( addr(parm), verify(parm),' ') );
```

関数 beans が実際に、そのパラメーターを POINTER および FIXED BIN(15) として宣言しているとします。上のコードがオプション BIFPREC(31) でコンパイルされ、z/OS のようなビッグ・エンディアン・システム上で実行される場合、コンパイラーは 2 番目の引数として 4 バイトの整数を受け渡し、2 番目のパラメーターはゼロであるかのように見えます。

関数 funky は、すべてのシステム上でどちらのオプションでも機能することに注意してください。

BIFPREC オプションは、組み込み関数 DIM、HBOUND、および LBOUND には影響しません。 CMPAT オプションは、次の 3 つの関数によって戻された FIXED BIN の結果の精度を判別します。CMPAT(V1) では、これらの配列処理関数は、FIXED BIN(15) の結果を戻し、CMPAT(V2) および CMPAT(LE) では、FIXED BIN(31) の結果を戻します。

関連情報:

『CMPAT(V2)』
Enterprise PL/I V3R2 以降、CMPAT(V2) がデフォルトです。 このデフォルトで、マイグレーションが容易になります。

CMPAT(V2)

Enterprise PL/I V3R2 以降、CMPAT(V2) がデフォルトです。このデフォルトで、マイグレーションが容易になります。

CMPAT(V2) には以下の効果があります。

- すべての記述子が、OS PL/I V2R3 コンパイラーおよび PL/I for MVS & VM コンパイラーで生成された記述子と同一になる。
- ストリングを戻す関数は (旧版のコンパイラーの場合と同様に)、戻り値にストリング・ロケーター記述子も使用する。

CMPAT(V2) は、Enterprise PL/I の新機能を使用できないようにするわけではありません。ただし、PL/I 記述子を検査または作成するアセンブラー・コードがある場合 (ストリングのみが対象であっても)、CMPAT(V2) オプションを使用する必要があります。例えば、DB2 にはそのようなアセンブラー・コードが含まれており、それを使って PL/I ストアード・プロシージャーを呼び出しているため、PL/I で記述したストアード・プロシージャーは、CMPAT(V2) を指定してコンパイルする必要があります。

注**:** Enterprise PL/I V5R1 の時点で、CMPAT(V1) オプションと CMPAT(LE) オプションはサポートされなくなりました。

EXTRN(FULL)

デフォルトでは、Enterprise PL/I コンパイラーは使用されていない EXTERNAL ENTRY を破棄することはしません。

このことは、削除されたエントリーの EXTRN が、リンカーに他の参照を解決させるために使用される場合に、問題を引き起こします。 例えば、プログラムが A というプロシージャーの内部で 2 次エントリー・ポイント B を呼び出したが、A の宣言は含んでいたものの A 自身への参照は含んでいなかった場合に、問題が発生します。

このオプションにより、宣言されたすべての外部 ENTRY に対して EXTRN が発行されることに注意してください。 ファイルをインクルードするときに、コードが使用する可能性がある宣言をすべて含めると、テキスト・デックに大量の EXTRN で取り込まれることになります。

LIMITS(EXTNAME(7))

Enterprise PL/I V3R2 以降、LIMITS(EXTNAME(7)) がデフォルトとなっています (以前は LIMITS(EXTNAME(100)) がデフォルトでした)。

このデフォルトは、新しいコンパイラーでのデフォルトを旧版のコンパイラーの通常の仕様に合わせることにより、マイグレーションを容易にします。旧版のコンパイラーでは、外部名は 7 文字に制限されており、より多くの文字数に対応するオプションも存在しませんでした。

また、LIMITS(EXTNAME(*n*)) で *n* > 8 の場合は、プリリンカーを使用するか、またはモジュールを PDSE に格納する必要があることに注意してください。

さらに、LIMITS(EXTNAME(7)) を指定した場合 (およびすべての旧版のコンパイラーの場合)、8 文字の名前を EXTERNAL として宣言すると、コンパイラーは最初の 4 文字と最後の 3 文字を使用して 7 文字の名前を作成し、この名前をリンカーに渡します。 それに対し、LIMITS(EXTNAME(8)) を指定した場合には、完全な 8 文字の名前がリンカーに渡されるため、旧版のコンパイラーで生成したコードとの間で非互換性が生じます。

例えば、名前 DEZEMBER を EXTERNAL として宣言すると、LIMITS(EXTNAME(7)) の場合には、リンカーには DEZEBER という名前が渡されますが、LIMITS(EXTNAME(8)) の場合には、DEZEMBER という名前が渡されます。

したがって、互換性を維持するために、LIMITS(EXTNAME(8)) は使用せず、デフォルトの LIMITS(EXTNAME(7)) を使用してください。

最後に、LIMITS(EXTNAME(7)) は PL/I 名にのみ適用されることに注意してください。アセンブラーや COBOL のルーチンには、8 文字の名前を指定できます (旧版コンパイラーの場合とまったく同じ)。

NORENT および WRITABLE

Enterprise PL/I V3R2 以降、NORENT がデフォルトです。それ以前のリリースでは、RENT がデフォルトでした。 WRITABLE オプションも、NORENT と組み合わせることで最高のパフォーマンスを提供するため、デフォルトになりました。

これにより、新しいコンパイラーは、旧版のコンパイラーと同様に、静的変数を書き込み可能にし、なおかつ REENTRANT にするための追加のコードを生成する (これが RENT オプションの動作です) ことをデフォルトでは行わなくなるため、マイグレーションが容易になります。

また、RENT オプションを使用する場合は、プリリンカーを使用するか、モジュールを PDSE に格納する必要があることに注意してください。

しかし、NORENT オプションを使用している場合、以下の条件がどちらも真であれば、NOWRITABLE オプションも使用する必要があります。

1. コードは REENTRANT である必要がある。
2. コードは CONTROLLED 変数または FILE を使用している。

Enterprise PL/I V3R4 以降、NOWRITABLE オプションに以下の 2 つのサブオプションがあり、それによって、コードの互換性が高くなる (低くなる) 可能性もあります。

FWS NOWRITABLE(FWS) オプションを使用すれば、ご使用のコードを、NOWRITABLE オプションを使用して Enterprise PL/I の以前のリリースで生成されたコードと互換性を持たせることができますが、Enterprise PL/I で生成されたコードと、PL/I for MVS & VM およびそれ以前のコンパイラーで生成されたコードの間で CONTROLLED 変数を共用させることは許されません。

PRV NOWRITABLE(PRV) オプションを使用すれば、Enterprise PL/I でコンパイルされたコードと旧 PL/I コンパイラーでコンパイルされたコードで CONTROLLED 変数を共用することが許されます。ただし、FETCH で CONTROLLED を使用している場合にこれらのコンパイラーによって課せられるのと同じ制限も課せられることになります。

SYSTEM

SYSTEM オプションは、通常、パラメーターが MAIN に渡される方法のみを制御します。 デフォルトは SYSTEM(MVS) です。

このオプションは、以下の場合を除き、すべてのプログラムに使用します。

SYSTEM(CICS)

SYSTEM(CICS) オプションは、すべての CICS MAIN プログラムに対して使用する必要があります。

SYSTEM(IMS)

SYSTEM(IMS) オプションは、IMS がパラメーターを値 (BYVALUE) で渡す IMS MAIN プログラムに対してのみ使用する必要があります。

SYSTEM(OS)

SYSTEM(OS) オプションは、z/OS UNIX が作成したパラメーター・リストを受け取る必要のある z/OS UNIX MAIN プログラムに対してのみ使用する必要があります。このオプションについての詳細は、「*Enterprise PL/I for z/OS* プログラミング・ガイド」を参照してください。

互換性をさらに向上させるためのデフォルト以外のオプションの選択

このセクションでは、旧版のコンパイラーと新しいコンパイラーとの互換性を向上させるために選択できる、いくつかのオプションについて説明します。

COMMON

COMMON オプションは、初期化されていない EXTERNAL STATIC に対し、コンパイラーが CM リンケージ・レコードを生成することを指定します。

このオプションにより、複数のルーチンで EXTERNAL STATIC 変数を宣言しているが、その初期化は 1 つのルーチンで行っているという場合に、移行をより容易に行うことができます。

ただし、このオプションは NORENT と LIMITS(EXTNAME(7)) の両方のオプションが指定されている場合にのみ有効であることに注意してください。

DFT(NOBIN1ARG)

DFT(NOBIN1ARG) コンパイラー・オプションは、通常コードの互換性を向上させますが、一部の新しい関数の使用が制限されます。

このオプションについて詳しくは、 130 ページの『1 バイトの FIXED BIN』を参照してください。

DEFAULT(LINKAGE(SYSTEM))

DFT(LINKAGE(SYSTEM)) を指定すると、パラメーター・リストが、旧版のコンパイラーで作成された場合と同じ方法で作成されます (最後のパラメーターのアドレスの高位ビットをオンにすることも含む)。

これは、C または JAVA で使用されるデフォルトのリンケージではありません。これらのデフォルトのリンケージは、新しいコンパイラーのデフォルトである DFT(LINKAGE(OPTLINK)) を指定することで得られます。 OPTLINK リンケージでは、最後のパラメーターはアドレスでない可能性があり (例えば BYVALUE FIXED BIN(31) の場合)、アドレスであった場合でも、高位ビットはオンにはなりません。さらに、OPTLINK リンケージでは、戻り値がある場合、この戻り値はレジスター 15 に戻される場合があります。

SYSTEM リンケージは、OPTIONS(COBOL) または OPTIONS(ASM) ルーチンに対して想定されます。

1 つの PL/I ルーチンが別のルーチンを呼び出す場合には、それらのルーチンが一致している限り、それらがどのリンケージを使用しているかは問題になりません。しかし、一部の非 PL/I ルーチンは、OPTIONS(ASM) として宣言されませんが、SYSTEM リンケージを使用します。 そこで、互換性とマイグレーションを最も簡単に実現するためには、DFT(LINKAGE(SYSTEM)) オプションを使用します。

ただし、SYSTEM リンケージをデフォルトにする場合は、そのリンケージを使用するすべての関数 (例えば C ライブラリー関数の `fread`) の宣言に、OPTIONS(LINKAGE(OPTLINK)) を追加する必要があります。例えば、`fread` は次のように宣言します。

```
dcl fread ext entry(...) options( linkage(optlink) );
```

DFT(OVERLAP)

DFT(OVERLAP) コンパイラー・オプションは通常、コードの互換性を向上させますが、パフォーマンスが若干低下します。

このオプションについて詳しくは、 121 ページの『代入元と代入先の重複』を参照してください。

NOREDUCE

NOREDUCE コンパイラー・オプションは、コードの互換性をわずかに向上させますが、パフォーマンスは大きく低下します。

このオプションについて詳しくは、 90 ページの『REDUCE』を参照してください。

NORESEXP

NORESEXP コンパイラー・オプションは、コードにより故意に ZERODIVIDE 条件を発生させる場合の、コードの互換性を高めます。

RESEXP コンパイラー・オプションを使用すると、コンパイラーがすべての制限された式をコンパイル時に評価することができます。 例えば、次のコードを持つプログラムは、コンパイル時に S レベルのメッセージを発行してコンパイルに失敗します。

```
if somevariable = goodvalue then;
    else
        put skip list( 1 / 0 );
```

NORESEXP コンパイラー・オプションが有効な場合、コンパイラーはこのステートメントにフラグを立てず、ZERODIVIDE 条件は、元の意図通りに実行時に発生します。

RULES(LAXCTL)

RULES(LAXCTL) コンパイラー・オプションは、コードの互換性をわずかに向上させますが、パフォーマンスは大きく低下します。

このオプションについて詳しくは、 91 ページの『RULES(NOLAXCTL)』を参照してください。

RULES(NOLAXINOUT NOLAXSEMI)

これらの RULES オプションのサブオプションは、生成されたコードを変更しないので、オブジェクトの互換性には効果がありません。

ただし、これらのサブオプションを指定した場合、新コンパイラーは、旧コンパイラーに似た動作をするようになります。これは、新コンパイラーはその時、旧コンパイラーが出していたような 2 つのメッセージを出すからです。特に、コンパイラーは、以下の条件において W レベル・メッセージを出します。

RULES(NOLAXINOUT)
: 初期化されていない可能性のあるスカラーが ASSIGNABLE BYADDR パラメーターとして渡されたことを検出した場合

RULES(NOLAXSEMI)
: コメント内部にセミコロンを検出した場合

NOWRITABLE

旧版のモジュールとの互換性を最大限に高めるには、NORENT オプションを選択する必要があります。

NORENT WRITABLE オプションを使用すると、コンパイラーは次のように静的ポインターを使用できます。

- CONTROLLED 変数を追跡するスタック用の基数として使用する
- FILE を表すストレージ用のハンドルとして使用する

NOWRITABLE オプションを指定すると、コンパイラーは静的ポインターを上記のいずれの目的にも使用しませんが、同じ機能を提供するためには、より多くのコード行を生成しなければならなくなります。

しかし、以下の条件がどちらも真である場合には、NOWRITABLE オプションを使用する必要があります。

1. コードは REENTRANT である必要がある。
2. コードは CONTROLLED 変数または FILE を使用している。

ただし、NOWRITABLE(FWS) オプションはパフォーマンスに対して潜在的に非常に強い悪影響を与える可能性があるため、上記のいずれかの項目が当てはまらない場合は、このオプションを使用しないでください。

パフォーマンスを向上させるためのオプションの選択

このセクションでは、コンパイラーのコード生成時のパフォーマンスを向上させるために選択できる、いくつかのオプションについて説明します。

このセクションでのすべての説明を無視し、コストに関係なく、とにかくパフォーマンス向上のみを試みる場合には、以下の推奨に従ってください。

1. 次のデフォルト・オプションを使用する。
   - REDUCE
   - NORENT
   - RULES(NOLAXCTL)
2. 次のデフォルトでない追加オプションを指定する。
   - ARCH(10)
   - BIFPREC(31)
   - DFT(NONASGN)
   - DFT(CONNECTED)
   - DFT(REORDER)
   - DFT(NOOVERLAP)
   - OPT(3)

ただし上記のすべてのオプションを使用しないことを選択する場合があるという考慮事項 (このセクションで説明) があるため、DFT(REORDER) と少なくとも OPT(2) の両方を使用しない限り、生成されたコードで良いパフォーマンスが得られることはありません。

ARCH

Enterprise PL/I V5R1 におけるデフォルトは ARCH(8) です。

Enterprise PL/I には、言語環境プログラム 1.13 (またはそれ以降) が必要です。また、言語環境プログラム 1.13 には、ARCH(5) をサポートするマシンが必要です。

コードを実行する最低レベルのマシンよりも高い ARCH レベルを指定した場合、コードはそれらのマシンで仕様例外によりプログラム異常終了を起こすおそれがあります。

8 より小さい ARCH 値を指定した場合、コンパイラーはその値を 8 に設定し直します。

BIFPREC(31)

BIFPREC(31) を指定すると、それが適用される組み込み関数を使用した場合に、コードのパフォーマンスが向上します。

ただし、BIFPREC(15) オプションは、非プロトタイプの ENTRY 宣言を使用した場合に互換性が向上します。

関連情報:

82 ページの『BIFPREC(15)』
BIFPREC オプションは、さまざまな組み込み関数によって戻された FIXED BIN の結果の精度を制御します。

DEFAULT(NONASGN)

オプション DFT(NONASGN) は、ASSIGNABLE として明示的に宣言されていないすべての STATIC 変数に、NONASSIGNABLE 属性を追加します。

STATIC 変数が実際に変更されない場合には、このオプションを指定すると、コンパイラーはそれらの変数を読み取り専用ストレージに配置できるため、より良いパフォーマンスを得ることができます (特に RENT オプションを指定した場合)。

DEFAULT(CONNECTED)

DEFAULT(CONNECTED) オプションは、パラメーターが連結されるか連結されないかデフォルトを設定します。

非連結配列とは、エレメントがストレージの隣接する部分に配置されていない配列のことです。 非連結配列は、次に示す両方の呼び出しによって渡されます。

```
dcl a(3,4) fixed bin;

dcl 1 x(5), 2 y fixed bin, 2 z fixed bin;

call f( a(*,1) );

call f( x.y );
```

新旧いずれのコンパイラーでも、非連結配列は完全にサポートされており、実際には、コンパイラーはどの配列パラメーターも連結されていないこと、つまり連続した複数の配列エレメントの間に別のバイトが含まれている可能性があることを想定しています。

この想定のため、コンパイラーはスローダウンし、またより多くのコードを生成しなければならないため、アプリケーションもスローダウンします。

新しい DFT(CONNECTED) コンパイラー・オプションを使用すると、コンパイラーは、受け取ったすべての配列が連結されていることを想定し、より良いコードを生成します。 そのため、配列の分断されたスライス (例えばカラム) を渡すことが決してない場合は、このオプションを指定することで、より良いパフォーマンスを得ることができます。

DEFAULT(REORDER)

コンパイラー生成コードからのパフォーマンスを最適化するには、OPTIMIZE(2) または OPTIMIZE(3) を DFT(REORDER) と一緒に使用してください。

OPTIMIZE(2) または OPTIMIZE(3) を DFT(REORDER) ではなく DFT(ORDER) と一緒に使用すると、実行時のパフォーマンスは最適ではなくなり、コンパイル時間はより長くなる場合があります。

DEFAULT(NOOVERLAP)

互換性のために DFT(OVERLAP) オプションの使用を検討されるかもしれませんが、DFT(NOOVERLAP) オプションを使用すると、はるかに優れたパフォーマンスを得ることができます。

このオプションについて詳しくは、 121 ページの『代入元と代入先の重複』を参照してください。

OPTIMIZE(2)/OPTIMIZE(3)

コンパイラー生成コードからのパフォーマンスを最適化するには、OPTIMIZE(2) または OPTIMIZE(3) を DFT(REORDER) と一緒に使用してください。

OPTIMIZE(2) または OPTIMIZE(3) を DFT(REORDER) ではなく DFT(ORDER) と一緒に使用すると、実行時のパフォーマンスは最適ではなくなり、コンパイル時間はより長くなる場合があります。

OPT(3) は OPT(2) よりも良いコードを生成しますが、コンパイラーでは、OPT(2) よりも OPT(3) の方が、プログラムをコンパイルする時間が長くなります (とくに大きなプログラムの場合)。 この理由で、コンパイラーは OPT(TIME) を OPT(2) にマップします。

REDUCE

REDUCE オプションを指定した場合、コンパイラーは、埋め込みバイトを上書きすることになる場合でも、構造体へのヌル・ストリングの代入をより少なく単純な操作に軽減することができます。

REDUCE オプションを使用すると、ヌル・ストリングを構造体に割り当てるために生成されるコードの行数が少なくなり、その結果として通常はコンパイルが高速になり、コードの実行速度が大きく向上します。 しかし、埋め込みバイトはゼロにリセットされることがあります。

例えば次の構造体では、`field11` と `field12` の間に 1 バイトの埋め込みがあります。

```
dcl
1 sample ext,
  5 field10 bin fixed(31),
  5 field11 dec fixed(13),
  5 field12 bin fixed(31),
  5 field13 bin fixed(31),
  5 field14 bit(32),
  5 field15 bin fixed(31),
  5 field16 bit(32),
  5 field17 bin fixed(31);
```

ここで、代入 `sample = '';` について考えてみましょう。

NOREDUCE オプションを指定した場合、8 つの割り当てが生成されますが、埋め込みバイトは変更されません。

しかし、REDUCE を指定した場合、その割り当ては 3 演算に削減されます。

NOREDUCE を指定した場合、次のようなコードが生成されます。

```
00004C  5810  3056        00015 |      L     r1,=A(@CONSTANT_AREA)(,r3,86)
000050  58E0  305A        00015 |      L     r14,=A(SAMPLE)(,r3,90)
000054  4100  0000        00015 |      LA    r0,0
000058  D206  E004  1000  00015 |      MVC   FIELD11(7,r14,4),+CONSTANT_AREA(r1,0)
00005E  5000  E000        00015 |      ST    r0,<s9:d0:l4>(,r14,0)
000062  5000  E00C        00015 |      ST    r0,<s9:d12:l4>(,r14,12)
000066  5000  E010        00015 |      ST    r0,<s9:d16:l4>(,r14,16)
00006A  5000  E014        00015 |      ST    r0,<s9:d20:l4>(,r14,20)
000072  5000  E018        00015 |      ST    r0,<s9:d24:l4>(,r14,24)
000076  5000  E01C        00015 |      ST    r0,<s9:d28:l4>(,r14,28)
00007A  5000  E020        00015 |      ST    r0,<s9:d32:l4>(,r14,32)
```

しかし、REDUCE を指定すると、次のようなコードが生成されます。

```
00004C  5810  3042        00015 |      L     r1,=A(SAMPLE)(,r3,66)
000050  58E0  3046        00000 |      L     r14,=A(@CONSTANT_AREA)(,r3,70)
000054  D703  1000  1000  00015 |      XC    _shadow1(4,r1,0),_shadow1(r1,0)
00005A  D206  1004  E000  00015 |      MVC   _shadow1(7,r1,4),+CONSTANT_AREA(r14,0)
000060  D717  100C  100C  00015 |      XC    _shadow1(24,r1,12),_shadow1(r1,12)
```

そのため、最も良いパフォーマンスを得るには、REDUCE コンパイラー・オプションを使用してください。

NORENT

NORENT オプションは、生成されたオブジェクト・コードの互換性を向上させるため、コンパイラーのデフォルトの 1 つになりましたが、このオプションは、NOWRITABLE オプションを同時に指定していない場合には、コードのパフォーマンスも大きく向上させます。

このパフォーマンス向上の理由は、RENT オプションを指定した場合には、すべてのロード・モジュールの初期化に要する時間が長くなることと、呼び出しおよび静的変数の参照に使用されるコードの長さが長くなることにあります。

ただし、コードが REENTRANT でなければならず、コードが CONTROLLED 変数または FILE を使用している場合は、RENT オプションか、または NORENT オプションと NOWRITABLE オプションの両方を使用する必要があることに注意してください。

NORENT で NOWRITABLE を使用し、アプリケーションが CONTROLLED 変数を使用している多くのプログラムからなる場合、NOWRITABLE(FWS) を使用するより、NOWRITABLE(PRV) を使用した方がパフォーマンスが向上します。ただし、NOWRITABLE(PRV) を使用すると、FETCH で CONTROLLED 変数を使用した場合の古い制限も課せられることになります。

RULES(NOLAXCTL)

RULES(LAXCTL) を指定すると、コンパイラーは、速度が大きく低下し、より時間のかかるコードをより多く生成する可能性があります。

ある大規模な顧客のプログラムの場合には、このオプションにより、コンパイル時間を 40%、ランタイムを 50% 削減しました。

このオプションを理解するために、次の宣言を考えてみましょう。

```
DCL
 01 VTAB(*) CTL,              /*                VALOREN-TABLE  */
  02  WA0102  CHAR(26),       /* MUTATIONSDATUM DB2-TIMESTAMP   */
```

```
02  WA0104I BIN FIXED(31),   /* PKEY AKTIONSNR-ID:            */
02  WA0104K CHAR(1),         /* PKEY VALOREN-KNZ:             */
02  WA0104V DEC FIXED(15,0), /* PKEY VALORENNR                */
02  WA0104L BIN FIXED(15),   /* PKEY VV_SEG_LFNR              */
02  WA0104A CHAR(4);         /* PKEY TERM_ID                  */
```

VTAB の境界は、明らかにコンパイル時には不明です。しかし、WA0104K の長さは本当に 1 でしょうか ? この構造体は通常、次に示す 2 つのステートメントのいずれかのようなステートメントによって割り振られます。

```
ALLOC VTAB( 100 );

ALLOC VTAB( N + M );
```

このいずれかの割り振りの後で、WA0104K の長さは 1 になります。

しかし、この構造体は次のように割り振られている可能性もあります。

```
ALLOC
  1 VTAB(17),
      2 WA0102,
      2 WA0140I,
      2 WA0104K CHAR(29);
```

この場合、WA0104K の長さは 29 になってしまいます。

コンパイラー・オプション RULES(LAXCTL) を指定すると、ストリングに対して最初に宣言された長さが定数であったにもかかわらず、上記に示したような割り振りが許可されます。 ただし、このオプションを指定すると、コンパイラーははるかに長いコード・シーケンスを生成するようになります。

これとは対照的に、コンパイラー・オプション RULES(NOLAXCTL) は、定数として宣言されたすべての長さおよび境界が実際に定数であることを想定します。この想定に違反した ALLOCATE ステートメントには、S レベル・メッセージ IBM2063 でフラグが付けられます。

したがって、このオプションを使用すると、ランタイムに発生する問題が除去され、コンパイル時にもランタイムにも、はるかに優れたパフォーマンスが得られます。

品質を向上させるためのオプションの選択

このセクションでは、コードの品質を向上させるか、または確実なものにするために選択できるオプションのいくつかについて説明します。

RULES(NOLAXDCL)

RULES(LAXDCL) を指定すると、コンパイラーは、宣言されていない変数ごとに、I レベルのメッセージのみを発行します。 これに対し、RULES(NOLAXDCL) を指定すると、E レベルのメッセージが出力されます。

コードに十分な品質を与えようとするのであれば、常に RULES(NOLAXDCL) を指定してコンパイルする必要があります。

しかし、Windows 上でこのオプションをデフォルトにした際、非常に多くのユーザーから苦情が寄せられたため、現在ではデフォルトではありません。

RULES(NOLAXDCL) を指定して以下のコードをコンパイルすると、コンパイラーは *starring_role* が宣言されていないことを示す E レベルのメッセージを出します。 このメッセージは、この名前がほぼ確実にタイプミスであることを警告します。これは、このコンパイラー・オプションを使用する必要性を示す 1 つの例です。

```
 x: proc( starting_role ) returns( fixed bin(31) );
   dcl starting_role fixed bin(31);
   return( starring_role + 1 );
end;
```

オプション RULES(NOLAXDCL) は、「実際に動作する」コードにもフラグを立てる場合があります。

```
read_in = fileread( file_in, addr(buffer), stg(buffer) );

if read_in = 0 then
  leave;
```

read_in が宣言されていない場合でも、このコードは動作します。しかし、*read_in* は属性として FLOAT を持つため、これは好ましくありません。

RULES(NOLAXIF)

コンパイラー・オプション RULES(NOLAXIF) を指定すると、コンパイラーは、属性 BIT(1) NONVARYING を持たないすべての条件式に、E レベルのメッセージと共にフラグを立てます。

IF、WHILE、UNTIL、および優位でない WHEN 文節内の式は、属性 BIT(1) NONVARYING を持つ必要があります。しかし、新旧すべてのコンパイラーは、これらの文節ではどのような計算式でも許可します。 例えば、次のようなステートメントを書くことができます。

```
dcl x fixed bin(31);

if x then ...
```

この IF ステートメントの意図は、次に示すステートメントと同じ意味であるかもしれません。

```
if x ¬= 0 then
```

しかし、新旧のコンパイラーとも、このステートメントを次のように解釈します。

```
if abs(x) ¬= 0 then
```

このステートメントや、これに類似したステートメントは、条件式がブールになるようにコーディングしたほうが、はるかに優れています。

そのため、この RULES(NOLAXIF) は、この良いコーディング習慣を実践するために使用できます。

RULES(NOLAXIF) を指定すると、コンパイラーは、BIT(8) 変数、つまり Y への参照のみからなる IF 文節にフラグを立てます。この場合、生成されたコードは、8 ビットのうちのいずれかがオンであればこの式を真として扱いますが、この IF 文節を Y ^= ''b に変更したほうが良い場合があります。

RULES(NOLAXIF) を指定すると、コンパイラーは次の形式の代入にもフラグを立てます。

```
x = y = z
```

RULES(NOLAXIF) オプションは、フラグが立てられたどのステートメントに対して生成されたコードでも有効になることに注意してください。

RULES(NOLAXLINK)

オプション RULES(LAXLINK) を指定すると、代入または比較の際の 2 つの ENTRY 変数または定数の宣言で指定した LINKAGE およびその他のオプションを、コンパイラーは無視します。

例えば、RULES(LAXLINK) オプションを使用すると、次の間違ったプログラムにフラグは立てられません。このプログラムを実行すると、ほぼ確実に異常終了が生じることになります。

```
dcl funtion ext entry returns( char(20) );
dcl subrtn  entry variable;

subrtn = function;

call subrtn;
```

これらのエラーをキャッチし、基本的なコーディング規格を実践するには、RULES(NOLAXLINK) オプションを指定する必要があります。

しかし、CICS プリプロセッサーによって以下の宣言が生成されるので、EXEC CICS ステートメントが含まれるプログラムで RULES(NOLAXLINK) オプションを使用するのはたいていの場合良い考えとは言えません。

```
DCL DFHEI0   ENTRY VARIABLE INIT(DFHEI01) AUTO;
DCL DFHEI01  ENTRY OPTIONS(INTER ASSEMBLER);
```

その結果、変数 DFHEI0 は EXEC CICS ステートメントに対して CICS プリプロセッサーが生成するコード内で使用されるため、コンパイラーは RULES(NOLAXLINK) に基づいて、OPTIONS(INTER ASSEMBLER) を使用して宣言されたエントリー DFHEI01 にフラグを立てますが、そのエントリーは OPTIONS 属性を指定せずに宣言された DFHEI0 に代入されます。

RULES(NOLAXMARGINS)

RULES(NOLAXMARGINS) を指定すると、右マージンの後にブランクがない行にフラグが立てられます。これは、コード、特にコメント終了マーカーが、誤って右に大きく桁送りされてしまった場合に、問題を検出するために役立ちます。

しかし、多くのソース・ファイルがシリアル番号やその他のデータを右マージンの後に含んでいるため、デフォルトは RULES(LAXMARGINS) です。

RULES(LAXSTRZ)

新しいコンパイラーは、ソースが既知の長さを持ち、ターゲットが既知の最大長を持ち、ソース長がターゲットの最大長よりも大きい場合に、すべてのストリング代入にフラグを立てます。 残念ながら、これによってコンパイラーは、後続のビット

や文字が「重要ではない」代入でも、フラグを立ててしまいます。コンパイラー・オプション RULES(LAXSTRZ) は、この「ノイズ」を軽減するために役立ちます。

RULES(LAXSTRZ) のもとでは、以下の条件のいずれかが真である場合、初期文節または代入ではメッセージが発行されなくなります。

- ビット変数のソースが、長さが超過しているが、超過ビットがすべて 0 である場合。
- 文字変数のソースが、長さが超過しているが、超過文字がすべてブランクである場合。

そのため、RULES(LAXSTRZ) が指定されていると、次のうち 2 番目のステートメントのみにフラグを立てられます。

```
dcl a char(4) init('ok   ');
dcl b char(4) init('error');
```

デフォルト・オプションは RULES(NOLAXSTRZ) ですが、RULES(LAXSTRZ) を指定したほうが、本当に問題のある代入に焦点を当てることができるため、より良い品質が得られる可能性があります。

RULES(NOMULTICLOSE)

新旧のすべてのコンパイラーでは、複数の DO、SELECT、BEGIN、または PROCEDURE グループを 1 つの END ステートメントで閉じることが許可されます。ただし、新しいコンパイラーでは I レベルのメッセージが発行されます。しかし、複数のグループを 1 つの END ステートメントで閉じることは、良いプログラミング習慣ではありません。コンパイラー・オプション RULES(NOMULTICLOSE) を指定すると、コンパイラーはそのようなコードに対して E レベルのメッセージと共にフラグを立てます。

例えば、このオプションを指定した場合、コンパイラーは次のコードに対して反応します。

```
a: do i = 1 to 17;
   b: do j = 1 to 29;
      t = x(i,j);         /* transpose i and j
      x(i,j) = x(j,i);
      x(j,i) = t;
   end b;              /* end of loop */
end a;
```

最初のコメントが閉じられていないため、両方の DO ループが `end a;` で閉じることに注意してください。

テスト用のオプションの選択

このセクションでは、開発中にコードをテストする必要が生じた場合に選択できる、いくつかのオプションについて説明します。

CHECK(CONFORMANCE)

CHECK オプションの CONFORMANCE サブオプションを指定すると、コンパイラーは一部のプロシージャーのプロローグに追加のコードを生成して、渡されたパラメーターが、プロシージャーによって期待されているものであるかどうかを検査します。

このオプションが適用される時点、このオプションが行う機能、およびコードのテストを展開する場合にこのオプションが非常に便利なツールになりうることについての詳細は、「プログラミング・ガイド」を参照してください。

GONUMBER

コンパイラー・オプション GONUMBER を指定すると、コンパイラーは「ステートメント番号テーブル」を生成します。 このテーブルを使用すると、エラー・ハンドラーは、発生した条件についてメッセージを生成する必要が生じた際に、その条件が発生した場所を、それを収容しているプロシージャー内でのオフセットによってだけでなく、ソース・プログラム内での位置によっても識別することができます。

この追加情報は、プログラム内でのエラーを分析する際に、非常に役立つ場合があります。 このオプションを使用しない場合は、エントリーのオフセットからソース・ステートメントを判別するために使用できるテーブルをコンパイラーに生成させるために、OFFSET オプションを使用する必要があります。

PREFIX

PREFIX コンパイラー・オプションを指定すると、ソースを編集する必要なしに、PL/I の条件を使用可能にできます。

次の 3 つの条件は、テスト中に使用可能にしておくと特に便利です。

- SIZE
- STRINGRANGE
- STRINGSIZE
- SUBSCRIPTRANGE

ただし、これらのうちどの条件を指定した場合にも、コンパイラーはより多くのコードを生成するため、コード生成時のパフォーマンスが著しく悪化する場合があります。 コンパイル全体に対して SIZE 条件を使用可能にすると、パフォーマンスが特に悪化する可能性があるため、このオプションを実動プログラムで使用することはお勧めできません。

TEST

デバッグ・ツールを使用している場合は、コンパイラーがシンボル・テーブルや、デバッガーに必要なその他の情報を生成するように、TEST オプションを指定する必要があります。

ただし、このオプションは実動プログラムでは使用しないでください。

第 12 章 新しいコンパイラーのメッセージについて

新しいコンパイラーは、従来のコンパイラーで発行されていたメッセージと非常によく似た多くのメッセージを発行します。ただし、新しいコンパイラーは多くの新しいメッセージも発行し、これらの一部は、新しいコンパイラーにマイグレーションするにあたって非常に重要になることがあります。 それらのメッセージに注意することで、起こりうる移行上の問題に気が付くことができます。この章では、このようなメッセージのうち、重要度の高いいくつかのメッセージについて説明します。

この章で説明する多くのメッセージは I レベルと W レベルのメッセージですが、これらのメッセージを無視してもよいということではありません。 実際、これらのメッセージは、「作業」コードで発生する可能性のあるエラーを強調するものです。

IBM1044: 1 バイトの FIXED BIN

この I レベル・メッセージは、Enterprise PL/I と旧 PL/I コンパイラーの違いについてアラートするものです。新しいコンパイラーによって生成されるメッセージは、次のようになります。

```
IBM1044I I FIXED BINARY with precision 7 or less is mapped to 1 byte.
```

これは Enterprise PL/I の 1 つの機能であり、1 バイト整数をサポートします。これは、特にデータが C または JAVA で交換される場合に、非常に便利な機能です。

ただし、これも従来のコンパイラーと新しいコンパイラーで違いがあります。従来のコンパイラーでは、例えば FIXED BIN(7) と宣言された変数は 2 バイトで割り振られ、SIZE が使用可能になっていなければ、1 バイト整数で許される -128 から 127 というかなり小さい値の範囲ではなく、-32768 から 32767 の範囲の値であることが想定されます。

意図的にこの新しい機能を活用する場合でない限り、おそらく、EXIT オプションを使用してこのメッセージの重大度を増してから、このメッセージを生成するすべてのコードを変更すべきです。

IBM1053: スケール付き FIXED BIN の評価

コードの一部をコンパイルしているときに、次のメッセージが表示されることがあります。

```
IBM1053I I Scaled FIXED operation evaluated as FIXED DECIMAL.
```

このメッセージが生成されるコードの例およびその場合の対応方法の説明については、 133 ページの『スケール係数および FIXED BIN を持つ算術組み込み関数』を参照してください。

© Copyright IBM Corp. 1999, 2016

IBM1065: 不正確な浮動小数点定数

この I レベル・メッセージは、Enterprise PL/I に問題の原因があることをアラートするものです。

```
IBM1065I I Float constant ... would be more precise if specified as a long float.
```

浮動小数点定数は、2 進の小数部 (.1E0b や .001E0b など) でうまく表すことができますが、10 進の小数部 (.1E0 や 3.1415E0 など) では正確には表せません。このメッセージは、そのような小数部が長精度浮動小数点として (例えば、6 桁を超える小数桁数を指定することによって) 指定された場合、小数部がもっと正確に表されることをアラートするものです。

IBM1091: FIXED BIN 精度の警告

この W レベル・メッセージは、最善の場合でプログラミングの欠陥があること、最悪の場合は問題のソースをアラートするものです。 新しいコンパイラーによって生成されるメッセージは、次のようになります。

```
IBM1091I W FIXED BIN precision less than storage allows.
```

Enterprise PL/I コンパイラーは、7、15、31、または 63 以外の精度を指定して SIGNED FIXED BIN 変数が宣言されるか、あるいは、8、16、32、または 64 以外の精度を指定して UNSIGNED FIXED BIN 変数が宣言された場合に、このメッセージを生成します。コンパイラーは、BIN、ADD、DIVIDE などの組み込み関数の結果が FIXED BIN になるのにもかかわらず、上記のような精度を指定している場合にも、このメッセージを出します。

例えば、変数を FIXED BIN(5) として宣言している場合、コンパイラーはその宣言にフラグを立てるので、その宣言を意図された FIXED BIN(15) に変更する必要がおそらくあります。

IBM1099: 混合した FIXED

コードの一部をコンパイルしているときに、次のようなメッセージが表示されることがあります。

```
IBM1099I W  FIXED DEC(7,2) operand will be converted to FIXED
            BIN(25,7). Significant digits may be lost.
```

メッセージ内の属性は異なる場合がありますが、これと同じメッセージを生成するコードの一部の例を次に示します。

```
 DCL
    1 REC_OUT,
       03  AVAIL        FIXED BIN(31),
       03  TOTAL_SPARE  FIXED DECIMAL(7,2),
       03  WORK_TOTAL   FIXED DECIMAL(7,2);

  AVAIL = 17;
  WORK_TOTAL = 12.2;

  TOTAL_SPARE = AVAIL + WORK_TOTAL;
```

新旧のコンパイラーは最後の代入をまったく同じ方法で実行し、どちらも、TOTAL_SPARE に 29.19 という値を代入します (予測される 29.20 という値ではな

く)。 ただし、新しいコンパイラーのみが、このステートメントをさらに詳しく調べることを促すメッセージを発行します。

このメッセージの意味、および一見間違いに思える上記のステートメントの結果が正しい理由を理解するには、指数演算以外の算術演算に適用される次の PL/I 規則をもう一度確認する必要があります。

1. どちらかのオペランドが FLOAT の場合、FIXED は FLOAT に変換される。
2. どちらかのオペランドが BINARY の場合、DECIMAL は BINARY に変換される。
3. DECIMAL(p,q) は、BINARY(1+log(10)*p, log(10)*q) に変換される。

したがって、FIXED BIN(31,0) 変数 *AVAIL* を FIXED DEC(7,2) 変数 *WORK_TOTAL* に加算した場合は、上記の規則に従って、 DEC(7,2) オペランドが強制的に BIN(25,7) に変換されます。

しかし 12.20 は、BIN(25,7) として正確に表せないため、実際には次のようにして変換されます。

```
( bin(12.20,31,0) * 2**7 ) /  10**2
```

この結果、約 12.195 という値が得られます。

この値に 17 を加算して変換し戻すと 29.19 という値が得られます。

上記のコンパイラーの動作はすべて正しいですが、おそらくユーザーの希望に沿うものではないでしょう。 その場合は、DECIMAL 組み込み関数を BINARY オペランドに適用するか、または新しいコンパイラー・オプション RULES(ANS) を指定することにより、この演算を強制的に DECIMAL で実行することができます。

RULES(ANS) を使用した場合、スケールされた FIXED BIN は使用できず、変換規則は、次のように経験の少ないユーザーが予想するものに近くなります。

```
if both operands are FIXED, then

  if either has a non-zero scale, any BIN becomes DEC
```

したがって、BIN(31,0) を DEC(7,2) に加算した場合、BIN(31,0) は DEC(10,0) に変換され、どの値も失われることはありません。

上記と同じ考慮事項が、次のコード・フラグメントにも適用されます。

```
dcl a dec fixed(15,3) init(2500000);
dcl zero bin fixed(31) init(0);
if (a ¬= zero) then
   put skip edit('dec fixed ¬= Zero')(a);
else
   put skip edit('dec fixed  = Zero')(a);
```

DEC(15,3) オペランドは BIN(31,10) に変換されます。

しかし BIN(31,10) に保持できる最大値は、2**21 すなわち 2_097_152 です。

したがって、この変換は正常に実行することができず、SIZE 条件が有効になっている場合は、SIZE 条件が発生します。 SIZE 条件が有効になっていない場合は、このコードは間違っており、変換を実行するために生成された CVB 命令によって、ZERODIVIDE 条件が発生します。

この場合も、新しいコンパイラーは該当する次のメッセージを発行します。

```
IBM1099I W    FIXED DEC(15,3) operand will be converted
              to FIXED BIN(31,10). Significant digits may be lost.
```

最後に、この場合も、RULES(ANS) コンパイラー・オプションを使用するか、または DEC 組み込み関数を BINARY オペランドに適用することにより、このコードを修正することができます。

IBM1181: 誤ってコーディングされた DO ループ

今までは常に正常にコンパイルされてきたプログラムが、次のメッセージを出すようになることがあります。

```
IBM1181I W   A WHILE or UNTIL option at the end of a series DO specifications
             applies only to the last specification.
```

このメッセージは次のようなステートメントに対して発せられます。

```
DO I = 1, 2 WHILE( X = 'Z' );
```

このメッセージは、この DO ループが I = 1 で (X = 'Z' が真であるかどうかに関係なく) 一度実行され、X = 'Z' が真であれば、I = 2 で実行されることを示しています。この DO ステートメントは、(作成者が意図していたものだと考えられますが) このステートメントと同じではありません。

```
DO I = 1 WHILE( X = 'Z' ), 2 WHILE( X = 'Z' )
```

これが意図していたものであれば、次のようにステートメントをコーディングするのが最適であると考えられます。

```
DO I = 1 TO 2 WHILE( X = 'Z');
```

また、DO グループの 2 回目の反復の前にのみ、X = 'Z' であることをテストしたかった場合、次のようなステートメントをコーディングするのが最適であると考えられます。

```
DO I = 1 TO 2 UNTIL( X ^= 'Z' );
```

IBM1206: BIT 演算子の誤用

この W レベル・メッセージは、起こり得るコーディング・エラーについてアラートするものです。 新しいコンパイラーによって生成されるメッセージは、次のようになります。

```
  IBM1206I W BIT operators should be applied only to BIT operands.
```

このメッセージを出現させるステートメントに対して新しいコンパイラーが生成したコードは、従来のコンパイラーが生成したコードと同じです。ただし、従来のコンパイラーは警告メッセージを発行しませんでした。

このメッセージを生じさせ、その原因となりうるコーディング・エラーの例として、次のコードを見てみましょう。

```
dcl (x,y) fixed bin;

if x = ¬y then
  ...

if x ¬ y then
  ...
```

最初の IF ステートメントで、ビット接頭否定演算子は FIXED BIN 変数 *y* に適用されますが、おそらくそういう意図はなかったと思われます。 同様に、2 番目の IF ステートメントで、ビット 2 項排他的論理和演算子は FIXED BIN 変数 *x* および *y* に適用されますが、おそらくこの場合もそういう意図はなかったと思われます。 実際には、おそらく両方のステートメントにタイプミスがあり、変数 *x* および *y* が等しくないことを検査するはずでした。

また、ビット単位演算子を本当に意図する場合は、BIT 組み込み関数 (または INOT および IEOR 組み込み関数の可能性もある) を使用して、それを明確にすることがおそらく最善なことでしょう。

IBM1208: 完全には初期化されていない配列

コードの一部をコンパイルしているときに、次のメッセージが表示されることもあります。

```
IBM1208I W   INITIAL list for the array WPPXS_TAB
                   contains only one item.
```

このメッセージは、例えば、次の宣言内の変数がプログラムで使用されている場合に出力されます。

```
DCL WPPXS_TAB(15) CHAR(3500) INIT((15)' ');
```

INIT((15)' ') 属性は、1 つのブランクから成るストリングの 15 個のインスタンスは指定しません。 *15* はストリング反復因数であるため、この INIT 文節は、(15 個のブランクから成る) 1 つのストリングのみを指定します。

この配列全体をブランクに初期化するには、次のようにコーディングする必要があります。

```
DCL WPPXS_TAB(15) CHAR(3500) INIT( (*) ('') );
```

新しいコンパイラーは、例えば次のような他の多くの同様の宣言についてもこのメッセージを出力します。

```
DCL LISTE(4,60:73)   CHAR(50)  INIT('');
DCL SPRACH_TAB(4)    CHAR(15)  INIT('');
```

最後に、この配列が構造体の一部である場合は、コンパイラーはそれ以降にその構造体でこの問題が発生するたびに、その構造体に対し、メッセージ IBM2603 と共にフラグを立てます。 EXIT オプションを使用することで、この問題に対してフラグを立てる回数を 1 つの構造体につき 1 回に減らすことができます。

IBM1215: 不完全な宣言

従来のコードの一部をコンパイルしているときに、次のようなメッセージが表示されることもあります。

```
IBM1215I W  The variable I is declared without any data attributes.
```

新しいコンパイラーは、例えば次のような宣言についてこのメッセージを発行します。

```
DCL  I, J FIXED BIN;
```

従来のコンパイラーはこの宣言についてメッセージを出力しませんが、新しいコンパイラーは上記のメッセージを発行します。これは、この宣言は、DCL (I,J) FIXED BIN; と同じ意味ではなく、実際は DCL I; DCL J FIXED BIN; と同じ意味であるためです。

IBM1216: 間違った構造体宣言

同様に、次の宣言について見てみましょう。

```
DCL
  1 S,
    2 A    CHAR(10),
    2 B,
      2 C  CHAR(3),
      2 D  CHAR(3);
```

従来のコンパイラーは、この宣言についてメッセージを出力しません。しかし新しいコンパイラーは、次のメッセージを出力します。

```
IBM1216I W  The structure member B is declared without any data
                  attributes. A level number may be incorrect.
```

このメッセージは、宣言内の考えられるエラー、すなわち *C* と *D* はレベル 2 ではなくレベル 3 で宣言されるべきであることを示しています。しかし上記の宣言の場合、C と D は *B* と同じ構造体レベルであるため、*B* はこれらの親ではなく、B にはデフォルト属性の FLOAT が割り当てられます。 これは、ほぼ間違いなくユーザーの希望に反する処理であり、この新しいメッセージは、この発生する可能性の高い問題への注意を促しています。

コンパイラーは、次のコードに対してもこのメッセージを出します。

```
DCL PARDIASE CHAR (20);
DCL 1 INDIASE1 BASED (PTPDIASE),
     2 C1CODIA CHAR      (1),
     2 C1FECDI DEC FIXED (9),
     2 C1DIADI CHAR (9),
     2 C1ABRDI CHAR (3),
     2 C1RESDI;
DCL PTPDIASE POINTER;
PTPDIASE = ADDR (PARDIASE);
. . .
INDIASE1 = '';
```

このメッセージは、変数 C1RESDI がデータ属性を指定せずに宣言されているという事実に対してフラグを立てるものです。 このため、FLOAT DEC(6) というデフォルト属性を取得することになり、構造体 INDIASE1 が 22 バイトを占めることを意味します。 しかし、この構造体は CHAR(20) フィールドのアドレスが割り当て

られているポインターをベースにしているため、INDIASE1 = '';という代入は、他の変数が使用している 2 バイトのストレージをブランクにしてしまいます。このコードでは、これによってライブラリー・ルーチン内で異常終了が発生することになります。 C1RESDI が CHAR(2) として宣言されているか、あるいは少なくとも CHAR(0) として宣言されていれば (しかも、CHAR(0) が正しい PL/I であれば)、問題は発生しません。

そのため、このセクションで説明している他の多くのメッセージと同様、このメッセージは E レベル・メッセージではありませんが、コードを変更して、コンパイルでこのメッセージが出ないようにするのが最適な方法です。

IBM1220: 無意味な比較

コードの一部をコンパイルしているときに、次のメッセージが表示されることもあります。

```
IBM1220I W  Result of comparison is always constant
```

新しいコンパイラーは、例えば次のようなコードがあった場合にこのメッセージを出力します。

```
DCL ZWSTRING     CHAR(80);
DCL ZWSTRING2    CHAR(8);

ZWSTRING = 'E R R O R';
.....
IF ZWSTRING2 = 'E R R O R' THEN
```

このメッセージが出力される理由は、「E R R O R」は最後の文字がブランクではない CHAR(9) であるため、決して CHAR(8) フィールドに等しくなることはないからです。

このメッセージを生成するコードは、問題があるため詳しく調べる必要があります。 実際に、このメッセージで DO ループ・ステートメントが示されている場合に何も処置をとらなかった場合、コードで無限ループが発生する可能性があります。 コンパイラーは、例えば次のような 3 つの実行可能ステートメントのすべてについてこのメッセージを出力し、最後のステートメントでは、LEAVE ステートメントにより終了されない限りループが無限に実行されます。

```
DCL ZZ9 PIC'ZZ9';
DCL N  FIXED BIN(15);

IF ZZ9 < 0 THEN  ...
IF ZZ9 <= 999 THEN ...
DO N = 1 TO 32768;  ...; END;
```

なお、LEAVE (または GOTO) ステートメントにより終了されるまで「無限に」実行したいループがある場合は、このループ・ステートメントを DO FOREVER を使用してコーディングするのが最適な方法です。

IBM1927: SIZE 条件

「作業」コードの一部をコンパイルしているときに、次のようなメッセージが表示されることもあります。

```
IBM1927I S  SIZE condition raised by attempt to convert
            32777 to SIGNED FIXED BIN(15)
```

例えば、次のコードがこのメッセージを出力します。

```
DCL I   BIN FIXED(15);

DCL
  1 S,
    2 A    CHAR(10),
    2 B    CHAR(32767);

 I = STG(S);
```

上の代入について、以下の点に注意してください。

- 代入元の STG(S) は 32777 に等しい。
- 代入先の I には FIXED BIN(15) という属性が設定されている。

従来のコンパイラーであれば、何のメッセージも発行されません。

新しいコンパイラーでは、32777 は値が大きすぎて FIXED BIN(15) に変換できないことが通知されます (FIXED BIN(15) 変数の最大値は 32767 であるため)。

このメッセージは、無視できない問題を示しており、S レベルのメッセージであるため、コードを変更する必要があります。

IBM1948: 制限された式評価

コードの一部をコンパイルしているときに、次のメッセージが表示されることがあります。

```
IBM1948I S ZERODIVIDE condition with ONCODE=320 raised while
           evaluating restricted expression.
```

このメッセージが生成されるコードの例およびその場合の対応方法の説明については、 86 ページの『NORESEXP』を参照してください。

IBM2063: 無効な ALLOCATE

コードの一部をコンパイルしているときに、次のメッセージが表示されることがあります。

```
IBM2063I S Specification of extent for variable-name in
           ALLOCATE statement is invalid since it was declared
           with a constant extent.
```

このメッセージが生成されるコードの例およびその場合の対応方法の説明については、 91 ページの『RULES(NOLAXCTL)』を参照してください。

IBM2402: ストレージ・オーバーレイ

このメッセージは、重要なコーディング・エラーである可能性があることをアラートするものです。

```
IBM2402I E  <variable x> is declared as BASED on the ADDR of <variable y>,
            but <variable x> requires more storage than <variable y>.
```

このメッセージの重要度は、変数がプログラム内でどのように使用されているかによって異なります。 例えば、X が 100 バイトの構造体であり、Y が CHAR(200) BASED(ADDR(X)) として宣言されている場合、コンパイラーは「note that, in this example, the message is issued only when X is not subscripted.」というメッセージを出します。 プログラムに Y = '' というステートメントも含まれている場合は、重大な問題になります (代入により、他の目的でコンパイラーが使用している可能性のある 100 バイトのストレージが消されてしまうためです)。この種の問題は、必ず訂正しなければなりません。

ただし、プログラムでは、以下のようなステートメント内でのみ Y を使用している場合があります。

- SUBSTR(Y,1,STG(X)) = '';
- SUBSTR(Y,1,STG(X)) = LOW(STG(X));

この場合には、コードを変更する必要はありません。

ただしこの場合、このメッセージが出ないように Y を宣言することができます。Y を X の後に宣言した場合には、Y を CHAR(STG(X)) BASED(ADDR(X)) として宣言できます。 これにより、このメッセージは発生しなくなり、コードに変更を加える必要はありません。しかし、希望であれば、上記の代入ステートメントを以下のように単純化することもできます。

- Y = '';
- Y = LOW(STG(X));

IBM2409: 関数内での RETURN;

このメッセージは、おそらくコーディング・エラーであることをアラートするものです。

```
IBM2409I E  RETURN statement without an expression is invalid inside a
            subprocedure that specified the RETURNS attribute.
```

コンパイラーは、関数内で (例えば、RETURNS オプションを持つ PROCEDURE 内で) RETURN; ステートメントを検出したときにこのメッセージを出します。このステートメントが実行されると、この関数の呼び出し元は、関数の結果を使用した場合に、初期化されていない値を使用して、予測不能で不定の悪い結果になる可能性があります。

このメッセージを生成するコードは、訂正する必要があります。

IBM2410: 関数内に RETURN がない

このメッセージは、別のコーディング・エラーであることをアラートするものです。

```
IBM2410I E  Function F contains no valid RETURN statement.
```

コンパイラーは、関数内で (例えば、RETURNS オプションを持つ PROCEDURE 内で) RETURN ステートメントがないことを検出したときにこのメッセージを出し

ます。この関数が呼び出されると、この関数の呼び出し元は、関数の結果を使用した場合に、初期化されていない値を使用して、予測不能で不定の悪い結果になる可能性があります。

このメッセージを生成するコードは、訂正する必要があります。

IBM2412: RETURNS オプションの欠落

このメッセージは、関連するコーディング・エラーであることをアラートするものです。

```
IBM2412I E  Procedure has no RETURNS attribute, but contains a RETURN statement.
            A RETURNS attribute will be assumed.
```

これは、メッセージ IBM2409 がフラグを立てる問題とは逆の問題です。式を持つ RETURN ステートメントはあるが、関数ではなくサブルーチンである PROCEDURE の内部 (例えば、RETURNS オプションを持たない PROCEDURE の内部) にあります。コンパイラーは、この PROCEDURE に対して RETURNS 属性を想定しますが、この想定された属性は、意図したものではない可能性があります。 もっと重要なことは、このルーチンの呼び出し元は CALL ステートメントを使用して呼び出しているため、他の目的のために割り振られたストレージに戻り値を割り当てることになり、予測不能で不定の悪い結果になる可能性があります。

このメッセージを生成するコードは、訂正する必要があります。

IBM2421: ENDFILE での CLOSE

このメッセージは、微妙なコーディング・エラーであることをアラートするものです。

```
IBM2421I E  A file should not be closed in its ENDFILE block.
```

ファイル用の ENDFILE ブロックでそのファイルをクローズさせようとしていますが、内部ライブラリー・エラーが発生する可能性があるため、そうすべきではありません。その代わり、ファイルに対する READ または GET ステートメントの後に検査が行われるフラグを設定するだけの ENDFILE ブロックを作成することが最善です。 このフラグがオンに表示されている場合は、メインライン・コードでファイルをクローズしてください。

このメッセージを生成するコードは、訂正する必要があります。

IBM2610: 精度の解釈

このメッセージは、PL/I 規則を誤って解釈している可能性があり、その結果、問題のソースとなっている可能性があることをアラートするものです。

```
IBM2610I W  One argument to BUILTIN X is FIXED DEC while the other is FIXED BIN
            or FLOAT. Compiler will not interpret precision as FIXED DEC.
```

このメッセージは、MULTIPLY、DIVIDE、ADD、および SUBSTRACT 組み込み関数に適用されます。以下の場合に、このメッセージが出される可能性が最も高くなります。

1. 組み込み関数の引数が 3 つまたは 4 つの場合に、その最後の引数がゼロである。
2. 1 つの引数が FIXED DEC(p1,0) である。
3. 1 つの引数が FIXED BIN(p2,0) である。

例えば、X が FIXED BIN(31) であると、コンパイラーは、式 MULTIPLY(X, 1000, 15) にこのメッセージでフラグを立てます (この式が FIXED DEC(15) 変数に代入される場合であっても)。これは、この関数の結果が属性 FIXED BIN(15) を持つためです。 この組み込み関数が FIXED DEC(15) の結果を生成することを意図していた場合 (例えば、乗算の結果が 2G より大きくなることが分かっているため)、このコードは意図したようには実行されず、有効データを失う結果となることがあります。

この MULTIPLY の結果が FIXED DEC になるよう強制したい場合には、DECIMAL 組み込み関数を FIXED BIN 引数に適用できます (MULTIPLY(DEC(X), 1000, 15) のように)。 PRECTYPE コンパイラー・オプションを使用して、コンパイラーが精度を解釈する方法を変更できますが、当然ながら、これにより他のステートメントの解釈も変更してしまう可能性があります。

IBM2611, IBM2612: 重複する WHEN

「作業」コードの一部をコンパイルしているときに、次のようなメッセージが表示されることもあります。

```
IBM2611I W   The binary value ... appears in more than one WHEN clause.
IBM2612I W   The character string ... appears in more than one WHEN clause.
```

このメッセージは、この章で説明している他のいくつかのメッセージよりも理解しやすく、次のようなコードにより生成されます。

```
SELECT( OPT );
  WHEN( 'f','F' )
    BUFROM = ETOS(OPTARG);
  WHEN( 'T','T' )
    BUTO = ETOS(OPTARG);
  WHEN( 'n','N' )
    MAXRECIN = ETOL(OPTARG);
  WHEN( 'k','K' )
    KFLG = ^KFLG;
  WHEN( 'm','M' )
    MAXERR = ETOL(OPTARG);
  OTHERWISE;
    /* ungueltige Option */
END;
```

メッセージは、上記の 2 つ目の WHEN 文節が、本来はおそらく `WHEN( 't', 'T' )` とコーディングされるはずであったことを示しています。

従来のコンパイラーでは何のメッセージも発行されず、また場合によっては生成されたコードが間違いでないこともあります。しかし、このメッセージを生成するコードがあれば、詳しく調べることをお勧めします。

IBM2617: PL/I 以外のラベルの引き渡し

このメッセージは、一部のソース・コードを編集する必要のある悪いコーディング実践を警告します。

一般に GOTO ステートメントを使用するのはあまり得策とは言えないプログラム実践ですが、特に LABEL 定数または変数を OPTIONS(ASM)、OPTIONS(COBOL) または OPTIONS(FORTRAN) を使用して宣言した ENTRY に渡す場合には、非 PL/I コードから、渡されたラベルを使用して再び PL/I コードに GOTO で戻そうとしてはなりません。 これを実行するコードがある場合には、変更が必要です。

IBM2621: ON ERROR SYSTEM の欠落

「作業」コードの一部をコンパイルしているときに、このメッセージが表示されることもあります。

```
IBM2621I W   ON ERROR block does not start with ON ERROR SYSTEM.
             An error inside the block may lead to an infinite loop.
```

新しいコンパイラーは、ステートメント ON ERROR SYSTEM から始まらないどの ON ERROR ブロックに対してもこのメッセージを生成します。 ON ERROR ブロックがこのステートメントで始まっておらず、ON ERROR ブロックにエラーがあれば、多くの場合、このブロックは再入され、「無限」ループが生じる可能性があります。

このメッセージを生成するコードは、訂正する必要があります。

IBM2622: 不完全にコーディングされた DO ループに対する警告

「作業」コードの一部をコンパイルしているときに、より不明瞭なこのメッセージが表示されることもあります。

```
IBM2622I W   ENTRY used to set the initial value in a DO loop will
             be invoked after any TO or BY values are set.
```

新しいコンパイラーは、次のようなコードに対してこのメッセージを生成します。

```
dcl jx    fixed bin;
dcl last  fixed bin init(10);

do jx = f() to last;
  put skip list( jx );
end;

f: proc returns( fixed bin );
  last = 4;
  return( 2 );
end;
```

このコードでは、ループ内で初期値を設定する関数 f は、ループ内の最終値を設定する変数 *last* の値も変更することに注意してください。 このメッセージは、ループの最終値を設定する変数 *last* を既にコンパイラーが使用した後で、この変数への変更が行われていることを警告しています。 この例の項を具体的に見てみると、ループは 2 から 4 ではなく、2 から 10 で実行されます。

これは、従来のコンパイラーがこのようなコードに対して行った処理と異なります。従来のコンパイラーでは、このループは 2 から 4 で実行されるはずです。

したがって、このコードを従来のコンパイラーの下で動作したのと同じように動作させるには、ソース・コードを変更する必要があります。 呼び出しルーチン内で他の変数を変更するような副次作用を持つ関数があるのは、良いプログラミング実践ではないため、いずれにせよ、ソース・コードの変更は得策です。また、上記のコードはあまり透過的ではありません。不明瞭な作用を持つ分かりにくいコードは、決して良くありません。

IBM2626: 長さゼロを持つ SUBSTR

コードが特に不完全な場合、このメッセージが表示されることもあります。

```
IBM2626I W   Use of SUBSTR with a third argument equal to 0 is
             somewhat pointless since the result will always be a
             null string.
```

コンパイラーがこのメッセージを出してコードにフラグを立てる場合は、ほぼ間違いなくコードに即座に修正する必要のあるエラーが見つかっています。

IBM2628: 大きな BYVLAUE パラメーター

従来のコンパイラーでは、BYVALUE 属性に対して非常に限られたサポートしかなかったため、旧コードのコンパイル時にこのメッセージが表示されることはあまりありません。

```
IBM2628I W   BYVALUE parameters should ideally be no larger than 32 bytes.
```

しかし、BYVALUE 属性をもっと使用し始めると、このメッセージが表示されることがあります。その場合は注意してください。小さいスカラーに対して、および理想としてはレジスターで渡されることができた変数に対して、BYVALUE 属性の使用を予約する必要があります。 一般的には、次のように宣言されます。

- REAL FIXED BIN
- REAL FLOAT
- POINTER
- OFFSET
- HANDLE
- ORDINAL
- CHAR(1)
- ALIGNED BIT(1)
- ALIGNED BIT(8)

ストリングや集合体のサイズが 4096 バイトより大きな BYVALUE 属性は、決して使用しないでください。

IBM2801: スケール付き FIXED BIN の導入

このメッセージは、PL/I 規則を誤って解釈している可能性があり、その結果、問題のソースとなっている可能性があることをアラートするものです。

```
IBM2801I I  FIXED DEC(p1,q1) operand will be converted to FIXED BIN(p2,q2).
            This introduces a non-zero scale factor into an integer operation
            and will produce a result with the attributes FIXED BIN(r,s).
```

このメッセージは、1 つのオペランドはスケール付きの FIXED DEC だが、もう 1 つのオペランドはスケールなしの FIXED BIN である算術演算に適用されます。PL/I 規則では、RULES(IBM) コンパイラー・オプションを指定すると、算術演算の 1 つのオペランドが DECIMAL で、もう 1 つのオペランドが BINARY であると、結果は BINARY になります。これは、DECIMAL オペランドがゼロ以外のスケール因数を持つ FIXED DEC で、BINARY オペランドがゼロのスケール因数を持つ FIXED BIN であっても適用されます。

例えば、X が FIXED DEC(5,1) で Y が FIXED BIN(15) の場合、式 X+Y の計算で、X は FIXED BIN(18,4) に変換され、結果は属性 FIXED BIN(20,4) を持つことになります。 小数部が .0 または .5 でない FIXED DEC(5,1) 値は FIXED BIN で正確に表すことができないため、コンパイラーは、W レベル・メッセージ IBM1099I も出します。

このメッセージが出ないようにし、それが暗示する問題を避けるために、FIXED BIN オペランドに DECIMAL 組み込み関数を適用することができます。例えば、X+DEC(Y) は、属性 FIXED DEC(8,1) を持つ結果を生成します。

IBM2804: 完全に最適化されていない比較

この I レベル・メッセージは、プログラムの熟練度が低く、エラーの可能性があることをアラートします。

```
IBM2804I I Boolean is compared with something other than '1'b or '0'b.
```

2 つの式の比較の結果、またはブールの AND、OR、または否定の結果は、ブールになります。 そのため、ブールは、'1'b または '0'b の値しか持つことができません。これらの値以外の値とブールを比較するようコーディングしている場合は、問題を示すことがあります (例えば、式 (a > b) = c は (a + b) = c という意味だったかもしれません)。

ブールを BIT(1) STATIC INIT('1'b) として宣言された値と比較した場合であっても、コンパイラーはこのメッセージを生成することに注意してください。 この状態はプログラミング・エラーではありませんが、コンパイラーは、値が BIT(1) VALUE('1'b) として宣言されて生成されたコードほど良好なコードを生成することができません。

IBM2810: スケール付き FIXED BIN の変換

コードの一部をコンパイルしているときに、次のメッセージが表示されることがあります。

```
IBM2810I I Conversion of FIXED BIN(31,16) to FIXED DEC(15,12) may
           produce a more accurate result than under the old
           compiler.
```

このメッセージが生成されるコードの例およびその場合の対応方法の説明については、 132 ページの『スケール付き FIXED BINARY からの変換』を参照してください。

IBM2811: DO 制御変数としての PICTURE の使用

このメッセージは、一部のソース・コードを編集する必要のある、不完全なコーディング実践をアラートするものです。OPT(0) を使用した場合でも、新しいコンパイラーは、制御変数に PICTURE 属性を持つ DO ループにフラグを立てます。コンパイラーは、次の通知メッセージを発行します。

```
IBM2811I I Use of PICTURE as DO control variable is not recommended.
```

一般に、DO ループ制御変数として PICTURE 変数を使用するのは、あまり得策とは言えないプログラミング実践です (特にローパフォーマンスになってしまう可能性があるため)。そのようなコードは、FIXED BIN 変数をループ制御変数として使用するように変更するのが得策です。

IBM2812: 不完全な TRANSLATE および VERIFY

このメッセージは、従来のコンパイラーでは OK だったけれども、新しいコンパイラーではさらに良い代替があるという、コーディング実践をアラートするものです。属性 STATIC INIT ではなく、属性 VALUE で名前付き定数を宣言することができるようになりました。

この変更は、特に次のようなコードで役立ちます。

```
test: proc( c );

  dcl c char(20);

  dcl upper char(26) static init('ABCDEFGHIJKLMNOPQRSTUVWXYZ');
  dcl lower char(26) static init('abcdefghijklmnopqrstuvwxyz');

  c = translate( c, upper, lower );
end;
```

名前付き定数 *upper* と *lower* は STATIC INIT として宣言されているため、新旧どちらのコンパイラーもランタイムに変換テーブルを作成します。これは、非常に負荷がかかります。 しかし、新しいコンパイラーでは、次の通知メッセージも発行されます。

```
IBM2812I I Argument number 2 to TRANSLATE built-in would lead to
               much better code if declared with the VALUE attribute.
    IBM2812I I Argument number 3 to TRANSLATE built-in would lead to
               much better code if declared with the VALUE attribute.
```

両方の宣言の STATIC INIT を VALUE に変更すると、これらのメッセージは出なくなり、コンパイラーはより良いコードを生成します。

PLIXOPT メッセージ

PLIXOPT 変数は、コンパイル時に指定できるランタイム・オプションが含まれた可変長文字ストリングです。これらのオプションでのエラーを診断するためにコンパイラーが出力するメッセージは、従来のコンパイラーで出力されていたメッセージとは異なります。ほとんどの場合、PL/I メッセージには、必要な詳細情報が示された関連する言語環境プログラム・メッセージがリストされます。

PLIXOPT 宣言を含むモジュールには、PLIXOPT ストリングに指定するランタイム・オプションの言語環境プログラム・エンコードを含む、コンパイラー生成の

CEEUOPT CSECT も含まれるようになりました。 小さなモジュールの場合、この CSECT はモジュールのオブジェクト・サイズが大幅に増大する原因となります。

コンパイラー・ユーザー出口の使用

より高い重大度をメッセージに与える場合、EXIT コンパイラー・オプションを使用できます。このオプションによって、任意の通知メッセージ、警告メッセージ、またはエラー・メッセージの重大度を非常に簡単に引き上げることができます。

このオプションの使用法について詳しくは、「*Enterprise PL/I for z/OS* プログラミング・ガイド」を参照してください。

第 13 章 作業コードを変更する必要がある場合について

この章では、新しいコンパイラーで、従来のコンパイラーとは異なるコードが生成される状況について説明します。この章で扱う問題は、Enterprise PL/I にマイグレーション済みのお客様にとって重要であるため、この章をよくお読みになり、これらの問題の影響を受ける可能性があるかどうかを確認してください。

以下の節で説明する問題のあるコードのいくつかにはコンパイラーによってフラグが立てられており、発行されたメッセージと関連付けられたコードを調べる必要があります (必要に応じて変更します)。 特に、次のメッセージを出力するコンパイルを調べる必要があります。

- IBM1063
- IBM1089

なお、オプション DECIMAL(NOFOFLONASGN) DFT(OVERLAP) および STATIC(FULL)を使用することにより、これらの問題のいくつかを解消できる場合があります。

間違ったコード

ご使用のコードは、PL/I の規則に準拠した正しいコードでなければなりません。Enterprise PL/I コンパイラーは、間違ったコードに関して従来のコンパイラーとは異なる結果 (異常終了を含む) を生成する可能性があります。場合によっては、幸運にも間違ったコードにより意図通りの結果が得られるかもしれませんが、こうしたコードを当てにしてはいけません。 間違ったコードは変更する必要があります。

規則が明白な場合もあります。例えば、配列の境界外にある索引を使用してその配列のエレメントに書き込もうとするユーザーはいないでしょう。 しかし、コードが間違っていて変更する必要があるという事実がほとんど分からない場合もあります。 このセクションでは、変更が必要な間違ったコードの事例について幾つか説明しますが、正しくないコードについて書くと際限がなくなりますので、間違ったコードすべてをリストしている訳ではありません。

宣言の順序を当てにする

ある変数を別の変数に続いて宣言する場合、ストレージ内でそれらの変数が隣り合っている、または 2 番目の変数がストレージ内で 1 番目の変数の後にあると推測してはなりません。

例えば以下のコードの場合、変数 *a* に割り振られているストレージは変数 *b* に割り振られているストレージの直後にない可能性もあるので、この代入は他の変数に割り振られているストレージの 100 バイトをオーバーレイすることがあり得ます。

```
dcl a char(100);
     dcl b char(100);
     dcl c char(200) based;
     addr(a)->c = '';
```

© Copyright IBM Corp. 1999, 2016

実際には変数 *b* が未使用の場合、コンパイラーはその変数にストレージをおそらく割り振りません。

無効な FIXED DECIMAL データを使用する

使用するすべての FIXED DECIMAL 変数は、有効なデータを含んでいる場合に限り使用してください。

FIXED DECIMAL 変数に無効データ (数字の桁や符号ニブルが間違っているなど) が含まれている場合、そうした変数を使用するとデータ例外が生じます。 類似した精度およびスケールを持つ変数間などの代入であっても (たとえバイト移動による代入でも)、データ例外が生じる場合があります。

しかし、そのような変数を 1 つ使用するだけでデータ例外が生じると考えないでください。例えば、前述の代入をある状況下でバイト移動を使用して実行する場合、データ例外は算術演算や比較などで使用しない限りは生じません。

無効な SUBSTR 参照を使用する

使用する SUBSTR 参照は、STRINGRANGE 条件が有効な場合にその条件が生じないようにして使用しなければなりません。

STRINGRANGE 条件が無効な場合 (デフォルトでは無効)、無効な SUBSTR 参照があるとコンパイルされたコードによって他の目的で割り振られたストレージが上書きされる可能性があり、それによりデータ破損や異常終了の原因となる場合があります。

例えば以下のコードの場合、変数 *n* の値が 100 より大きくて SUBSTR 参照が無効な場合には、他の変数に割り振られているストレージが生成されるコードによって上書きされる可能性があります。

```
dcl f ext entry;
     dcl a char(100);
     call f( 'test' || substr(a,1,n) );
```

PREFIX(STRINGRANGE) コンパイラー・オプションを使用してプログラムをコンパイルしてテストする際に、こうした正しくないコードを簡単に検出できます。

V3R8 で USAGE コンパイラー・オプションに導入された SUBSTR サブオプションを使用して、この正しくないコードの一部が受け入れられるようにできます。 ただし、このオプションは使用しないようお勧めします。代わりに、ご使用のコードを修正してください。上の例では、*a* の宣言を、少なくとも *n* によって想定される最大値と同じ長さになるよう変更することによって、コードを修正します。*n* の最大値が不明であれば、*a* の宣言を長さ 32767 になるよう変更することができます。

一部の状態では、旧コンパイラーでは、形式 SUBSTR(*X*,1,*N*) の SUBSTR 参照のコードも生成されました (ここで、*X* は CHAR、*N* は 32767 より大きい数字)。しかし、このような参照は無効であり、使用可能になっていれば STRINGSIZE が発生しました。 新しいコンパイラーでは、SUBSTR 参照の長さは、CHAR および BIT 参照では 32768 未満、GRAPHIC および WIDECHAR 参照では 16384 未満であることが強制されるため、これらの規則に従わないすべてのコードは訂正する必要があります。

異なる EXTERNAL 宣言を使用する

複数のコンパイル単位で EXTERNAL 変数を宣言する場合には、そのような複数の宣言が一致していなければなりません。 特に 2 つの宣言内のすべての属性を一致させてください。

例えば 1 つのコンパイル単位で EXTERNAL FILE を属性 KEYED ENV(VSAM) を複数指定して宣言する場合には、この最初のものにリンクされている他のプログラムでも同じ属性を指定して宣言する必要があります。

間違った PLITABS 宣言を使用する

コードに PLITABS の宣言が含まれている場合、ページ・サイズ、行サイズ、および他の値が有効でなければなりません。また、PLITABS 構造体の最初のフィールドも有効でなければなりません。

このフィールドには、構造体によって設定されるタブの数を指定するフィールドへのオフセットを保持することになっており、これが真でないと、Enterprise PL/I ライブラリー・コードは正常に動作しません。

変数を初期化する

初期化しないで変数を使用すべきではありません。初期化されていない変数を使用しているプログラムは無効で、訂正が必要です。

こうしたコードを訂正する最善の方法は、必要な変数に INITIAL 属性を追加することです。 ただし、変数を初期化する他の方法もあり、それらの代替手段についてこのセクションで取り上げます。

AUTOMATIC の初期設定

このトピックでは、さまざまなオプションが有効であるときに、コンパイラーがどのように AUTOMATIC を初期設定するかについて説明します。

AUTOMATIC 変数に以下のいずれかの属性がある場合にコンパイラー・オプション INITAUTO を指定すると、INITIAL 属性が指定されていない AUTOMATIC 変数に対して適切な INITIAL 属性が追加されます。

- FIXED または FLOAT
- PICTURE、CHAR、BIT、GRAPHIC または WIDECHAR
- POINTER または OFFSET

詳細については、「プログラミング・ガイド」を参照してください。

コンパイラー・オプション DFT(INITFILL) を使用すると、すべての AUTOMATIC ストレージは指定のバイト値 (またはバイト値を指定しない場合には '00'x) で埋められます。以下の属性を持つ変数を初期化するのに使用できます。

- FIXED BIN
- FLOAT
- VARYING または VARYINGZ
- POINTER または OFFSET

またコンパイラー・オプション INITFILL は他のすべての AUTOMATIC 変数も指定 (またはデフォルト) のバイト値で埋めますが、こうした変数は実際には適切に初期化されません。 例えば、DFT (INITFILL) を介して初期化された FIXED DEC 変数を使用すると、すぐにデータ例外が生じます。

ランタイム・オプションの 3 番目のサブオプションを 00 に設定すると (例 STORAGE(,,00))、すべてのルーチン (ライブラリー・ルーチンを含む) 内のすべての AUTOMATIC ストレージは 16 進数値 00 で埋められます。 これは、DFT(INITFILL) コンパイラー・オプションと同じ効果および有効性を持っていますが、アプリケーション内のすべてのルーチンに適用され、パフォーマンスにかなり悪い影響を与えるという点が異なります。 またコンパイラーはこのオプションが使用されているかどうかを認識しないため、OPT(2) または OPT(3) を使用してコンパイルされたコードに対して望ましくない影響を与える可能性があります。変数が初期化されていないとコードが無効になりますし、最適化プログラムがこのランタイム・オプションを使用して修復できないコードの最適化方法を選択することになる場合があります。

ランタイム・オプションの 3 番目のサブオプションを CLEAR に設定すると (例 STORAGE(,,CLEAR))、MAIN が呼び出される前にすべての AUTOMATIC ストレージは 16 進数値 00 で埋められます。 これは DFT(INITFILL) コンパイラー・オプションと同じ効果および有効性を持っていますが、ただし MAIN ルーチンに対してのみ適用される点が例外です。 またコンパイラーはこのオプションが使用されているかどうかを認識しないため、OPT(2) または OPT(3) を使用してコンパイルされたコードに対して望ましくない影響を与える可能性があります。変数が初期化されていないとコードが無効になりますし、最適化プログラムがこのランタイム・オプションを使用して修復できないコードの最適化方法を選択することになる場合があります。

BASED の初期設定

コンパイラー・オプション INITBASED は、INITAUTO が AUTOMATIC に対して作用するのと同様に、BASED に対して作用します。

CONTROLLED の初期設定

コンパイラー・オプション INITCTL は、INITAUTO が AUTOMATIC に対して作用するのと同様に、CONTROLLED に対して作用します。

STATIC の初期設定

コンパイラー・オプション INITSTATIC は、INITAUTO が AUTOMATIC に対して作用するのと同様に、STATIC に対して作用します。

このオプションを指定しないと、初期化されていないすべての STATIC ストレージは 2 進ゼロで埋められます。 当然、前述のコンパイラー・オプション DFT(INITFILL) およびランタイム STORAGE オプションを使用する場合にはこれが当てはまりますが、これにより、FIXED DEC 変数などの多くの変数では無効な値を持つことになります。

使用されない宣言の保持

このセクションでは、新しいコンパイラーが未使用の宣言のストレージ割り振りをどのように処理するかについて説明します。

使用されない INTERNAL STATIC の保持

INTERNAL 静的変数が使用されていない場合、コンパイラーは、この変数にストレージを割り振りません。

例えば次の宣言が、変数 *build_data* への唯一の参照である場合、この変数にはストレージは割り振られず、この変数の初期値は生成されるテキストには含まれません。

```
dcl build_data char(30) var static
               init('Compiled in build 17');
```

レベル 1 静的変数で ABNORMAL 属性が指定されている場合、コンパイラーは、この変数にストレージを割り振ります。 例えば、上記の変数を保持するには、上記の宣言を次のように変更します。

```
dcl build_data char(30) var static abnormal
               init('Compiled in build 17');
```

ABNORMAL 属性は、すべての変数または静的変数に無差別に設定してはいけません。この結果、コンパイルの速度が低下するとともに、生成されたコードのパフォーマンスが低下するからです。

コンパイラー・オプションの STATIC(FULL) を指定すると、コンパイラーは ABNORMAL 属性をすべての静的変数に適用します。 これはずさんな解決策であり、推奨されません。

例外が発生するようになった間違ったコード

SIZE が無効になっている場合の FIXEDOVERFLOW

新旧のコンパイラーのどちらでも、代入元を数値代入先に代入しようとした場合に代入元の値が大きすぎると、SIZE 条件が発生します (SIZE 条件が有効になっている場合)。ただし、SIZE 条件が有効になっていない場合は、プログラムは間違っており、どのような処理が発生するのかは予測できません。 このようなプログラムは修正する必要があります。

従来のコンパイラーでは、どのような条件も発生しない場合があります。例えば、次のプログラムを見てみましょう。

```
dcl A fixed dec(3);
dcl B pic'9';

A = 123;
B = A;
```

代入元 *A* の値は大きすぎて *B* には収まらないため、SIZE 条件が有効になっている場合は、SIZE 条件が発生します。しかし SIZE 条件が無効になっている場合は、従来のコンパイラーではどのような条件も発生しません。 これは、プログラムが正

しいこと示しているのではなく、実際にはプログラムは間違っており、変更する必要があります。 例えば、*B* を *A* の 1 の桁だけに設定する場合は、上記のコードを次のように変更します。

```
dcl A fixed dec(3);
dcl B pic'9';

A = 123;
B = mod(A,10);
```

また、従来のコンパイラーでは、非常によく似たコードについて条件が発生することがあります。例えば、次のプログラムを見てみましょう。

```
dcl X fixed dec(5);
dcl Y fixed dec(4);
dcl Z fixed dec(5);

X = 99999;
Y = X + 1;
Z = X + 1;
```

式 *X + 1* の値は大きすぎて *Y* と *Z* のどちらにも収まらないため、SIZE 条件が有効になっている場合は、両方のステートメントについて SIZE 条件が発生します。しかし SIZE 条件が無効になっている場合は、従来のコンパイラーでは *Y* への代入については何の条件も発生せず、*Z* への代入については FIXEDOVERFLOW が発生します。この場合もまた、プログラムは間違っており、変更する必要があります。

新しいコンパイラーでは、これらのステートメントを一貫性を持って処理しますが、結果は、有効であるターゲット属性とコンパイラー・オプションによって異なります。

- SIZE 条件が無効になっている場合、ターゲットに PICTURE 属性があれば、生成されたコードは FIXEDOVERFLOW 条件を発生させません。

  より正確には、浮動小数点でない PICTURE ターゲットに対して以下のいずれかのデータ・タイプを持つソース式を代入するときに、SIZE 条件が使用不可になっていると、生成されたコードは FIXEDOVERFLOW 条件を発生させません。

  – FIXED BIN
  – FIXED DEC
  – 浮動小数点でない PICTURE
- SIZE 条件が無効になっている場合、ターゲットに FIXED DEC 属性があれば、以下のようになります。
  – デフォルトのコンパイラー・オプション DECIMAL(FOFLONASGN) が有効であると、生成されたコードは FIXEDOVERFLOW 条件を発生させます。
  – コンパイラー・オプション DECIMAL(NOFOFLONASGN) が有効であると、生成されたコードは FIXEDOVERFLOW 条件を発生させません。

上記の説明は代入の場合にのみ適用されることに注意してください。加算または乗算などの演算で、15 桁を超える (LIMITS(FIXEDDEC(15,31)) オプションが有効である場合には 31 桁を超える) 結果が生成される場合には、例外が発生します。通常、発生する例外は FIXEDOVERFLOW ですが、生成されるマシン・インストラクションによっては、指定例外など、その他の例外が発生することがあります。

同様に、BIT 変数が FIXED BIN 変数に代入されるときに、BIT 変数が大きすぎて有効に変換できず、SIZE が有効になっていなかった場合 (したがって、プログラムが無効だった場合)、以下のようになる可能性があります。

- 従来のコンパイラーでは、時として何の条件も発生せず、単にターゲットに 0 を代入します。
- 新規のコンパイラーでは、変換がライブラリー呼び出しで行われる場合、SIZE 条件が発生します。

例えば、次のプログラムを見てみましょう。

```
dcl A bit(32) aligned;
dcl B fixed bin(31);

A = '80000000'bx;
B = A;
```

代入元 *A* の値は大きすぎて *B* には収まらないため、SIZE 条件が有効になっている場合は、SIZE 条件が発生します。新規のコンパイラーでは、SIZE 条件が無効になっている場合でも、このコードに対して SIZE 条件が発生します。 しかし SIZE 条件が無効になっている場合は、従来のコンパイラーではどのような条件も発生しません。 これは、プログラムが正しいことを示しているのではなく、実際にはこのプログラムは間違っており、変更する必要があります。

無効な割り振り

従来のコンパイラーでは、次のコードは「正常に」動作しました。

```
dcl vdptr  pointer;
dcl vdcom  char(2000) based(vdptr);

dcl
  1 vdcommarea based(addr(vdcom)),
    2 vda  char(1000),
    2 vdb  char(1000),
    2 vdz  char(1);

alloc vdcom;

vdcommarea = '';
```

このコードは有効な PL/I コードではありません。その理由は、2001 バイトの領域を使用して、2000 バイトの割り振られたストレージ部分をオーバーレイしてはいけないからです。OS PL/I V2R3 のランタイム環境では、幸いにもこのコードは「正常に動作」しましたが、言語環境プログラムのランタイム環境では、このコードではエラーが発生します。

無限ループが発生するようになった間違ったコード

このセクションでは、無限ループを引き起こす間違ったコードについて説明します。このようなシチュエーションでは、新しいコンパイラーはメッセージでアラートを出します。コンパイラーにそのようなメッセージを発行させるコードがある場合は、そのコードを詳しく調べ、場合によっては変更する必要があります。

偶数精度の PICTURE ループ制御変数

配列を初期化する次のプログラムについて見てみましょう。

```
winter: proc;
   dcl n pic'99';
   dcl a(0:99) fixed bin ext;
   do n = 0 to 99;
      a(n) = n;
   end;
end;
```

このコードは、有効な PL/I コードではありません。 SIZE 条件が有効になっている場合、*n* の値が 99 になった後に SIZE 条件が発生します (想定される次の値 100 は PIC'99' 変数にとって大きすぎるため)。

最適なパフォーマンスを得るためには、ループ制御変数として PICTURE 変数を使用することは通常、適切ではありません。 しかし、上記のコードの場合は、新しいコンパイラーによって、このループを無限に実行させるコードが生成されるため、PICTURE 変数を使用することは非常に不適切なことです。

DO ステートメントの定義上、上記のループは、新旧コンパイラーのどちらでも無限ループを発生させる次のコードと同じ意味です。

```
n = 0;
if n > 99 then go to loop_exit;
loop_body:;
   a(n) = n;
n = n + 1;
if n <= 99 then go to loop_body;
loop_exit:;
```

ただし、DO ループを使用している元のコードの場合、従来のコンパイラーでは、不正な方法により、厳密には間違っているコードが生成されます。

新しいコンパイラーでは、次のメッセージが発行されて、ユーザーにこの状況についての警報が出されます。

```
IBM1089I W    Control variable in DO loop cannot
              exceed TO value, and loop may be infinite.
IBM1220I W    Result of comparison is always constant.
IBM1220I W    Result of comparison is always constant.
```

コンパイラーにメッセージ IBM1089 を発行させるコードがある場合は、このコードを詳しく調べる必要があります (場合によっては変更します)。 また、EXIT オプションを使用して、このメッセージの重大度を上げることもできます。

コードを修正する場合は、DO ループ制御変数の属性を PICTURE から FIXED BIN(31) に変更します。

最後に、この問題は、DO ループ制御変数が PICTURE'(*n*)9' であり、*n* が偶数でループ制限が 10**-n*-1 であるいずれのループでも発生するため注意してください。

またこの問題は、コンパイラーによってフラグが立てられていない形で生じる可能性があります。例えば、配列を初期化する次のプログラムについて見てみましょう。

```
sommer: proc;
   dcl n pic'999';
   dcl a(0:999) fixed bin ext;
   do n = 0 to 998 by 2;
      a(n) = n;
   end;
end;
```

この場合、TO の値 998 は *n* が想定する最大値よりも小さいため、コンパイラーはメッセージ IBM1089 を発行しません。 しかし *n* が値 998 を想定した後にループを次に通る際、*n* には値 0 が割り当てられ、ループが繰り返されます。

またこの問題は、BY 値が負であるときにも生じ得ます。

```
eiki: proc;
   dcl n pic'999';
   dcl a(0:99) fixed bin ext;
   do n = 79 to 1 by -2;
      a(n) = n;
   end;
end;
```

しかし *n* が値 1 を想定した後、次にループを通る際、 *n* は 2 減少して値 1 が割り当てられてループが繰り返されます。

異なる結果を生成する代入

このセクションでは、従来のコンパイラーと新しいコンパイラーとの間で結果が異なる可能性がある割り当てについて説明します。コードを互換性があるものにするために、そのコードを変更するか、または特定のコンパイラー・オプションを使用しなければならない場合があります。

代入元と代入先の重複

`P->Z = Q->Z;` という代入について見てみましょう。ここで、Z は CHAR(6) BASED です。

OPT(0) を使用した場合、従来のコンパイラーは、代入元をまず 6 バイトの一時変数に代入してから、この一時変数を代入先に代入します。

ただし OPT(2) を使用した場合、従来のコンパイラーは、1 つの MVC を使用してこの代入を実行します。

代入元と代入先が重複している場合は、これらの異なるインプリメンテーション方法により異なる結果が得られます。

新しいコンパイラーでは、DEFAULT コンパイラー・オプションの OVERLAP サブオプションにより、次のようにこの動作が制御されます。

- DFT(NOOVERLAP) を使用した場合は、コンパイラーは、代入元と代入先がまったく重複していないものと想定します。
- DFT(OVERLAP) を使用した場合は、コンパイラーは、必要に応じてより保守的なコードを生成します。

例えば SUBSTR(A,4,6) = SUBSTR(A,3,6); という代入について、A = 'abcdefghijklm' である場合、コンパイラーは A の値を次のように設定します。

- 従来のコンパイラーは、A = 'abccdefghjklm' を設定します。
- 新しいコンパイラーは、DFT(OVERLAP) が使用されていれば、A = 'abccdefghjklm' を設定します。
- 新しいコンパイラーは、DFT(NOOVERLAP) が使用されていれば、A = 'abccccccjklm' を設定します。

したがって、最小限の作業で最大限の互換性を得るには、コンパイラー・オプション DFT(OVERLAP) を指定するとよいでしょう。

ただしこのオプションを指定した場合、コンパイラーは、代入元と代入先が重複しないことが分かっている状況では低速なコードを生成するとともに、他のいくつかの最適化処理を省略します。 代入元と代入先が重複しないようにコードを変更してから DFT(NOOVERLAP) を使用することをお勧めします。

例えば、次の代入について考えてみます。

```
SUBSTR(A,4,6) = SUBSTR(A,3,6);
```

この代入は、次の代入で置き換えることができます。

```
temp_Char6 = SUBSTR(A,3,6);
SUBSTR(A,4,6) = temp_Char6;
```

float 間の代入

新しいコンパイラーは、3.1415926E0 や 1E-02 などの FLOAT DECIMAL リテラルを、このリテラルが使用されているコンテキストは調べずに、このリテラルの属性のみを調べて、内部浮動小数点表記に変換します。

例えば、3.1415296E0 の属性は FLOAT DEC(8) であるため、新しいコンパイラーは、この値を長精度浮動小数点に変換します。 しかし、1E-02 の属性は FLOAT DEC(1) であるため、新しいコンパイラーは、この値を短精度浮動小数点に変換します。

リテラルが代入または INITIAL 文節で使用されている場合、コンパイラーは、必要に応じて、このリテラルの浮動小数点値を代入先または初期化対象の属性に変換します。

ただし従来のコンパイラーは、このようなリテラルが使用されているコンテキストを調べて、リテラルを直接その対象の属性に変換します。 従来のコンパイラーの動作は、PL/I 式評価に関する規則に厳密に従っていないため、新しいコンパイラーの場合とは異なる結果が生成されることがあります。 次のコード・フラグメントを見てください。

```
dcl z float dec(06) init(0);
dcl s float dec(06);
dcl q float dec(17);

s = 1e-2;
q = s;
put skip data( q );
```

```
q = 1e-2;
put skip data( q );
q = 1e-2 + z;
put skip data( q );
```

上記の *q.* への 3 つの代入のすべてにおいて、代入元の属性は短精度浮動小数点数の属性であり、代入元の値は同じになるはずです。 しかし、従来のコンパイラーでは、3 つの PUT ステートメントの結果は次のようになります。

```
Q= 9.999997913837432861E-03;
Q= 9.999999999999999999E-03;
Q= 9.999999999999999999E-03;
```

新しいコンパイラーでは、3 つの PUT ステートメントの結果は次のようになります。

```
Q= 9.999997913837432860E-03;
Q= 9.999997913837432860E-03;
Q= 9.999997913837432860E-03;
```

このような相違は、正確に表現できない float 型リテラルについてのみ発生します (同値の 2 進小数がない 1E-2 のような小数など)。

上記のような状態をアラートするために、コンパイラーは、正確に表現できない短精度浮動小数点リテラルを検出すると、メッセージ IBM1065I を出します。

従来のコンパイラーを使用した場合と同じ結果を得るには、次のいずれかの方法でソースを変更する必要があります。

1. FIXED DECIMAL リテラルに適用される FLOAT 組み込み関数と必要な精度を使用して定数を指定する。

   例えば、1E-2 を長精度浮動小数点値にするには、FLOAT(.01,7) として指定し、拡張浮動小数点値にするには、FLOAT(.01,17) として指定します。
2. 十分な数のゼロをリテラルに追加して必要な精度を指定する。

   例えば、1E-2 を長精度浮動小数点値にするには、1.000000E-2 として指定し、拡張浮動小数点値にするには、1.00000000000000000E-2 として指定します。
3. 新しい D 形式または Q 形式を使用して必要な精度を指定する。

   例えば、1E-2 を長精度浮動小数点値にするには、1D-2 として指定し、拡張浮動小数点値にするには、1Q-2 として指定します。

上記の 1 つ目と 2 つ目の変更は新旧のコンパイラーで許容されますが (どちらのコンパイラーでも同じ結果が生成されます)、3 つ目の変更は、新しいコンパイラーでのみ有効なので注意してください。

異なる結果を生成するその他のステートメント

このセクションには、従来のコンパイラーと新しいコンパイラーとの間で結果が異なる可能性がある、いくつかの他のステートメントに関する情報があります。

印刷不能な文字が含まれた STREAM 入出力

いくつかのシナリオでは、'00'x、'0C'x から '0F'x、または '15'x という値を持つ文字が、PUT FILE ステートメントの出力に含まれていた場合は、これらの文字の代わりにピリオド ('4B'x) が出力されます。

シナリオを下にリストします。

- コードがバッチで実行され、STDSYS オプションを指定してコンパイルされ、ファイルが SYSPRINT で、SYSPRINT が SYSOUT に送信される。
- コードが z/OS UNIX で実行され、ファイルが、コマンド・ウィンドウに書き込まれる STREAM OUTPUT ファイルである
- コードが TSO で実行され、ファイルが、TSO 端末に書き込まれる STREAM OUTPUT ファイルである。

初期化されていない EXTERNAL STATIC

未初期化 EXTERNAL STATIC 変数のストレージ割り振りにおいて、従来のコンパイラーと新しいコンパイラーは異なります。

従来のコンパイラーでは、EXTERNAL STATIC と宣言されたのに INITIAL 値の指定がない変数は、ストレージを割り振られることはありませんでした (そしてタイプ CM のリンケージ・エディター ESD が発行されました)。 その変数のストレージは、リンクしている他のプログラム・オブジェクトで定義する必要がありました。 事実、実際に割り振られるストレージは、宣言で指定する (または暗黙に指定する) ものよりも大きい可能性もありました。例えば、以下のコード宣言について考えてみます。

```
dcl testpcl ext static, pcl char(16) based(addr(testpcl));
```

変数 *testpcl* は (暗黙の) FLOAT DEC(6) 属性を持ち、そのため 4 バイトのストレージのみを割り振られるかのように見えます。このプログラムとリンクするすべてのプログラムが同じ PL/I 宣言で *testpcl* を宣言した場合、4 バイトが割り振られます。しかし、それが例えば *testpcl* を 16 バイトの CSECT として定義するアセンブラーとリンクされていた場合、リンカーはその変数に 16 バイトを割り振ります。

新しいコンパイラーは現在、そのような変数に 4 バイトを割り振ります (そして、タイプ SD で長さ 4 のリンケージ・エディター ESD を発行します)。 その変数を例えば 16 バイト領域のベースとして使おうとすると、エラーになります。

変数を宣言し、そのストレージ割り振りを他のモジュール内の宣言によって決定する場合は、RESERVED オプションを使用して宣言してください。 例えば、上の宣言は次のようになります。

```
dcl testpcl ext static reserved, pcl char(16) based(addr(testpcl));
```

ただし、オプション COMMON を指定してコンパイルした場合、Enterprise PL/I コンパイラーはタイプ CM のリンケージ・エディター ESD も発行し、そのコードは旧コンパイラーと同じ動作を行います。

不完全に宣言された FILE

従来のコンパイラーと新しいコンパイラーとの間で、不完全に宣言された FILE の処理方法がどのように異なるかを知っておく必要があります。

旧コンパイラーでは、あるルーチンで RECORD などの一部の属性を指定して EXTERNAL FILE を宣言しているが、この最初のルーチンとリンクした別のルーチンでは、ファイルを宣言していなかったか、または属性なし (FILE 以外) で宣言していた場合、この別のルーチンは、その別のルーチンが最初にそのファイルをオープンしたとしても、最初のルーチンで宣言された属性を使用することになります。

Enterprise PL/I の場合は、これとは異なる処理を行います。つまり、別のルーチンは最初のルーチンからの属性を「確認」せず、代わりに STREAM などのデフォルト属性をファイルに適用します。 これによって、問題を引き起こす可能性があります。

このコードを訂正して、すべてのルーチンで等しく FILE を宣言するようにする必要があります。実際には、すべての EXTERNAL 変数をすべてのルーチンで等しく宣言してください。

仮引数と配置

PL/I 言語解説書 に記述されているように、引数の配置がパラメーター記述と異なる場合は、仮引数が作成されます。しかし、従来のコンパイラーは、CHARACTER NONVARYING に関してはこのルールに従いましたが、CHARACTER VARYING に関しては従いませんでした。 新しいコンパイラーでは、このルールを一貫して適用します。

例えば、次のコードについて考えてみます。

```
dcl x entry( unaligned char(8) );
dcl y entry( unaligned char(8) varying );
dcl a aligned char(8);
dcl b aligned char(8) varying;
call x( a );
call y( b );
```

この例では、両方の CALL ステートメントに対して仮引数が作成されるべきですが、2 番目の CALL に対しては Enterprise PL/I コンパイラーのみが仮引数を作成します。

なお、DEFAULT(DUMMY(UNALIGNED)) コンパイラー・オプションを使用することで、コンパイラーが仮引数を作成するかどうか判断するときに、配置の不一致を無視させることができることに注意してください。 このオプションが有効である場合は、コンパイラーは上記の例のどちらの CALL に対しても仮引数を作成しません。

仮引数と CONTROLLED

「PL/I 言語解説書」に記述されているように、引数が CONTROLLED ストリング (または領域) である場合は (ALLOCATE ステートメントが長さまたはエクステントを変更した可能性があり、それによりストリングの長さや領域サイズが、呼び出されるルーチンの要求と異なってしまったため)、仮引数が作成されます。新しいコンパイラーはこのルールを一貫して適用しますが、従来のコンパイラーはこれを行いません。

Enterprise PL/I では、RULES(NOLAXCTL) オプションが有効でストリングの長さまたは領域サイズが定数でない限り、このルールが適用されます。しかし、従来の

コンパイラーは必ずしもこのルールに一貫して従っていたわけではありません (従来のコンパイラーに RULES(NOLAXCTL) オプションと同等のオプションはなかったので、本来は適用されるべきでしたが)。

例えば、次のコードについて考えてみます。

```
dcl x entry( char(8) );
dcl a controlled char(8);
dcl 1 b(2) controlled, 2 c char(8);
call x( a );
call y( b(1).c );
```

この例では、両方の CALL ステートメントに対して仮引数が作成されるべきですが、2 番目の CALL に対しては Enterprise PL/I コンパイラーのみが仮引数を作成します。

ポインター算術

ポインター算術を含む式では、ポインターはアドレスであると想定されます。 ですからポインターに値を追加する場合、ソース・ポインターに高位ビットがオンだった場合でも、結果ポインターでは高位ビットがオンでない場合があります。

パフォーマンスが劣るコード

このセクションでは、パフォーマンス改善のために変更を考慮しなければならない可能性があるコードについて説明します。

FIXED DEC をループ制御変数として使用

FIXED DECIMAL または PICTURE 制御変数が使用されている DO ループは、FIXED BINARY 制御変数が使用されているループよりもパフォーマンスが大幅に劣ります。

ループ制御変数の宣言を FIXED DEC から FIXED BIN(31) に変更することにより、コードのパフォーマンスを大幅に向上させることができます。

FIXED BIN(15) をループ制御変数として使用

FIXED BIN(15) 制御変数が使用されている DO ループは、 FIXED BIN(31) 制御変数が使用されているループよりもパフォーマンスが劣ります。

OPT(2) または OPT(3) を使用した場合、コンパイラーは、I レベルのメッセージ IBM1063 を発行して、FIXED BIN(15) 制御変数が使用されているコードにフラグを立てます。ループ制御変数の宣言を FIXED BIN(15) から FIXED BIN(31) に変更することにより、コードのパフォーマンスを向上させることができます。

TOTAL を使用した入出力

ENVIRONMENT 属性の TOTAL オプションはサポートされていないため、このオプションを使用して行われるファイルへの入出力操作は、一般にパフォーマンスが劣ります。

第 14 章 作業コードを変更する必要がある可能性のある場合について

この章では、新しいコンパイラーで、従来のコンパイラーとは異なるコードが生成されるその他の状況について説明します。ただし前章とは異なり、これらの相違点はやや不明瞭です。これらが本書に記載されている理由は、詳細な情報を提供するため、およびユーザーがこれらによる影響を受ける可能性があるためです。

例外が発生するようになったコード

このセクションには、新しいコンパイラーにおいて例外を発生させるコードに関する情報があります。

ERROR にプロモートされる ZERODIVIDE および OVERFLOW

新しいコンパイラーでは、ZERODIVIDE または OVERFLOW の処理方法が改善されています。

従来のコンパイラーでは、ZERODIVIDE 条件または OVERFLOW 条件が発生して、この条件用の ON ユニットがあった場合、この ON ユニットの END ステートメントに到達すると、プログラムは、この条件を発生させたマシン・インストラクションの次のマシン・インストラクションの処理を続行していました。

この条件がハードウェア例外によって発生した場合は、プログラムが、演算の結果得られた何らかの不明な値がある状態で処理を続行したことを示しており、多くの場合、この結果さらにエラーが発生しました。

新しいコンパイラーでは、ZERODIVIDE 条件または OVERFLOW 条件が ON ユニットによって処理されずに放置された場合は、この条件は ERROR にプロモートされます。

使用不可の場合に発生する条件

旧コンパイラーでは、CONVERSION または SUBSCRIPTRANGE などの条件が使用不可になっている場合は、その条件はほとんど発生しませんでした。新しいコンパイラーでは、条件を使用不可にすると、その条件が発生しないことが表明されます。それでも、その条件は発生する可能性があります。

一部のコード・シーケンスでは、これによって、コンパイラーはより高速なコードを生成できます。 例えば、CHAR(1) の FIXED BIN への代入では、CONVERSION が使用可能になっている場合は、変換はライブラリーの呼び出しによって実行されます。 それが、CONVERSION が使用不可になっている場合は、変換は、CHAR(1) 値の左のニブルを「AND 演算でゼロにする」非常に単純なインライン・コードで実行されます。 このコードは、NOCONVERSION によって、変換条件がこのステートメントで発生することがないと表明することではじめて可能になります。 この表明が真ではない場合は、プログラムは無効になります。

© Copyright IBM Corp. 1999, 2016

ただし、CHAR(2) の FIXED BIN への代入では、変換は、(それらの 2 文字で保持できる候補が多すぎるため) 常にライブラリーの呼び出しによって実行され、NOCONVERSION が有効になっている場合でも、ソースに有効な数値が含まれていなければ CONVERSION 条件が発生します。 (なお、CHAR(2) ソースに数字のみが含まれていることが分かっている場合は、EDIT 組み込み関数、またはソースに基づくか和集合演算されたものとして宣言された変数の適切なピクチャー・ストリングを使用して、このライブラリーの呼び出しを回避することもできます。)

同様に、SUBSCRIPTRANGE が使用不可の場合は、すべての添え字が有効であることが表明されます。 ほとんどのステートメントでは、添え字の妥当性を検査するコードがコンパイラーによって生成されず、無効な添え字が存在した場合にはプログラムでエラーが発生することになります。 ただし、PUT DATA ステートメントで添え字参照が使用された場合は、ライブラリー・ルーチンによってその参照が評価され、無効な添え字が存在した場合は、(使用不可になっていても) SUBSCRIPTRANGE 条件が発生します。

無効な RETURN

コードの断片が、値が返されるべきではないときに値を返そうと試みた場合、従来のコンパイラーでは条件は発生しませんが、新しいコンパイラーは ERROR 条件を発生させることでこのエラーのアラートを出します。

やや非常識ではあるが例としては分かりやすい、次のコード・フラグメントについて見てましょう。

```
call y;

x: proc returns( pointer );
  y: entry;
  return( sysnull() );
end;
```

Y においてプロシージャーが開始されたとき、どのような値も戻されるべきではないにもかかわらず、コードは値を戻そうとするため、このプログラム・フラグメントは間違っています。

従来のコンパイラーでは、無効な戻り値を返そうとした場合、どのような条件も意図的に発生させられず、プログラムでは、さまざまな形でエラーが発生することがありました (場合によっては「正常に」終了することさえありました)。

新しいコンパイラーでは、生成されたコードにより、 ONCODE=9004 が設定された ERROR 条件が意図的に発生させられます。

GOTO の欠点

GOTO の欠点がコードに見つかった場合、従来のコンパイラーでは保護例外が発生しますが、新しいコンパイラーでは ERROR 条件が発生します。

次のコード・フラグメントを見てください。

```
dcl x(4) label;

goto x(n);
x(4):;
put skip list( n );
x(3):;
```

```
put skip list( n );
x(2):;
put skip list( n );
x(1):;
put skip list( n );
```

n < 1 または *n* > 4 であり、かつ SUBSCRIPTRANGE 条件が有効になっていない場合、このプログラムは間違っています。

従来のコンパイラーでは、通常は記憶保護例外が発生しました。

新しいコンパイラーでは、ONCODE=9003 が設定された ERROR 条件が発生して、次のメッセージが表示されます。

```
IBM0751S  ONCODE=9003  A GOTO was attempted to an element of a label constant
          array, but the subscripts for the element were not those of any
          label in that array.
```

NOFOFL のスコープ

Enterprise PL/I では、FIXEDOVERFLOW/NOFIXEDOVERFLOW (または FOFL/NOFOFL) プレフィックスは、FIXED DECIMAL 演算にのみ適用されます。ただし、(NO)FOFL プレフィックスは、PROCEDURE または BEGIN ステートメントに適用されている場合、そのブロックと、その中に静的に含まれているブロックにのみ適用されます。このプレフィックスは、これらのブロック内から動的に呼び出される他のコードには適用されません。

同様に、(NO)FOFL プレフィックスが CALL ステートメントまたは関数呼び出しを含むステートメントに適用される場合、プレフィックスでの設定は、呼び出されたルーチン内のコードには適用されません。ルーチンの呼び出し前または後の、FIXED DECIMAL 計算にのみ適用されます。

例外が発生しなくなったコード

このセクションには、新しいコンパイラーにおいて例外を発生させないようになったコードに関する情報があります。

FIXED BIN について発生する FIXEDOVERFLOW

従来のコンパイラーでは、FIXED BIN 演算によって 31 ビットより多く必要な結果が得られた場合は、FIXEDOVERFLOW (FOFL) 条件が発生しました。新しいコンパイラーでは、どのような FIXED BIN 計算についても FOFL 条件は発生しません (ただし FIXED DEC 計算については、必要に応じて FOFL 条件が発生します)。

例えば、従来のコンパイラーでは、100_000 という値の FIXED BIN 変数を自乗すると、FOFL 条件が発生しました。

この新しいコンパイラーでの変更により、PL/I 言語が C 言語や Java 言語に適合したものになるとともに、コンパイラーは、8 バイト整数の加算と減算を実行するインライン・コードを生成することができるようになります。

実際は、すべてのコードが C コンパイラーまたは新しい PL/I コンパイラーを使用してコンパイルされている場合、ランタイムの初期化時には、整数の FOFL を有効にする PSW 内のビットは設定されません。 このビットは、メイン・モジュール

内に従来の PL/I コードが含まれている場合にオンに設定され、これにより、新しいコードの一部のパフォーマンスが低下することがあります。

数値変数にブランクを代入する際に生じる CONVERSION

従来のコンパイラーと新しいコンパイラーとの間で、数値変数へのブランクの代入における動作が異なります。

従来のコンパイラーでは、1 つ以上のブランクで構成されている文字ストリング (他の文字は何もない) が数値変数に代入された場合に、CONVERSION (または CONV) 条件が生じました。 しかし (長さゼロの文字ストリングを、ブランクのみで構成される文字ストリングと同じであるとみなす場合であっても)、数値変数に対して長さゼロの可変長文字ストリングが代入された場合には、CONVERSION は生じませんでした。

新しいコンパイラーでは、ブランク・ストリングと同じであると見なされる文字ストリングを数値変数に代入しても、CONVERSION 条件は生じません。

過度に大きな集合体をマッピングした際に生じる ERROR

ご使用のコードで調節可能エクステントを使用して集合体を宣言する場合、そのサイズは実行時に決定されます。 サイズが 2G より大きく、コンパイラーが変数をマップするためにライブラリー・ルーチンへの呼び出しを生成すると、ERROR 条件が生じます。

しかし調節可能エクステントを使用した単純な集合体の場合、SIZE 条件が有効でない限りは、コンパイラーはインライン・コードを生成して変数のサイズを決定します。 変数のサイズが 2G より大きく SIZE が有効になっていない場合には、いかなる条件も生じずにプログラムは無効になります。 言うまでもなく、集合体のサイズが適度であれば、SIZE が無効な場合に比べてはるかにパフォーマンスが良くなります。

異なる方法でマップされるストレージ

このセクションでは、従来のコンパイラーと新しいコンパイラーとの間で、ストレージに関する違いがあるシチュエーションについて説明します。それらの違いは、プログラムがどのように動作するかに影響します。

1 バイトの FIXED BIN

変数が精度 7 以下の FIXED BIN として宣言された場合、その変数は PL/I for MVS & VM 以前の環境では 2 バイトのストレージを占有しましたが、 Enterprise PL/I の環境では 1 バイトのストレージを占有します。 変数が構造体の一部をなす場合は、通常、このために構造体のマップ方法が変わり、プログラムの動作に影響する可能性があります。

例えば、構造体がファイルから読み込まれる場合、 Enterprise PL/I で読み込まれるバイト数は、PL/I for MVS & VM または以前の PL/I リリースの場合より少なくなります。

この違いを回避するには、変数の精度を 8 から 15 まで (両端の値を含む) の範囲の値に変更できます。

この相違点のために問題が発生している場所を見つけるのに役立てるために、コンパイラーは、メッセージ IBM1044 を出して、精度が 7 以下の FIXED BIN にフラグを立てます。

DEFAULT コンパイラー・オプションの (NO)BIN1ARG サブオプションは、プロトタイプ化されていない関数に渡される 1 バイトの REAL FIXED BIN 引数をコンパイラーが処理する方法を制御します。

- BIN1ARG の場合、コンパイラーはプロトタイプ化されていない関数に FIXED BIN 引数を現状のままで渡します。
- しかし NOBIN1ARG の場合、コンパイラーは、プロトタイプ化されていない関数に渡された 1 バイトの REAL FIXED BIN 引数を 2 バイトの FIXED BIN に一時的に代入してから、代わりにそれを渡します。

以下の例について見てみましょう。

```
dcl f1 ext entry;
dcl f2 ext entry( fixed bin(15) );

call f1( 1b );
call f2( 1b );
```

DEFAULT(BIN1ARG) を指定した場合、コンパイラーは 1 バイトの FIXED BIN(1) 引数のアドレスをルーチン f1 に渡し、2 バイトの FIXED BIN(15) 引数のアドレスをルーチン f2 に渡します。 しかし DEFAULT(NOBIN1ARG) を指定した場合には、コンパイラーはどちらのルーチンに対しても、2 バイトの FIXED BIN(15) 引数のアドレスを渡します。

ルーチン f1 が COBOL ルーチンの場合、そのルーチンに対して 1 バイトの整数引数を渡すと、COBOL は 1 バイトの整数をサポートしていないため問題が生じることに注意してください。 この場合、DEFAULT(NOBIN1ARG) を使用するのも有用ですが、エントリー宣言で引数属性を指定する方が良いでしょう。

それで BIN1ARG がデフォルトのサブオプションですが、互換性を増すためには NOBIN1ARG サブオプションを指定する方が助けになる場合もあります。

異なる方法で処理される宣言

このセクションには、従来のコンパイラーと新しいコンパイラーとの間で処理が異なる宣言に関する情報があります。

INITIAL 属性を持つ AREA

新しいコンパイラーは、AREA の INITIAL 属性を無視します。このため、AREA の INITIAL 文節は代入ステートメントに変換する必要があります。

例えば、次のコード・フラグメントの中で、配列のエレメントは a1、a2、a3、および a4 に初期化されません。

```
dcl (a1,a2,a3,a4) area;
dcl a(4) area init( a1, a2, a3, a4 );
```

ただし、コードを次のように書き換えれば、配列は初期化されます。

```
dcl (a1,a2,a3,a4) area;
dcl a(4) area;

a(1) = a1;
a(2) = a2;
a(3) = a3;
a(4) = a4;
```

コンパイラーは、メッセージ IBM1196 を出力して、INITIAL 属性を持つ AREA の宣言にフラグを立てます。

異なる方法で処理される変換

このセクションには、従来のコンパイラーと新しいコンパイラーとの間で結果が異なる可能性がある変換に関する情報があります。

FLOAT から文字への変換

FLOAT (BIN または DEC) から CHARACTER に変換する場合、新旧のコンパイラーで生成される結果には、最後の数字に違いが生じることもあります。

この違いは、基礎となる浮動小数点値または結果を得るための計算に違いがあることを示しているわけではありません。 通常、この違いを無視しても問題ありません。

スケール付き FIXED BINARY からの変換

通常、スケール付き FIXED BINARY を使用するとコンパイラーは非効率的なコードを生成するので、使用しないのが最善と言えます。 さらに、スケール付き FIXED BINARY から FIXED DECIMAL に変換する場合、新しいコンパイラーが生成する結果は従来のコンパイラーの結果とは異なる (より正確) 場合があります。

例えば、次のコードについて考えてみます。

```
dcl i fixed bin(15) init(290);
dcl s fixed bin(31,16);
dcl d fixed dec(15,12);

d = i / 365;
put skip data( d );
s = i / 365;
d = s;
put skip data( d );
```

従来のコンパイラーでは、2 つの PUT ステートメントの結果は次のようになります。

```
D= 0.794509887700;
D= 0.794509887700;
```

新しいコンパイラーでは、2 つの PUT ステートメントの結果は次のようになります。

```
D= 0.794509887695;
D= 0.794509887695;
```

上記の 2 番目の代入ではスケール付き FIXED BIN から FIXED DEC への変換が明らかに関係していますが、最初の代入にも、式計算の PL/I 規則では式 *i / 365* の属性は FIXED BIN(31,16) ですので、そのような変換がかかわっています。

ここで生じていることを理解するには、除算の結果が代入された後の変数 *s* の内容を考察すると役立ちます。 その後、*s* は 16 進値 0000CB65 を保持します。 FIXED BIN(31,0) 数値として見ると、値 52069 となりますが、スケール因数 16 があるので、値 52069/2**16 を表します。 この値は、数学的には 52069*5**16/10**16 と等しくなります。 ですから、底 2 から底 10 に変換するには、コンパイラーはこの値を 5**16 (または 152587890625) で乗算します。 そのようにして、スケール因数 16 の FIXED DEC 値が生成されます。ですからスケール因数 12 のターゲット結果を生成するには、最後の 4 桁が除去されます。

電卓で検証できるように 52069 の 5**16 倍は 7945098876953125 で、新しいコンパイラーで生成される結果では最後の数桁が除去されます。

従来のコンパイラーで生成される結果が異なるのは、生成されるコードが *s* を 152587890625 ではなく 152587890626 によって乗算するためです。このようにすると、結果の正確性が低くなります。

この問題を完全に回避するには、小数部の伴う結果が生じる可能性のあるすべての除法を 10 進法で必ず実行するようにします。これを実行する簡単な方法の 1 つは、DECIMAL 組み込み関数を使用することです。 例えば前述の最初の代入で、式 *i / 365* を *dec(i) / 365* に変更すると、代入の結果は 0.794520547945 となります。

上記のような状況に関して警告するため、コンパイラーはスケール付き FIXED BIN から FIXED DEC への変換を検出すると、メッセージ IBM2810I を発行します。

異なる方法で処理される組み込み関数

このセクションには、従来のコンパイラーと新しいコンパイラーとの間で、サポートに違いがある組み込み関数に関する情報があります。

スケール係数および FIXED BIN を持つ算術組み込み関数

RULES(IBM) コンパイル時オプション (デフォルト) を指定した場合、ゼロ以外のスケール因数を持つ FIXED BIN として変数を宣言できます。スケールされた FIXED BIN による 2 項演算、プレフィックス演算、および比較演算は、従来のコンパイラーと同じセマンティクスを使用して実行されます。ただし、 ADD、DIVIDE、または MULTIPLY 組み込み関数は、ゼロ以外のスケール因数を持つ FIXED BIN の結果は生成しません。

これらの組み込み関数は、次のどちらかの条件が満たされる場合、従来のコンパイラーでは FIXED BIN として評価されていましたが、新しいコンパイラーでは FIXED DEC として評価されます。

- これらの組み込み関数の引数はゼロ以外のスケール因数を持つ FIXED BIN である。
- これらの組み込み関数の引数はゼロのスケール因数を持つ FIXED BIN であるが、第 4 の引数としてゼロ以外の値が指定されている。

例えば、新しいコンパイラーは、次の代入ステートメントの DIVIDE 組み込み関数を FIXED DEC 式として評価します。

```
dcl (i,j) fixed bin(15);
dcl x     fixed bin(15,2);

...

x = divide(i,j,15,2);
```

この例では、結果は FIXED DEC(15,2) ではなく FIXED DEC(6,1) であることに注意してください。 一般的なケースでは、(p,q) の結果は (t,s) であり、t = 1 + ceil(p/3.32)、s = ceil(q/3.32) です。 属性 FIXED DEC(p,q) を持つ結果を得るには、DECIMAL 組み込み関数をすべての FIXED BIN 引数に適用してください。したがって、この例では、式は DIVIDE(DEC(X), DEC(Y) 15, 2) となります。PRECTYPE コンパイラー・オプションを使用して、コンパイラーが精度を解釈する方法を変更することもできますが、これにより他のステートメントの解釈も変更してしまう可能性があります。

コンパイラーは、メッセージ IBM1053 を出して、この相違点にフラグを立てます。

DBCS 文字ストリングの変換用ストリング処理組み込み関数

Enterprise PL/I コンパイラーでは引き続き CHAR 組み込み関数がサポートされていますが、現在では CHAR は CHARACTER の省略形と見なされます。CHAR 組み込み関数の結果は、最初の引数に GRAPHIC 型がある場合を除いては、旧 PL/I for MVS & VM コンパイラーでの結果と同じです。

- PL/I for MVS & VM コンパイラーでは、結果はシフト・コードで囲まれたその引数のバイト値でした。
- Enterprise PL/I コンパイラーでは、結果は GRAPHIC ストリングから CHARACTER への変換によって生成されたストリングです。変換できない場合は、CONVERSION 条件が発生します。

例えば、X が GRAPHIC(3) であり、バイト .A.B.C を保持している場合、結果は次のようになります。

- PL/I for MVS & VM コンパイラーでは、GRAPHIC(X) は <.A.B.C> になります。
- Enterprise PL/I コンパイラーでは、GRAPHIC(X) は ABC になります。

例

次の例は、旧コンパイラーによって生成された結果を得るためのコードの変更方法を示しています。

例 **1**

```
add
    dcl so char(1) value ('0e'x), si char(1) value('0f'x);
then replace
    A = CHAR(X);
by
    UNSPEC(A) = UNSPEC(SO) || UNSPEC(X) || UNSPEC(SI);
```

例 **2**

```
replace
    CHAR(X)
by
    OLDCHAR(X)
where OLDCHAR is defined by
    oldchar: proc(x) returns( char(32767) var );
      dcl x graphic(*);
      dcl a char(32767) var;
      dcl d char(2*length(x));
      a = '0e'x;
      unspec(d) = unspec(x);
      a = a || d;
      a = a || '0f'x;
      return( a );
    end;
```

CHARACTER 組み込み関数と CHARGRAPHIC 組み込み関数について詳しくは、「*Enterprise PL/I for z/OS* プログラミング・ガイド」を参照してください。

MACRO プリプロセッサーの相違点

このトピックでは、新旧の MACRO プリプロセッサーの相違に応じて、コードの変更が必要になる状況について説明します。

マクロ・プリプロセッサーとストリング

従来のコンパイラーでは、マクロ・プリプロセッサーはストリング内やコメント内で囲まれているテキスト以外はすべて大文字になりました。 けれども従来のコンパイラーでは '...' で区切られたテキストだけがストリングとして認識され、 "..." で区切られたテキストはストリングとしては認識されず、大文字になりました。

新しいコンパイラーでもデフォルトのプリプロセッサー・オプション CASE(UPPER) を指定すると、ストリング内やコメント内で囲まれたテキスト以外はすべて大文字になります。 しかし新しいコンパイラーでは、'...' および "..." のどちらで区切られたテキストもストリングとして認識され、どちらも大文字にはなりません。

SQL プリプロセッサーの前にマクロ・プリプロセッサーを実行し、SQL ステートメントに以下のようなコードがある場合には、この違いが問題を引き起こす可能性があります。

```
WHERE "system" = 'Wilmer'
```

従来のコンパイラーでは、これは `WHERE "SYSTEM" = 'Wilmer'` になりました。

新しいコンパイラーでは、`WHERE "system" = 'Wilmer'` になります。

おそらく、後者の場合は DB2 で生成したい結果とはならないでしょう。 その場合には、(前処理の前に) ソースを変更して、"..." で区切られたテキストを大文字にする必要があります。

SQL プリプロセッサーの相違点

新旧の SQL プリプロセッサーの相違に応じて、コードの変更が必要になる場合があります。

Enterprise PL/I for z/OS V4R2 以降、SQL プリプロセッサーは LOB オプションをサポートしていません。プログラムがプリプロセッサーによる LOB 宣言の変換方法に依存している場合は、それを変更する必要があります。 この点および他の SQL プリプロセッサーの変更に関する詳細情報については、 159 ページの『Enterprise PL/I V4R1 からのマイグレーション』を参照してください。

第 15 章 新しいオブジェクトのリンク

この章では、新しい Enterprise PL/I コンパイラーによって作成されたオブジェクト・モジュールをリンク・エディットする際の考慮事項について説明します。

コードのリンクについて詳しくは、「*Enterprise PL/I for z/OS* プログラミング・ガイド」を参照してください。

プリリンカーと PDSE に関する考慮事項

Enterprise PL/I のデフォルトのコンパイラー・オプション NORENT と LIMITS(EXTNAME(7)) を使用している場合は、プリリンカーも PDSE も使用する必要はありません。

AMODE(24) に関する考慮事項

AMODE(24) をサポートするには、SCEELKED の前に連結された SIBMAM24 に、アプリケーション・プログラムをリンクする必要があります。

AMODE(24) アプリケーションのビルドについて詳しくは、 78 ページの『AMODE(24) の制約事項』を参照してください。

PLICALLA または PLICALLB エントリーの使用

PLICALLA または PLICALLB を Enterprise PL/I プログラム内のメインエントリー・ポイントとして使用する場合は、SCEELKED の前に SIBMCAL2 を連結する必要があります。

CHANGE カード

RENT オプションが指定されている場合、または LIMITS(EXTNAME(*n*)) オプションで 8 よりも大きい *n* が指定されている場合、Enterprise PL/I は、リンク・エディットの際の CHANGE カードの使用をサポートしません。

© Copyright IBM Corp. 1999, 2016

第 16 章 言語環境プログラムの新しいコンパイラーとの使用

言語環境プログラムのランタイムを Enterprise PL/I と一緒に使用する場合、言語環境プログラムと従来の OS PL/I ランタイムとの違いについて知っておいてください。

37 ページの『第 6 章 マイグレーション前の考慮事項』で説明した考慮事項の多くは、同様に新しいコンパイラーにも当てはまります。 新しいコンパイラーを使用する際の、実行時の考慮事項について詳しくは、その章を参照してください。

113 ページの『第 13 章 作業コードを変更する必要がある場合について』にも、以前のバージョンの PL/I と Enterprise PL/I のランタイム結果の相違に関する有用な情報が記載されています。

適切なランタイム・オプションの使用

言語環境プログラムでは、OS PL/I ランタイムでは使用可能なオプションの一部が使用できなくなり、また一部は名前変更、再定義、または他のオプションとマージされました。 また、いくつかの重要な新しいオプションが使用できるようになりました。

除去されたオプション

- COUNT
- FLOW

名前変更されたオプションとマージされたオプション

- HEAP により HEAP が再定義されました。
- LANGUAGE は NATLANG に置き換えられました。
- REPORT は RPTSTG に置き換えられました。
- ISASIZE と ISAINC は STACK にマージされました。
- SPIE と STAE は TRAP にマージされました。

重要な新しいオプション

- ABTERMENC
- ALL31
- DEPTHCONDLMT
- ERRCOUNT
- MSGFILE
- STORAGE
- XUFLOW

ランタイム・オプションについて詳しくは、「*z/OS* 言語環境プログラム プログラミング・リファレンス」を参照してください。ただし、次の重要事項に注意してください。

- OS PL/I との互換性を得るには、次のオプションを使用します。
  - ABTERMENC(RETCODE)
  - DEPTHCONDLMT(0)

© Copyright IBM Corp. 1999, 2016

- ERRCOUNT(0)
- TRAP(ON)
- XUFLOW(ON)

- AMODE(24) アプリケーションでは、次のオプションを指定する必要があります。
  - ALL31(OFF)
  - STACK(,,BELOW)
- パフォーマンスが重要なアプリケーションでは、決して RPTSTG(ON) を使用してはいけません。
- パフォーマンスが重要なアプリケーションでは、決して STORAGE(,,00) を使用してはいけません。
- マルチスレッド・アプリケーションでは、POSIX(ON) を指定する必要があります。

アセンブラーのメインプログラムからの PL/I の呼び出し

言語環境プログラム準拠のアセンブラー・ルーチンが Enterprise PL/I サブルーチンに制御を渡す方法には、3 とおりがあります。

1. 静的にリンクした Enterprise PL/I サブルーチンにブランチする。
2. 言語環境プログラムのマクロ CEEFETCH を使用して、別個にリンクした Enterprise PL/I サブルーチンにブランチする。
3. LOAD や BALR などのアセンブラー命令を使用して、別個にリンクした Enterprise PL/I サブルーチンにブランチする。

   この場合は、言語環境プログラムと PL/I に固有のランタイム環境を確実に初期化するために、言語環境プログラムと Enterprise PL/I のシグニチャー CSECT である CEESG011 で明示的にリンクする必要があります。

アセンブラーに関するその他の問題について詳しくは、 51 ページの『アセンブラー・サポートにおける相違点』を参照してください。

結果が異なる可能性がある場合について

言語環境プログラムでは、一部の戻りコード、ランタイム・メッセージ、ダンプ、およびストレージ報告書が、従来の OS PL/I ランタイムにおいて受け取るものとは異なります。さらに、Enterprise PL/I は、言語環境プログラム提供の (OS PL/I の場合よりも精度の高い) 数学組み込みルーチンを起動するようになりました。

戻りコード

PLIRETC 組み込みサブルーチンは FIXED BIN(31) 引数を受け入れるようになったため、値は < 999 以下である必要はありません。

これに対応して、PLIRETV 組み込み関数は FIXED BIN(31) 値を戻すようになりました。

言語環境プログラムのランタイムにより、重大度 3 の条件を表すためにユーザー戻りコードに 3000 が追加され、言語環境プログラムでは、以下を除くすべての PL/I 条件が重大度 3 として分類されます。

- ATTENTION (SIGNAL ステートメントについて発生した場合)
- CONDITION
- ENDPAGE
- FINISH
- NAME
- PENDING
- STRINGRANGE
- STRINGSIZE
- UNDERFLOW

ランタイムにメッセージが発行される場合

言語環境プログラムを使用している場合は、ON ユニットが設定されている条件について、いくつかのランタイム・メッセージが発行されるタイミングが少し異なります。

- 言語環境プログラムを使用していない場合は、ZERODIVIDE または ERROR などの条件が発生すると、ランタイムは、この条件の ON ユニットを呼び出す前にメッセージを発行します。
- 言語環境プログラムを使用している場合は、ZERODIVIDE または ERROR などの条件が発生すると、ランタイムは、ON ユニット内の END ステートメントが実行される場合にのみメッセージを発行します。

この変更により、ランタイムが独自のメッセージを発行することなく、ユーザーが条件を処理して (また希望に応じてユーザー独自のメッセージを発行)、GOTO によりアプリケーションを続行できるようになります。

ON ユニットがない場合は、ランタイムの動作に変更はありません。

また、ERROR ON ユニットが制御を受け取るまで、ON ERROR SNAP によって生成される SNAP トレースバック・メッセージが継続して出されます。

Enterprise PL/I プログラムを言語環境プログラムの下で実行している場合は、ファイル入出力エラーが OPEN 処理中に検出されるようになり、その結果、異なるけれどもさらに意味のあるエラー・メッセージやエラー・コードが表示されます。結果的にエラーによって、TRANSMIT または従来の PL/I で受け取った他の条件ではなく、UNDEFINEDFILE 条件が生じます。

ランタイム・メッセージの意味

PL/I for MVS & VM と Enterprise PL/I では同じランタイム・メッセージのセットが共用されているため、よく理解して柔軟に解釈する必要のあるメッセージが発行されることがあります。

例えばランタイムが、Enterprise PL/I プログラム内の UNDEFINEDFILE についてメッセージを発行した場合、Enterprise PL/I では現在 VM はサポートされていませんが、このメッセージでは、MVS と VM の両方の構成が示されます。しかしその意味は明らかです。

また、GONUMBER コンパイラー・オプションを指定してコンパイルした場合、ランタイム・メッセージでは、例外が発生した「ステートメント」が示されます。Enterprise PL/I では、この「ステートメント」は、例外を発生させたステートメントのソース・プログラム内の行番号です。

最終的に、言語環境プログラム・ランタイムにおけるランタイム・メッセージの形式と内容は、OS PL/I ランタイムの場合と異なります。ランタイム・メッセージについての詳細な説明は、「*z/OS* 言語環境プログラム ランタイム・メッセージ」に記されています。

ランタイム・メッセージの出力先

言語環境プログラムでは、ランタイム・メッセージは、ランタイム・オプション MSGFILE で指定された宛先に出力されます。

MSGFILE のデフォルトの宛先は SYSOUT であり、従来のランタイムのデフォルトであった SYSPRINT ではありません。Enterprise PL/I では、ランタイム APAR PQ78307 用の PTF を適用した後でのみ、MSGFILE(SYSPRINT) がサポートされます。

数学組み込み関数

新しいコンパイラーは、数学組み込み関数 (SIN または COS など) を評価するため、および浮動小数点指数のために、言語環境プログラムで用意されているルーチンを呼び出します。 これらのルーチンは、OS PL/I V2R3 ライブラリーで用意されているルーチンよりも正確であるため、最後の桁が異なる結果を生成することがあります。

この違いの例として、三角法の教科書の巻末に記載されているような表を作成する次のプログラムについて見てみましょう。

```
trigtab: proc options(main);

  dcl degrees   fixed dec(5,1);
  dcl minutes   fixed dec(3,1);

  do degrees = 0 to 359;
    put skip edit( degrees ) ( f(5) );
    do minutes = 0 to .9 by .1;
      put edit( sind(degrees+minutes) ) ( f(9,4) );
    end;
  end;

end;
```

このプログラムの出力は、次のようになります。

```
  0   0.0000   0.0017   0.0035   0.0052   0.0070   ...
  1   0.0175   0.0192   0.0209   0.0227   0.0244   ...
```

作成される表は、使用する数学ライブラリーによって異なりますが、その場合でも 5 種類の値しかありません。 例えば、言語環境プログラムより前の数学ライブラリーを使用している従来のコンパイラーでは、140.1 の結果は 0.6414 になりますが、言語環境プログラムの数学ライブラリーを使用している従来のコンパイラーでは、

結果は 0.6415 になります。 新しいコンパイラーでは言語環境プログラムの数学ライブラリーのみが使用されるため、新しいコンパイラーでも結果は 0.6415 になります。

ダンプ

現在でも PLIDUMP を呼び出すとダンプが生成されますが、ダンプの形式、内容、および宛先は言語環境プログラムによって制御されるようになりました。

この結果生じる多くの相違点 (ほとんどは小さなものですが) について詳しくは、48 ページの『PLIDUMP の相違点』を参照してください。

ストレージ報告書

ランタイム・ストレージ報告書の形式、内容、および宛先が変更されました。

ランタイム・ストレージ報告書について詳しくは、「*z/OS* 言語環境プログラム プログラミング・リファレンス」での RPTSTG オプションに関する説明を参照してください。

言語環境プログラムでは、ランタイム・ストレージ報告書の見出しを指定するために、PLIXHD 宣言は使用されません。 ただし、言語環境プログラムの呼び出し可能サービス CEE3RPH を使用して見出しを指定することができます。

前提条件言語環境プログラム PTF

このトピックには、Enterprise PL/I を使用して PL/I アプリケーションをコンパイルおよび実行するために必要な PTF がリストされています。

- z/OS バージョン 1 リリース 4 の場合: PTF UQ70042 (APAR PQ66155) および PTF UQ88264 (APAR PQ88268)。
- z/OS バージョン 1 リリース 5 の場合: PTF UQ80236 (APAR PQ78173) および PTF UQ88263 (APAR PQ88065)。
- z/OS バージョン 1 リリース 6 の場合: PTF UQ92073 および UQ92088 (APAR PQ92870 および PQ93118)。
- z/OS バージョン 1 リリース 7 の場合: PTF UK06652 (APAR PK10630)。

第 17 章 CPU とストレージの使用効率を向上させるためのチューニング

言語環境プログラムへのマイグレーション後、パフォーマンスを最大限に引き出すために、アプリケーションの再チューニングを行う必要があります。 CPU とストレージの効率を同時に最大化できるとは限りません。 CPU の効率を高めるためにストレージの使用量を増やす必要が生じることはよくあり、その逆も同様です。この章では、言語環境プログラムの環境でアプリケーションを再チューニングするために役立つ一般的なヒントを示します。

パフォーマンスを高めるためのコンパイラー・オプションの選択について詳しくは、88 ページの『パフォーマンスを向上させるためのオプションの選択』を参照してください。

アプリケーションのパフォーマンスを向上させるために使用できるツールについて詳しくは、「*z/OS* 言語環境プログラム プログラミング・ガイド」、「*z/OS Language Environment Installation and Customization under OS/390*」、または「*z/OS Language Environment* カスタマイズ」、および「*Enterprise PL/I for z/OS* プログラミング・ガイド」を参照してください。

CPU 使用効率の向上

このトピックには、優れた CPU 使用効率を実現するために役立つヒントがあります。

- 言語環境プログラムによって実行される GETMAIN と FREEMAIN の数を削減します。

  ストレージ報告書を作成するには、言語環境プログラムの RPTSTG(ON) オプションを使用します。 対応する言語環境プログラムのストレージ・ランタイム・オプションに、報告されるストレージの量を指定します。
- 言語環境プログラムによって実行される LOAD と DELETE の数を削減します。

  通常使用される言語環境プログラムのライブラリー・ルーチンは、(E)LPA 内に配置します。 下のリストは、PL/I 用の推奨候補を示しています。
  - CEEBINIT (LPA)
  - CEEPLPKA (ELPA)
  - CEEEV010 (ELPA) (現在も OS PL/I アプリケーションを使用している場合)
  - CEEEV011 (ELPA) (Enterprise PL/I アプリケーション用)
  - CEEBLIIA (LPA) (再リンクされていない OS PL/I アプリケーション用)
  - IBMRLIB1 (LPA)

  (E)LPA 内に配置できるライブラリー・ルーチンの完全なリストについては、「*z/OS Language Environment Installation and Customization under OS/390*」または「*z/OS Language Environment* カスタマイズ」を参照してください。
- ライブラリー・ルーチン間での AMODE の切り替えを避けます。

© Copyright IBM Corp. 1999, 2016

言語環境プログラム ALL31(ON) オプションを指定できるように、可能ならばアプリケーションに対して AMODE(31) を使用してください。 ALL31(ON) を有効にすれば、ライブラリー・ルーチン間で AMODE の切り替えは行われません。

- PL/I 条件の過度の使用を避けます。

  すべての PL/I 条件処理は大きな負荷を発生させるため、適切な場合にのみ使用する必要があります。 PL/I 条件処理を使用し過ぎると、アプリケーションのパフォーマンスが低下します。

- DF/SMS に用意されている、システム決定の BLKSIZE を使用します。

  MVS の場合、ブロック化できる出力ファイルに対しては BLKSIZE(0) を使用します。 DF/SMS が最適なブロック・サイズを決定するので、これに従えばファイルのパフォーマンスを高めることができます。

- 言語環境プログラムのライブラリー・ルーチン保存機能 (LRR) を使用します。

  LRR を使用すれば、CPU パフォーマンスを高めることができます。 LRR を使用すると、アプリケーションの終了時に、言語環境プログラムはストレージ内の特定の言語環境プログラム・リソースを保持します。 ストレージに残っている言語環境プログラム・リソースが再利用されるので、 LRR を使用するプログラムの起動は大幅に高速化します。

  例えば、IMS/DC 環境で LRR を使用してパフォーマンスを向上できます。

  LRR を使用するとストレージ内に言語環境プログラム・リソースが長期間残るので、その状況に対応できるだけのストレージが使用できるかどうかを調べる必要があります。

ストレージ使用効率の向上

このトピックには、優れたストレージ使用効率を実現するために役立つヒントがあります。

- まだ OS PL/I プログラムを再コンパイルしていない場合は、言語環境プログラムを使用して再リンクします。

  再リンクした OS PL/I ロード・モジュールは、言語環境プログラムのスタブしか含まれていないためサイズが小さくなります。

- アプリケーションを AMODE(31) および RMODE(ANY) にします。

  ほとんどの場合、アプリケーションは 16M ラインより上にロードされます。 言語環境プログラムが制御ブロックの一部を 16M ラインより上に割り振ることができるようにするための、言語環境プログラム ALL31(ON) オプションを指定することができます。

- HEAPPOOLS(ON) オプションを使用しないようにします。

  HEAPPOOLS オプションは、(PL/I および旧来の PL/I for MVS & VM コードにではないものの) Enterprise PL/I に適用されます。HEAPPOOLS(ON) オプションを指定すると、たいへん多くのストレージが ANYHEAP に割り振られる結果になる場合があります。

- 可能ならば、言語環境プログラムの HEAP(,,ANY) オプションを使用します。

  PL/I では、次の条件が満たされる場合、言語環境プログラムはヒープ・ストレージを 16M ラインより上に割り振ります。
  - 要求発行者が AMODE(31) を使用している。
  - HEAP(,,ANY) が有効になっている。
  - メインプログラムが AMODE(31) を使用している。
- 可能ならば、言語環境プログラムの STACK(,,ANY) オプションを使用します。

  アプリケーションは AMODE(31) を使用している必要があります。 PL/I では、アプリケーションが言語環境プログラムを使用して再リンクされており、かつアプリケーションに編集されたストリーム入出力が含まれていない場合、言語環境プログラムは、スタック・ストレージを 16M ラインより上に割り振ります。
- IBM が提供している言語環境プログラムのストレージ・オプションのデフォルト値を分析し、必要に応じて可能ならば変更し、アプリケーションにとって最適にします。

  なお、小さい値を指定することが必ず良いとは限らないことに注意してください。小さい値を使用すると、言語環境プログラムが最初に割り振るストレージの量は少なくなりますが、その結果アプリケーションの実行期間中に GETMAIN と FREEMAIN が発行される回数が増える可能性があります。この場合、GETMAIN の負荷は非常に大きくなります。
- 通常使用される言語環境プログラムのライブラリー・モジュールは、(E)LPA 内に配置します。

  (E)LPA 内のライブラリー・ルーチンはアプリケーション領域のストレージを占有しないので、アプリケーションが使用できるストレージの量が多くなります。(E)LPA に入れることをお勧めするライブラリー・ルーチンについては、 145 ページの『CPU 使用効率の向上』を参照してください。

サブシステム環境でのパフォーマンス向上

このトピックには、特定のサブシステム環境において優れたパフォーマンスを実現するために役立つヒントがあります。

CICS 環境

EXEC CICS LINK の代わりに PL/I FETCH/CALL ステートメントを使用します。 PL/I の FETCH/CALL ステートメントのパスの長さは、EXEC CICS LINK のパスの長さよりもはるかに短いです。

IMS 環境

言語環境プログラムのライブラリー・ルーチン保存 (LRR) 機能を使用して、各トランザクションごとに言語環境プログラムによって実行される、LOAD/DELETE と GETMAIN/FREEMAIN の数を減らします。

共通して使用される言語環境プログラムのライブラリー・モジュール、および頻繁に使用されるトップレベル・アプリケーションをプリロードします。

とりわけ、入出力を伴うプログラムの場合は、モジュール IBMPOIOA をプリロードするか、IBMPOIOA を LPA に配置することは特に有益です。

第 18 章 既存の PL/I アプリケーションへの Enterprise PL/I プログラムの追加

既存のアプリケーションに Enterprise PL/I プログラムを追加する場合は、既存のプログラムを Enterprise PL/I で再コンパイルするか、または新たに記述した Enterprise PL/I プログラムをインクルードします。 既存のアプリケーションに Enterprise PL/I プログラムを追加すると、ユーザーのニーズに応じて、既存のプログラムを段階的にアップグレードすることができます。この章には、オブジェクトおよびロード・モジュールに関する考慮事項と、混合アプリケーションにおける条件処理に関する情報があります。

重要:

- 既存のアプリケーションに Enterprise PL/I プログラムを追加した場合、このアプリケーションは言語環境プログラムで実行する必要があります。
- 旧コードでマルチタスキングが実行されている場合は、新旧のオブジェクト・コードを混合することはできません。
- 新旧のコードを混合した場合は、FETCH からの FETCH は実行できません。

オブジェクト・モジュールおよびロード・モジュールに関する考慮事項

すべての PL/I ソースを再コンパイルするよう強くお勧めします。ただし、ソースを再コンパイルしない場合、また Enterprise PL/I コードと従来の PL/I オブジェクトを混用する場合、新しいコードのコンパイルに推奨されるオプションついて、さらに従来のオブジェクト・コードと新しいオブジェクト・コードを混用するときの制約事項について知っておく必要があります。

従来の PL/I オブジェクトと混用する Enterprise PL/I コードを再コンパイルするときには、以下のオプションを使用する必要があります。

- CMPAT(V2)

  注: 従来のコードが CMPAT(V1) でコンパイルされた場合、そのコードは CMPAT(V2) または CMPAT(V3) で再コンパイルする必要があります。
- LIMITS(EXTNAME(7))
- NORENT
- BACKREG(5)
- BIFPREC(15)

また、 81 ページの『第 11 章 新しいコンパイラーのオプションについて』 で説明したように、次のオプションのいくつかまたはすべてを使用することもできます。

- COMMON
- DEFAULT(LINKAGE(SYSTEM))
- DEFAULT(OVERLAP)

© Copyright IBM Corp. 1999, 2016

- EXTRN(FULL)
- NOWRITABLE(PRV)

NOWRITABLE(PRV) オプションを使用しないと、新コードと旧コードで CONTROLLED 変数を共用することはできません。

上記のオプションをすべて使用した場合でも、新旧のオブジェクト・コードの混合に関しては、次の制約事項があります。

- FILE 変数および定数は、1 つの例外を除いて新旧のコード間で共有できません。 SYSPRINT のみ、旧コードが LE でリンクされていれば新旧のコードで共有できます。 ただし、従来のコードによって書き出されたファイルを新しいコードによって読み込むこと、およびその逆は可能です。
- 従来のコードを使用するときは必ず、従来の製品からのフェッチ/リリースに関する制限がすべて適用されます。 特に、新しい MAIN が従来のモジュールの FETCH と CALL を正常に実行した場合、従来のモジュールは別のモジュールの FETCH を以後実行できなくなります。
- アプリケーション内に旧コードが存在する場合は、DLL コードを呼び出すことはできません。
- OS PL/I V1R4 オブジェクトと Enterprise PL/I オブジェクトを混合するサポートはありません。
- V2R3 以前 (ただし、V1R4 より後) の OS PL/I でコンパイルされた従来のコードについては、以下の制約事項が適用されます。
  - 言語環境プログラムを使用してリンクされていない従来の MAIN は、新しいモジュールの FETCH を実行できません。
  - 以下の条件下である場合を除き、新しい MAIN は、従来のモジュールに対して CALL も FETCH も実行できません。
    - 新しい MAIN 内にリンクされたシグニチャー CSECT CEESG010 がある。
    - 従来のモジュールが、APAR PK23270 用の PTF が適用された後で、または明示的な INCLUDE SYSLIB(CEESG010) によって、SCEELKED と再リンクされている。

これまで Enterprise PL/I では、旧コードで I/O 処理を行う場合、MAIN は旧コンパイラーでコンパイルしなければならないという制約事項がありました。言語環境プログラム 1.10 以降を使用している場合、この制約事項は現在では適用されません。

SYSPRINT の共用

相互に必要な APAR PK01919 (Enterprise PL/I) と PK016197 (PL/I for MVS & VM) によって出荷される機能拡張を使用すると、別プログラム・レベルおよび複数の別プログラム環境でも、SYSPRINT を Enterprise PL/I と PL/I for MVS & VM との間で共用することができます。

以下は、この共用 SYSPRINT がサポートする制限と範囲です。

- コンパイラー・オプション STDSYS を使用してはいけません。

- SYSPRINT は、EXTERNAL、STREAM、OUTPUT、または PRINT を、デフォルト属性または宣言された属性として持つ必要があります。
- 共用 SYSPRINT は、SYSOUT または永続データ・セットに対して指定することが可能でした。
- 共用 SYSPRINT は、MSGFILE(SYSPRINT) が指定されていて、混合環境に事前初期化済みのプログラムおよびストアード・プロシージャー、またはそのいずれかが存在しない場合にサポートされます。
- 複数の別プログラム環境では、オープンされた最初の SYSPRINT によって、SYSPRINT の属性が決定します。 2 番目以降の SYSPRINT は、最初の SYSPRINT からすべての属性を継承します。
- SYSPRINT は、アプリケーション全体でオープンされたままです。 処理終了時にのみクローズされる SYSPRINT を除いて、他のファイルはすべてエンクレーブ終了時にクローズされます。
- 共用 SYSPRINT の明示的なクローズが Enterprise PL/I または PL/I for MVS & VM のどちらかによって行われます。後で SYSPRINT に書き込もうとするには、SYSPRINT を明示的または暗黙的に再度オープンする必要があります。 2 回目のオープンで再使用されるデータ・セットへ SYSPRINT を経路指定した場合、これまで書き込まれたデータが失われる可能性があります。
- SYSPRINT は、初期スレッド (Enterprise PL/I マルチスレッド化) またはメインタスク (PL/I for MVS & VM マルチタスキング) によってのみ、(明示的または暗黙的に) オープンできます。 2 次スレッドおよびサブタスクは、明示的または暗黙的に SYSPRINT をオープンすべきではありません。また、明示的に SYSPRINT をクローズすべきではありません。
- TSO では、旧来の PL/I と SYSPRINT は共用できません。

共用 SYSPRINT のサポートに伴い、属性のオーバーライドが以下の点で変更されました。

- SYSPRINT が SYSOUT に経路指定されている場合、ENVIRONMENT オプションまたは OPEN ステートメントに指定された SYSPRINT 属性によって、DD ステートメントで指定された対応するオプションがオーバーライドされる。
- SYSPRINT がデータ・セット (TEMPORARY、NEW、または OLD) に経路指定されている場合、プログラムによって指定された属性と DD ステートメントで指定された属性が一致しなければ、UNDEFINEDFILE 条件が発生する。

マイグレーションを補助するために、APAR PK63659 で、新しい一時環境変数 PLI_SYSPRINT_ATTR_OVERRIDE が導入されました。 共用 SYSPRINT が変更される前と同じ動作を得るには、PARM パラメーターまたは PLIXOPT ストリングで、PLI_SYSPRINT_ATTR_OVERRIDE=YES を指定します。 これによって、SYSPRINT が TEMPORARY または NEW データ・セットに経路指定された場合に、属性のオーバーライドが許可されます。 なお、属性のオーバーライドは、SYSPRINT が既存または従来のデータ・セットに経路指定された場合には許可されず、SYSPRINT が SYSOUT に経路指定された場合には常に許可されます。

また、この新規環境変数のサポートは単に一時的なものであるということにも注意してください。 言語環境プログラム 1.10Z 以降では、この環境変数は無視されます。 影響を受けるプログラムおよび JCL は変更する必要があり、変更しなければ UNDEFINEDFILE 条件が発生します。

ランタイム・オプションの考慮事項

新 PL/I コードによって割り振られたストレージを旧 PL/I コードで解放しようとする場合には、新旧混合の PL/I コードで HEAPPOOLS オプションは使用できません。

条件処理に関する考慮事項

条件処理では、従来の PL/I プログラムと Enterprise PL/I プログラムとを「別々の言語」として見なす必要があります。

従来の PL/I と Enterprise PL/I のどちらにも、固有のシグニチャー CSECT (OS PL/I と PL/I for MVS & VM については CEESG010、VisualAge PL/I および Enterprise PL/I については CEESG011) があるとともに、言語環境プログラム内にそれぞれ別々のランタイム・ライブラリーが保持されています。

これはすなわち、一方の PL/I ソース・プログラム (従来のまたは新しい PL/I) でソフトウェア条件が発生して、もう一方の PL/I ソース・プログラム (発生元が従来の PL/I の場合は新しい PL/I、または発生元が新しい PL/I の場合は従来の PL/I) によって処理されると予期される場合、この例外を処理するはずのプログラムは、条件発生元のプログラムとはまったく別のランタイム・ライブラリーを使用しているため、これを検知することさえできないということです。

ハードウェア条件 (ZERODIVIDE など) が発生した場合は、言語環境プログラムによって 2 つの個別 PL/I ランタイム・ライブラリー間の差異が埋められるため、ソフトウェア条件の場合よりも、新旧の PL/I の境界を越えて適切に処理される可能性が高くなります。

PL/I ソース・プログラムの実行単位への区分化

新旧 PL/I モジュール間の混合と条件処理に関する制約事項に対応するためには、PL/I ソース・プログラムを実行単位に区分化する必要があります。

PL/I ソース・プログラムを実行単位に区分化する際には、十分に注意する必要があります。目的は、新旧の PL/I モジュールの混合に関するすべての制約事項を、定義する実行単位の範囲内に収めることです。 例えば、プログラム A で CONTROLLED EXTERNAL 変数が定義され、プログラム B で、この変数が参照されるとともに、プログラム C と共用されるファイル変数が作成される場合、これら 3 つのプログラム A、B、C が正常に動作するよう、これらをすべて Enterprise PL/I でコンパイルする必要があります。

最後に、新旧のコードを混合する場合は、新旧のコンパイラー間での、さまざまな言語構造体の処理方法の相違に注意する必要があります。詳しくは、 113 ページの『第 13 章 作業コードを変更する必要がある場合について』を参照してください。

第 19 章 旧リリースの Enterprise PL/I から Enterprise PL/I V5R1 へのマイグレーション

このマイグレーション・ガイド では、OS PL/I または PL/I for MVS & VM から Enterprise PL/I V5R1 にマイグレーションするときのマイグレーション作業に焦点を当てています。既に Enterprise PL/I バージョン 3 リリース、V4R1、V4R2、V4R3、V4R4、または V4R5 に移行している場合、Enterprise PL/I V5R1 には比較的容易にマイグレーションできます。

この章では、コンパイラー・オプションおよびコンパイラー・メッセージにおける相違点に焦点を当てますが、その他にコンパイラー出力に関していくつかの相違点があり、旧リリースの Enterprise PL/I のユーザーに影響を及ぼす可能性があります。

- コンパイラー自体は ARCH(8) でコンパイルされており、古いハードウェアを備えたマシンでコンパイラーを使用すると、コンパイラーが停止します。
- マクロ・プリプロセッサーが、コンパイラー・リストにコメントとして、%include、%xinclude、%inscan、および %xinscan を残すようになりました。
- 1 つのファイル内のリストには、行番号用に 7 つのカラムが組み込まれました。
- ブロックが使用する、AUTOMATIC ストレージのストレージ・オフセット (ブロックごと) 順のリストも、MAP 出力に組み込まれました。
- アセンブラー・リストのニーモニック・フィールド長が増加し、長いニーモニックを持つ新しい z/OS 命令をさらにサポートすることが可能になりました。
- より多くの右マージンが、属性、相互参照、およびメッセージ・リストで使用されます。
- コンパイラーが生成する SYSADATA に若干の変更点があります。
  - プロシージャー・レコードとその関連ステートメントのチェーニングが変更され、コンパイルのブロック構造を素早く判別できるようになりました (詳細は「プログラミング・ガイド」の付録に記載されています)。
  - エディション番号および sysadata レベル番号が更新されました (これらの値をコードで使用すると、プロシージャー・レコードの新旧両方のチェーニングを処理できます)。
- MAXNEST オプションを使用すると、DO、IF、または PROCEDURE ステートメントのネストが深すぎる一部の古いコードにフラグを立てることができます。

Enterprise PL/I V4R5 からのマイグレーション

V5R1 コンパイラーは、V3R9 コンパイラー、V4R1 コンパイラー、V4R2 コンパイラー、V4R3 コンパイラー、V4R4 コンパイラー、V4R5 コンパイラーと同様に、PDSE にインストールする必要があります。さらに、言語環境プログラムのランタイム・オプション XPLINK は、コンパイラーの始動時には必ず ON にする必要があります。IBMZPLI を使用してバッチで、または **pli** コマンドを使用して z/OS UNIX システム・サービス でコンパイラーを始動した場合は、コンパイラー自体が

© Copyright IBM Corp. 1999, 2016

必ず XPLINK(ON) で実行されます。ただし、それ以外の方法でコンパイラーを始動した場合は、ユーザーは XPLINK(ON) が有効であることを確認する必要があります。

IBMZIOP にあるオプション・ストリングが相互に同じである場合、またオーバーライド可能ではないオプションのいずれかと競合するオプションをユーザーが指定した場合、コンパイラーは警告メッセージを出すようになりました。

DSNH030 SQL プリプロセッサー・メッセージの重大度を簡単に変更できるよう、独自のプリプロセッサー・メッセージ IBM3317 が用意されています。

Enterprise PL/I V5R1 では、いくつかの新しいオプションが追加され、いくつかのオプションに新しいサブオプションが追加され、いくつかのオプションのデフォルトが変更されました。特に、RULES サブオプション LAXDCL のデフォルトが変更されたため、以前は戻りコード 0 で終了していた一部のコンパイルが、戻りコード 8 で終了するようになります。

Enterprise PL/I V5R1 における新規オプションと変更されたオプション

新規オプション

- BRACKETS
- DECOMP | NODECOMP
- LP (LP(32) がデフォルトです。)
- NULLDATE | NONULLDATE
- OFFSETSIZE (OFFSETSIZE(4) がデフォルトです。)

新規サブオプションがある既存のオプション

- DECIMAL の (NO)TRUNCFLOAT サブオプション
- INSOURCE の FIRST サブオプションおよび ALL サブオプション
- RULES の (NO)LAXSTMT サブオプションおよび (NO)UNREFBASED サブオプション
- RULES の NOPROCENDONLY(ALL|SOURCE) サブオプション
- XREF の EXPLICIT サブオプションおよび IMPLICIT サブオプション

デフォルトが変更された既存のオプション

オプション	**V5R1** でのデフォルト	**V4R5** でのデフォルト
ARCH	8	7
LIMITS	FIXEDDEC(15,31)	FIXEDDEC(15)
RULES	NOLAXDCL	LAXDCL
	NOLAXIF	LAXIF
	NOLAXSCALE	LAXSCALE
	NOMULTICLOSE	MULTICLOSE

除去されたサブオプションがある既存のオプション

- ARCH の 7 サブオプション

- FLOAT の AFP サブオプションおよび NOAFP サブオプション
- CMPAT の V1 サブオプションおよび LE サブオプション

除去されたオプション

- HGPR | NOHGPR

リリースを表すコンパイラー・オプション・デフォルト設定はすべて、PXOPT*vr* という名前のメンバーにリストされています。 *v* はバージョン番号、*r* はリリース番号で、例えば、PXOPT51 であれば V5R1、PXOPT45 であれば V4R5 です。このメンバーは SIBMZSAM データ・セットにあります。このデータ・セットには、サポートされている PL/I リリースすべての PXOPT*vr* メンバーが入っています。デフォルト設定における変更点と、あるリリースから次のリリースに追加された新規オプションを調べるには、2 つのリリースの PXOPT*vr* ファイルを比較してください。また、望ましい設定が入った独自のオプション・ファイルを作成するためのテンプレートとして、PXOPT*vr* ファイルを使用することもできます。

Enterprise PL/I V4R4 からのマイグレーション

V4R5 コンパイラーは、V3R9 コンパイラー、V4R1 コンパイラー、V4R2 コンパイラー、V4R3 コンパイラー、および V4R4 コンパイラーと同様に、PDSE にインストールする必要があります。さらに、言語環境プログラムのランタイム・オプション XPLINK は、コンパイラーの始動時には必ず ON にする必要があります。IBMZPLI を使用してバッチで、または **pli** コマンドを使用して z/OS UNIX システム・サービス でコンパイラーを始動した場合は、コンパイラー自体が必ず XPLINK(ON) で実行されます。ただし、それ以外の方法でコンパイラーを始動した場合は、ユーザーは XPLINK(ON) が有効であることを確認する必要があります。

Enterprise PL/I V4R5 には、新規オプションがいくつか含まれるほか、新規サブオプションを持つ旧オプションもいくつか含まれます。ただし、これらの新規オプション (CMPAT(V3) オプションと、LIMITS オプションの FIXEDBIN サブオプションを除く) のデフォルトでは、Enterprise PL/I V4R4 コンパイラーによって生成されるコードや、V3R3 以降のいずれかのリリースによって生成されるコードと互換性のある実行可能コードが生成されます。

以前のリリースでコンパイルされたコードや、CMPAT(V3) でコンパイルされたコードはすべて、再コンパイルする必要があります。

一部の変換においては、LIMITS(FIXEDBIN(M1,M2)) オプション内の M2 の値によって、ある程度、結果の属性が決定されます。そのため、このオプションのデフォルトを変更すると、別の結果を生み出すコードがコンパイラーによって生成される可能性があります。その例をいくつか紹介します。

1. FIXED DEC(p,q) から FIXED BIN に変換すると、ソースは、1 + MIN(CEIL(p*3.32), M2) と同じ精度を持つ FIXED BIN に変換されます。そのため、FIXED DEC(15) は、FIXED BIN に変換される場合に LIMITS(FIXEDBIN(31,31)) の下では FIXED BIN(31) に変換されます。しかし、これは LIMITS(FIXEDBIN(31,63)) の元では FIXED BIN(51) に変換されます。
2. FIXED DEC(p,q) から BIT に変換すると、ソースは、MIN(CEIL((p-q)*3.32), M2) と同じ精度を持つ FIXED BIN に変換されます。そのため、FIXED

DEC(11) は、BIT に変換される場合に LIMITS(FIXEDBIN(31,31)) の下では BIT(31) に変換されますが、LIMITS(FIXEDBIN(31,63)) の下では BIT(37) に変換されます。

3. BIT から FIXED BIN に変換すると、ソースは FIXED BIN(M2) に変換されます。

PL/I V4R4、V4R3、V4R2、V4R1、およびバージョン 3 の各リリースで使用したときと同じコンパイラー・オプション設定を PL/I V4R5 で使用すると、V4R5 でコンパイルされたコードと、それより前のリリースでコンパイルされたコードとを混在させることができます。プログラム・セマンティクスを変更するコンパイラー・オプションの設定を変更しない限り、すべてのコードを再コンパイルする必要はありません。 例えば、オブジェクトを混合する際には ARCH または RULES オプションを変更できますが、BACKREG、BIFPREC、または CMPAT オプションを変更する場合に、オブジェクトを混合することはできません。

現在、コンパイルに使用されるプログラムによっては、以下の理由により障害が発生する可能性があります。

- 現在、コンパイラーは、自らをフェッチしようとする FETCH ステートメントを持つすべてのコンパイル単位にフラグを立てます。
- 現在、コンパイラーは、プリプロセッサー・サブオプションが引用符で囲まれていないと失敗します。例えば、コンパイラーは、 PP(MACRO(CASE(ASIS))) が指定されると失敗しますが、正しく指定された PP(MACRO('CASE(ASIS)')) は受け入れます。

Enterprise PL/I V4R5 における新規オプションと変更されたオプション

新規オプション

- FILEREF | NOFILEREF
- MAXBRANCH

新規サブオプションがある既存のオプション

- RULES(NOLAXQUAL) オプションの FORCE サブオプション。
- RULES(NOLAXNESTED) オプションおよび RULES(NOPADDING) オプションの ALL サブオプションおよび SOURCE サブオプション。

Enterprise PL/I V4R3 からのマイグレーション

V4R4 コンパイラーは、V3R9、V4R1、V4R2、および V4R3 コンパイラーと同様に、PDSE にインストールする必要があります。さらに、言語環境プログラムのランタイム・オプション XPLINK は、コンパイラーの始動時には必ず ON にする必要があります。IBMZPLI を使用してバッチで、または **pli** コマンドを使用して z/OS UNIX システム・サービス でコンパイラーを始動した場合は、コンパイラー自体が必ず XPLINK(ON) で実行されます。ただし、それ以外の方法でコンパイラーを始動した場合は、ユーザーは XPLINK(ON) が有効であることを確認する必要があります。

Enterprise PL/I V4R4 には、いくつかの新規オプションと、新規サブオプションがあるいくつかの既存オプションがあります。ただし、これらの新規オプションおよ

びサブオプションのデフォルトでは、Enterprise PL/I V4R3 コンパイラーまたは V3R3 以降の任意のリリースによって生成されるコードと互換性のある実行可能コードが生成されます。

PL/I V4R3、V4R2、V4R1、およびバージョン 3 の各リリースで使用したものと同じコンパイラー・オプション設定を V4R4 で使用すれば、V4R4 でコンパイルされたコードと、それより前のリリースでコンパイルされたコードとを混在させることができます。プログラム・セマンティクスを変更するコンパイラー・オプションの設定を変更しない限り、すべてのコードを再コンパイルする必要はありません。 例えば、オブジェクトを混合する際には ARCH または RULES オプションを変更できますが、BACKREG、BIFPREC、または CMPAT オプションを変更する場合に、オブジェクトを混合することはできません。

Enterprise PL/I V4R4 における新規オプションと変更されたオプション

新規オプション

- INCLUDE | NOINCLUDE

新規サブオプションがある既存のオプション

- DEFAULT オプションの NULLSTRPTR サブオプションの STRICT サブオプション。

新規関数がある既存のオプション

- STMT オプションを指定すると、ソースおよびメッセージのリストに、論理ステートメント番号とソース・ファイル番号の両方が入ります。

戻りコードに対する変更

V4R4 以降、コンパイラーは、STATIC 属性を持たないものの INITIAL 項目が 100 個を超える変数の宣言にフラグを立てます。こうした宣言を含むプログラムのコンパイルは、戻りコード 0 で終了していましたが、戻りコード 4 で終了するようになりました。

CICS および SQL プリプロセッサーでは DFT(ASCII) コンパイラー・オプションをサポートしていませんでしたが、DFT(ASCII) コンパイラー・オプションの使用にフラグを立てて S レベルのメッセージを出すようになりました。この新規 S レベル・メッセージによって、コンパイルは戻りコード 12 で終了することになります。

SQL プリプロセッサーでは常に、WIDECHAR に CCSID 値 1200 が割り当てられます。

Enterprise PL/I V4R2 からのマイグレーション

V4R3 コンパイラーは、V3R9、V4R1、および V4R2 コンパイラーと同様に、PDSE にインストールする必要があります。 さらに、言語環境プログラムのランタイム・オプション XPLINK は、コンパイラーの始動時には必ず ON にする必要があります。IBMZPLI を使用してバッチで、または **pli** コマンドを使用して z/OS UNIX システム・サービス でコンパイラーを始動した場合は、コンパイラー自体が必ず

XPLINK(ON) で実行されます。ただし、それ以外の方法でコンパイラーを始動した場合は、ユーザーは XPLINK(ON) が有効であることを確認する必要があります。

Enterprise PL/I V4R3 には、新規オプションがいくつか含まれるほか、新規サブオプションを持つ旧オプションもいくつか含まれます。ただし、これらの新しいオプションおよびサブオプションのデフォルトでは、Enterprise PL/I V4R2 コンパイラー (または V3R3 以降の任意のリリース) によって生成されるコードと互換性がある実行可能コードが生成されます。

V4R2、V4R1、およびバージョン 3 の各リリースで使用したものと同じコンパイラー・オプション設定を PL/I V4R3 で使用すれば、V4R3 でコンパイルされたコードと、それより前のリリースでコンパイルされたコードとを混在させることができます。 プログラム・セマンティクスを変更するコンパイラー・オプションの設定を変更しない限り、すべてのコードを再コンパイルする必要はありません。 例えば、オブジェクトを混合する際には ARCH または RULES オプションを変更できますが、BACKREG、BIFPREC、または CMPAT オプションを変更する場合、それはできません。

Enterprise PL/I V4R3 における新規オプションと変更されたオプション

新規オプション

- CASERULES
- DEPRECATENEXT
- MSGSUMMARY

新規 **SQL** オプション

- ONEPASS | TWOPASS。 ONEPASS SQL オプションは V4R3 で復活しました。 このオプションは V4R2 では除去されていました。
- DEPRECATE

新規サブオプションがある既存のオプション

- ARCH オプションの 10 サブオプション
- DEPRECATE オプションの STMT サブオプション
- IGNORE オプションの ASSERT サブオプション
- RTCHECK オプションの NULL370 サブオプション
- RULES オプションの (NO)CONTROLLED、(NO)LAXNESTED、および (NO)RECURSIVE サブオプション
- RULES(NOUNREF) オプションの SOURCE | ALL サブオプション

除去されたサブオプションがある既存のオプション

- ARCH オプションの 5 サブオプション
- RULES オプションの (NO)STOP サブオプション。 DEPRECATE(STMT(STOP)) を使用して同じ機能を実行できます。
- XINFO オプションの XMI サブオプション

プリプロセッサー・メッセージ番号の変更

PL/I V4R3 では、CICS および SQL のバックエンド・エラーを報告するメッセージの番号が変更されました。 新たな番号を以下に示します。 これらのメッセージの重大度を変更するため、またはこれらのメッセージの出現を追跡するために EXIT オプションを使用している場合は、適切な更新を行ってください。

- IBM3000I I for informational messages from the back end.
- IBM3250I W for warning messages from the back end.
- IBM3500I E for error messages from the back end.
- IBM3750I S for severe messages from the back end.

Enterprise PL/I V4R1 からのマイグレーション

V4R2 のコンパイラーは、V3R8、V3R9、および V4R1 コンパイラーと同様に、PDSE にインストールする必要があります。 さらに、言語環境プログラムのランタイム・オプション XPLINK は、コンパイラーの始動時には必ず ON にする必要があります。IBMZPLI を使用してバッチで、または **pli** コマンドを使用して z/OS UNIX システム・サービス でコンパイラーを始動した場合は、コンパイラー自体が必ず XPLINK(ON) で実行されます。ただし、それ以外の方法でコンパイラーを始動した場合は、ユーザーは XPLINK(ON) が有効であることを確認する必要があります。

Enterprise PL/I V4R2 には、いくつかの新規オプションと、新規サブオプションがあるいくつかの既存オプションがあります。ただし、これらの新しいオプションおよびサブオプションのデフォルトでは、Enterprise PL/I V4R1 コンパイラー (または V3R3 以降の任意のリリース) によって生成されるコードと互換性がある実行可能コードが生成されます。

V4R1、およびバージョン 3 の各リリースで使用したものと同じコンパイラー・オプション設定を PL/I V4R2 で使用すれば、V4R2 でコンパイルされたコードと、それより前のリリースでコンパイルされたコードとを混在させることができます。プログラム・セマンティクスを変更するコンパイラー・オプションの設定を変更しない限り、すべてのコードを再コンパイルする必要はありません。 例えば、オブジェクトを混合する際には ARCH または RULES オプションを自由に変更できますが、BACKREG、BIFPREC、または CMPAT オプションを変更する場合に、オブジェクトを混合することはできません。

Enterprise PL/I V4R2 における新規オプションと変更されたオプション

新規オプション

- PPLIST
- UNROLL

新規サブオプションがある既存のオプション

- CHECK は (NO)STORAGE をサブオプションとしてサポートします。
- DEFAULT は (NO)PSEUDODUMMY をサブオプションとしてサポートします。

- RULES は、NOLAXENTRY(LOOSE | STRICT)、(NO)LAXRETURN、および (NO)SELFASSIGN をサブオプションとしてサポートします。

Enterprise PL/I V4R1 およびバージョン 3 との SQL プリプロセッサーの相違点

新しいコンパイラーと従来のコンパイラーとの間で、SQL プリプロセッサーに以下の違いがあることに注意してください。

除去された **SQL** プリプロセッサーのオプション

V4R2 コンパイラーでは、SQL プリプロセッサーの以下のオプションはサポートされなくなりました。

- LOB(DB2 | PLI)
- ONEPASS | TWOPASS

  プリプロセッサーは TWOPASS オプションがオンであると常時想定して機能するようになりました。
- SCOPE | NOSCOPE

  実際には、SCOPE は常時オンです。 ただし、SCOPE の使用時に以前は課されていた制約事項が除去されました。 SQL プリプロセッサーは、コンパイラーと同じ規則を使用して名前を解決するようになりました。

LOB 宣言の処理

SQL プリプロセッサーは LOB オプションをサポートしなくなりました。プログラムが SQL プリプロセッサーにより生成されたコード内での LOB の表現方法に依存している場合は、プログラムを変更する必要があります。

SQL TYPE は、他の PL/I データ属性が使用可能な任意の場所で使用できるようになりました。 これにより、プリプロセッサーによる LOB 宣言の変換方法に対する依存をコード内から除去して、さらに単純で分かりやすいコードを作成することができます。

例えば、以下の例の変数 XML_DOC_STRUC は、CLOB タイプの特定の実装に依存しています。 したがって、V4R2 の SQL プリプロセッサーを使用する場合、コンパイラーはこのコードをコンパイルできません。

```
DCL
  1 DOCM_STRUC,
    2 MODEL_EXECN_ID_STRUC  FIXED BIN(31),
    2 DOCM_TYPE_CD_STRUC    CHAR(1),
    2 XML_DOC_STRUC,
      3 XML_DOC_ARRY_LENGTH FIXED BIN(31),
      3 XML_DOC_ARRY_DATA,
        4 XML_DOC_DATA1(3)  CHAR(32767),
        4 XML_DOC_DATA2     CHAR(4099);

DCL MODEL_EXECN_ID_ARRY(5)  FIXED BIN(31);
DCL DOCM_TYPE_CD_ARRY(5)    CHAR(1);
DCL XML_DOC_ARRY(5)         SQL TYPE IS XML AS CLOB(100K);

EXEC SQL FETCH NEXT ROWSET FROM DOCM_CSR FOR 5 ROWS
          INTO  :MODEL_EXECN_ID_ARRY
               ,:DOCM_TYPE_CD_ARRY
               ,:XML_DOC_ARRY;
XML_DOC_STRUC = XML_DOC_ARRY(I);
```

SQL TYPE を使用して、前述の例のコードを以下のコードに変更することができます。コンパイラーはそれを V4R2 の SQL プリプロセッサーまたはプリコンパイラーを使用してコンパイルできますが、V4R1 以前のリリースのプリプロセッサーを使用してコンパイルすることはできません。

```
DCL
1 DOCM_STRUC,
  2 MODEL_EXECN_ID_STRUC  FIXED BIN(31),
  2 DOCM_TYPE_CD_STRUC    CHAR(1),
  2 XML_DOC_STRUC         SQL TYPE IS XML AS CLOB(100K);

DCL MODEL_EXECN_ID_ARRY(5)  FIXED BIN(31);
DCL DOCM_TYPE_CD_ARRY(5)    CHAR(1);
DCL XML_DOC_ARRY(5)         SQL TYPE IS XML AS CLOB(100K);

EXEC SQL FETCH NEXT ROWSET FROM DOCM_CSR FOR 5 ROWS
         INTO  :MODEL_EXECN_ID_ARRY
              ,:DOCM_TYPE_CD_ARRY
              ,:XML_DOC_ARRY;

XML_DOC_STRUC = XML_DOC_ARRY(I);
```

無効なホスト変数の参照

新しい SQL プリプロセッサーは、従来のプリプロセッサーではフラグを立てなかった以下の 2 種類の無効なホスト変数参照に、警告メッセージでフラグを立てます。

- 構造の配列、および配列を含む構造は、ホスト参照としては無効です。以下の宣言を例として考えてみます。

  ```
  dcl 1 A(<bounds>), 2 B <data-type>;
  dcl 1 A, 2 B (<bounds>) <data-type>;
  ```

  A をホスト参照として使用する場合、旧 SQL プリプロセッサーはこの参照を有効なものとして受け入れます。 しかし、新規 SQL プリプロセッサーはこれに警告メッセージでフラグを立てます。 この場合、参照を A.B と変更する必要があります。

- 構造を含む構造はホスト参照としては無効です。この宣言を例として考えてみます。

  ```
  dcl 1 X, 2 X, 3 Y <data-type>, 3 Z <data-type>, ... /* and no other level-2 items */
  ```

  X をホスト参照として使用する場合、旧 SQL プリプロセッサーはそれを有効なものとして受け入れます。 新規 SQL プリプロセッサーはこれに警告メッセージでフラグを立てます。この場合、参照を X.X と変更する必要があります。

SQL プリプロセッサー・メッセージの処理

Enterprise PL/I for z/OS V4R2 より前は、SQL プリプロセッサーのバックエンドで生成されたメッセージについては、機能 ID は SQL でした。IBM 提供のコンパイラー・ユーザー出口 (IBMUEXIT) を使用して、メッセージを抑止したり、メッセージの重大度を変更したりすることができました。

V4R2 以降は、すべてのメッセージの機能 ID は IBM であり、IBM 提供の IBMUEXIT を使用してこれらのメッセージの重大度を変更することはできません。

ただし、IBMUEXIT を変更して、DB2 メッセージの重大度を変更したり、それらを完全に抑止したりすることはできます。 これをどのように実行するかについて詳しくは、「*Enterprise PL/I for z/OS V4R2* プログラミング・ガイド」 で第 22 章『ユーザー出口の用法』を参照してください。

Enterprise PL/I バージョン 3 (すべてのリリース) からのマイグレーション

Enterprise PL/I V4R1 には、いくつかの新規オプションと、新規サブオプションがあるいくつかの既存オプションがあります。ただし、これらの新しいオプションおよびサブオプションのデフォルトでは、Enterprise PL/I V3R9 コンパイラー (または V3R3 以降の任意のリリース) によって生成されるコードと互換性がある実行可能コードが生成されます。

バージョン 3 の各リリースで使用したものと同じコンパイラー・オプション設定を PL/I バージョン 4 で使用すれば、V4R1 でコンパイルされたコードと、それより前のリリースでコンパイルされたコードとを混在させることができます。 プログラム・セマンティクスを変更するコンパイラー・オプションの設定を変更しない限り、すべてのコードを再コンパイルする必要はありません。 例えば、オブジェクトを混合する際には ARCH または RULES オプションを自由に変更できますが、BACKREG、BIFPREC、または CMPAT オプションを変更する場合に、オブジェクトを混合することはできません。

Enterprise PL/I V4R1 における新規オプションと変更されたオプション

新規オプションおよび変更されたサブオプションのリストは、次のとおりです。 これらについては「プログラミング・ガイド」で詳しく説明されています。

新規オプション

- DEPRECATE
- XREF

追加されたサブオプションがある既存のオプション

- ARCH は 9 をサブオプションとしてサポートします。
- GONUMBER は (NO)SEPARATE をサブオプションとしてサポートします。
- RULES は (NO)GLOBALDO および (NO)PADDING をサブオプションとしてサポートします。

除去されたサブオプションがある既存のオプション

- CHECK オプションの STORAGE サブオプション。
- LANGLVL オプションの SAA および SAA2 サブオプション。

除去された **SQL**

- (NO)OPTIONS。 SQL プリプロセッサー・オプションは常にリストされます。

Enterprise PL/I バージョン 3 リリースでの変更点

この情報では、Enterprise PL/I バージョン 3 の旧リリースに対して行われたいくつかの変更をリストしています。

Enterprise PL/I V3R9

- Enterprise PL/I V3R9 以降、DEFAULT オプションのデフォルト・サブオプションは ORDER ではなく REORDER になっています。
- また、V3R9 コンパイラーでは、COMPACT および TUNE オプションのサポートが除去されました。

Enterprise PL/I V3R8

Enterprise PL/I V3R8 以降、CMPAT の新しい V3 サブオプションのため、コンパイラーで生成されたいくつかのメッセージ挿入は、タイプ FIXED BIN(63) の 8 バイト整数になっています。この変更は、EXIT コンパイラー・オプションで起動されるユーザー独自のルーチンを作成していない限り、影響はありません。この場合、メッセージ挿入の可能なタイプの SELECT ステートメントがある場合は、恐らく、その SELECT ステートメントに新しい WHEN 節を追加する必要があります。

Enterprise PL/I V3R7

Enterprise PL/I V3R7 以降、以下の組み込み関数の資料は除去され、V3R8 以降はこれらの関数はサポートされなくなりました。

- ACOSF
- ASINF
- ATANF
- COSF
- EXPF
- LOGF
- LOG10F
- SINF
- TANF

Enterprise PL/I V3R6

V3R6 の場合のみ、CEESTART オプションのデフォルトは CEESTART(LAST) であることに注意してください。 このデフォルトにより、コンパイラーは CEESTART CSECT を、生成されたオブジェクト・デックの終わりに配置します。 これは、リンカー CHANGE カードを使用している場合は必要ですが、旧リリースのコンパイラーで行われたものとは異なります。

さらに、オブジェクトのバインド時に ENTRY CEESTART リンカー・カードを使用しない場合は、これによりコードが正しくない動作をします。CEESTART(FIRST) オプションの使用をお勧めします。

Enterprise PL/I V3R5

Enterprise PL/I V3R5 以降、PP オプションを複数回指定した場合のコンパイラー動作が変わりました。V3R5 より前には、最後の指定によって前の指定がすべて置き換えられましたが、V3R5 以降、オプションは (RULES

およびその他のオプションと同様に) 追加式になりました。したがって、PP(CICS) PP(SQL) を指定した場合は、PP(CICS SQL) を指定した場合と同じことになります。

V5R1 で導入されたメッセージ

このトピックでは、V5R1 で導入された新規メッセージや変更されたメッセージ (コンパイラー・メッセージとプリプロセッサー・メッセージの両方) について説明します。

新規および変更されたコンパイラー・メッセージ

V5R1 で導入された新規および変更されたメッセージを以下にリストします。これらのメッセージの多くは、特定のコンパイラー・オプションが有効である場合にのみ生成されます。 さらに詳しい総合的な説明については、「*Enterprise PL/I for z/OS* メッセージおよびコード」を参照してください。

- IBM2606: REFER において属性の選択が不十分であることを示す (IBM1045 に使用される)
- IBM2464: RULES(NOLAXSTMT) 違反を示す
- IBM2465: NULLSTRPTR オプションにおいて代入が無効であることを示す
- IBM2466: NULLSTRPTR オプションにおいて比較が無効であることを示す
- IBM2654: ADDR において INITIAL と BASED が一緒に使用されていることを示す
- IBM2655: オプションがオーバーライド可能オプションと競合しているかどうかを示す
- IBM2656: 単純 DEFINED のオカレンスを示す
- IBM2825: ライブラリー呼び出しによる BIT 変換を示す
- IBM2826: ライブラリー呼び出しによる BIT 変換を示す
- IBM2297: LP(64) なしで ALLOC31 が使用されていることを示す
- IBM2298: CHECK(STG) なしで ALLOCSIZE が使用されていることを示す
- IBM2299: BETWEEN において範囲が無効であることを示す
- IBM2300: SMF が使用不可であることを示す
- IBM2301: SMF の登録に問題があることを示す

新規および変更されたプリプロセッサー・メッセージ

V5R1 で導入された新規および変更されたメッセージを以下にリストします。さらに詳しい総合的な説明については、「*Enterprise PL/I for z/OS* メッセージおよびコード」を参照してください。

- IBM3317: DB2 メッセージ DSNH03 を示す
- IBM3605: 型定義が重複していることを示す
- IBM3606: 構造定義において ID が重複していることを示す
- IBM3607: 符号なし序数に負符号が付いていることを示す
- IBM3608: 精度付きの序数が、それらの値として小さすぎることを示す
- IBM3849: 未定義のタイプを示す

- IBM3850: タイプの定義の前に、そのタイプが使用されていることを示す
- IBM3851: 増加していない値が序数に使用されていることを示す
- IBM3852: 4 バイト整数として大きすぎる値が序数に使用されていることを示す
- IBM3857: 継続構造の説明を示す
- IBM3858: 名前なし序数が前もって定義されていないことを示す
- IBM3859: ストレージ属性を持つ構造定義を示す
- IBM3860: 構造の配列を指定する構造定義を示す
- IBM3950: アスタリスク反復因数の使用が無効であることを示す
- IBM3951: アスタリスク反復因数の使用が無効であることを示す
- IBM3952: マクロ・パラメーター付きで INITIAL が使用されていることを示す
- IBM3953: 初期値が多すぎる INITIAL 属性を示す

V4R5 で導入されたメッセージ

このトピックでは、V4R5 で導入された新規メッセージや変更されたメッセージ (コンパイラー・メッセージとプリプロセッサー・メッセージの両方) について説明します。

新規および変更されたコンパイラー・メッセージ

V4R5 で導入された新規メッセージや変更されたメッセージを以下にリストします。これらのメッセージの多くは、特定のコンパイラー・オプションが有効である場合にのみ生成されます。 さらに詳しい総合的な説明については、「*Enterprise PL/I for z/OS* メッセージおよびコード」を参照してください。

- IBM1700: BYVALUE の場合に AREA を示すフラグ (IBM1700: flags AREAs if BYVALUE)
- IBM1894: サブスクリプトがある REINIT 参照を示すフラグ (IBM1894: flags REINIT reference if it has subscripts)
- IBM2120: RETURNS で AREA を示すフラグ (IBM2120: flags AREAs in RETURNS)
- IBM2121: パラメーター・サイズと異なるサイズを持つ AREA 引数を示すフラグ (IBM2121: flags AREA arguments with size unequal to the parameter size)
- IBM2281: BETWEEN および INLIST に対する無効な引数を示すフラグ (IBM2281: flags invalid arguments to BETWEEN and INLIST)
- IBM2282: レベル 1 ではない REINIT 参照を示すフラグ (IBM2282: flags REINIT reference that is not level 1)
- IBM2283: 無効なストレージ・クラスを持つ REINIT 参照を示すフラグ (IBM2283: flags REINIT reference with invalid storage class)
- IBM2284: LOCNEWVALUE でタイプの不一致を示すフラグ (IBM2284: flags type mismatches in LOCNEWVALUE)
- IBM2285: PLISTCK に対する無効な引数を示すフラグ (IBM2285: flags invalid arguments to PLISTCK)

- IBM2286: PLISTCKE に対する無効な引数を示すフラグ (IBM2286: flags invalid arguments to PLISTCKE)
- IBM2292: 自らに対して FETCH を実行しようとする試みを示すフラグ (IBM2292: flags attempts to FETCH oneself)
- IBM2293: CMPAT(V1) の下での JSON 関数の使用を示すフラグ (IBM2293: flags use of JSON functions under CMPAT(V1))
- IBM2294: CMPAT(V1) または CMPAT(V2) の下での長ストリングの使用を示すフラグ (IBM2294: flags use of long strings under CMPAT(V1) or CMPAT(V2))
- IBM2295: BIFPREC(15) の下での長ストリングの使用を示すフラグ (IBM2295: flags use of long strings under BIFPREC(15))
- IBM2442: 埋め込みを持つ構造体を示すフラグ (IBM2442: flags structures with padding)
- IBM2642: OPTIONS(REENTRANT) の使用を示すフラグ (IBM2642: flags use of OPTIONS(REENTRANT))
- IBM2649: INLIST において重複するバイナリー引数を示すフラグ (IBM2649: flags duplicate binary arguments in INLIST)
- IBM2650: INLIST において重複する順序引数を示すフラグ (IBM2650: flags duplicate ordinal arguments in INLIST)
- IBM2651: MAXBRANCH オプションの違反を示すフラグ (IBM2651: flags violations of the MAXBRANCH option)
- IBM2652: INIT がないときに REINIT を示すフラグ (IBM2652: flags REINIT when there is no INIT)
- IBM2653: 引用符で囲まれていないプリプロセッサー・オプションを示すフラグ (IBM2653: flags preprocessor options that are not enclosed in quotation marks)

新規および変更されたプリプロセッサー・メッセージ

V4R5 で導入された新規メッセージや変更されたメッセージを以下にリストします。さらに詳しい総合的な説明については、「*Enterprise PL/I for z/OS* メッセージおよびコード」を参照してください。

- IBM3309: 4 および 8 を除く値と SYSPOINTERSIZE との比較を示すフラグ (IBM3309: flags compares of SYSPOINTERSIZE to values other than 4 or 8)
- IBM3334: マクロ・プリプロセッサーでの DEPRECATENEXT(ENTRY) を示すフラグ (IBM3334: flags DEPRECATENEXT(ENTRY) in the MACRO preprocessor)
- IBM3660: マクロ・プリプロセッサーでの DEPRECATE(ENTRY) を示すフラグ (IBM3660: flags DEPRECATE(ENTRY) in the MACRO preprocessor)
- IBM3896: 文字リテラルに変更できない VALUE 参照を示すフラグ (IBM3896: flags VALUE reference not reducible to a character literal)
- IBM3897: 数値リテラルに変更できない VALUE 参照を示すフラグ (IBM3897: flags VALUE reference not reducible to a numeric literal)
- IBM3898: サポートされないタイプを持つ VALUE 参照を示すフラグ (IBM3898: flags VALUE reference with unsupported type)

- IBM3899: あいまい参照を示すフラグ (IBM3899: flags ambiguous reference)
- IBM3900: 不明なドット修飾参照を示すフラグ (IBM3900: flags unknown dot-qualified reference)
- IBM3901: サポートされない組み込み関数の使用を示すフラグ (IBM3901: flags use of unsupported built-in function)
- IBM3902: 構造体ではない組み込み引数を示すフラグ (IBM3902: flags built-in argument that is not a structure)
- IBM3903: REAL FIXED ではない数値 VALUE 参照を示すフラグ (IBM3903: flags numeric VALUE reference that is not REAL FIXED)
- IBM3993: プリプロセッサーで失敗する表明を示すフラグ (IBM3993: flags assertions that fail in a preprocessor)

V4R4 で導入されたメッセージ

このトピックでは、V4R4 で導入された新規メッセージや変更されたメッセージ (コンパイラー・メッセージとプリプロセッサー・メッセージの両方) について説明します。

新規および変更されたコンパイラー・メッセージ

V4R4 で導入された新規および変更されたメッセージを以下にリストします。これらのメッセージの多くは、特定のコンパイラー・オプションが有効である場合にのみ生成されます。 さらに詳しい総合的な説明については、「*Enterprise PL/I for z/OS* メッセージおよびコード」を参照してください。

- IBM2261: WIDEPIC でのオーバーパンチおよび通貨記号の使用を示すフラグ
- IBM2262: WIDEPIC での A および X シンボルの使用を示すフラグ
- IBM2263: REFER 項目としての COMPLEX WIDEPIC の使用を示すフラグ
- IBM2264: LOCATES 記述子での無効な属性を示すフラグ (IBM2264: flags invalid attributes in the LOCATES descriptor)
- IBM2265: LOCATES 記述子での非定数エクステントを示すフラグ (IBM2265: flags non-constant extents in the LOCATES descriptor)
- IBM2266: 唯一の引数が LOCATES 属性を持たない場合の、組み込み関数を示すフラグ
- IBM2267: 最初の引数が LOCATES 属性を持たない場合の、組み込み関数を示すフラグ
- IBM2268: 引数が LOCATES 属性を持たない場合の、疑似変数を示すフラグ
- IBM2269: OFFSET 以外のいずれかで使用されている LOCATES 属性を示すフラグ
- IBM2270: 複数の記述を指定して使用されている LOCATES 属性を示すフラグ
- IBM2271: 最初の引数がスカラーではない場合の、組み込み関数を示すフラグ
- IBM2272: 2 番目の引数がスカラーではない場合の、組み込み関数を示すフラグ
- IBM2273: OFFSET 引数が修飾されていない場合の、組み込み関数を示すフラグ
- IBM2274: 2 番目の引数が LOCATES 属性を持たない場合の、組み込み関数を示すフラグ

- IBM2275: 3 番目の引数が AREA ではない場合の、組み込み関数を示すフラグ
- IBM2276: 引数に LOCATES 属性が含まれている必要がある場合の、組み込み関数を示すフラグ
- IBM2277: NOINCLUDE の下での %INCLUDE の使用を示すフラグ
- IBM2648: INITIAL 項目が多すぎる、STATIC 以外の宣言を示すフラグ

新規および変更されたプリプロセッサー・メッセージ

V4R4 で導入された新規および変更されたメッセージを以下にリストします。さらに詳しい総合的な説明については、「*Enterprise PL/I for z/OS* メッセージおよびコード」を参照してください。

- IBM3775: PP(CICS) または PP(SQL) での DFT(ASCII) の使用を示すフラグ (IBM3775: flags the use of DFT(ASCII) with PP(CICS) or PP(SQL))
- IBM3878: SQL 初期化中の問題を示すフラグ
- IBM3967: プロシージャー外での CALL ステートメントの使用を示すフラグ
- IBM3968: 未定義の CALL 参照を示すフラグ
- IBM3969: 入り口ではない CALL 参照を示すフラグ
- IBM3970: 関数である CALL 参照を示すフラグ
- IBM3971: STATEMENT オプションを持つ CALL 参照を示すフラグ
- IBM3985: ステートメントで左括弧と右括弧の間に使用されたセミコロンを示すフラグ
- IBM3986: 構文が無効である IF ステートメントを示すフラグ
- IBM3987: 開始が無効であるステートメントを示すフラグ

V4R3 で導入されたメッセージ

このトピックでは、V4R3 で導入された新規または変更されたメッセージを記載しています。これにはコンパイラー・メッセージとプリプロセッサー・メッセージの両方が含まれています。

新規および変更されたコンパイラー・メッセージ

V4R3 で導入された新規および変更済みのメッセージを以下にリストします。 これらのメッセージの多くは、特定のコンパイラー・オプションが有効である場合にのみ生成されます。 さらに詳しい総合的な説明については、「*Enterprise PL/I for z/OS* メッセージおよびコード」を参照してください。

- IBM2643: flags violations of the DEPRECATENEXT(BUILTIN) option
- IBM2644: flags violations of the DEPRECATENEXT(INCLUDE) option
- IBM2645: flags violations of the DEPRECATENEXT(ENTRY) option
- IBM2646: flags violations of the DEPRECATENEXT(VARIABLE) option
- IBM2647: flags violations of the DEPRECATENEXT(STMT) option
- IBM2450: flags bad arguments to Y4DATE, Y4JULIAN, and Y4YEAR
- IBM2451: flags assignments of the form x = y = z under the RULES(NOLAXIF) option

- IBM2452: flags the use of the ROUND built-in function with a negative scale factor under the RULES(NOLAXSCALE) option
- IBM2453: flags violations of the RULES(NOLAXNESTED) option
- IBM2454: flags violations of the DEPRECATE(STMT) option
- IBM2455: flags violations of the CASERULES(KEYWORD) option
- IBM2456: flags violations of the RULES(NORECURSIVE) option
- IBM2457: flags the conflict between the DFT(RECURSIVE) and RULES(NORECUSIVE) options
- IBM2458: flags the violation of the RULES(NOCONTROLLED) option
- IBM2000: flags failed assertion by using the ASSERT statements within the compiler
- IBM2237: flags the third argument to ALLCOMPARE if the argument is not a CHAR(2) constant
- IBM2238: flags the third argument to ALLCOMPARE if the argument has invalid values
- IBM2239: flags the invalid use of unspecified STRUCT types
- IBM2240: flags arithmetic operations on handles for unspecified structures
- IBM2241: flags the use of the SIZE and NEW type functions with unspecified structures
- IBM2242: flags the subtraction of handles to different types
- IBM2243-IBM2254: flag mismatches of the attributes derived from the PROCEDURE statement for the ENTRY constant with those in its explicit declaration.
- IBM2255: flags invalid argument types to UTF8
- IBM2256: flags UTF8 results that are too long
- IBM2257: flags the UTF8 source that is invalid UTF-16
- IBM2258: flags the invalid argument type to the UTF8TOCHAR or UTFTOWCHAR built-in function
- IBM2259: flags the UTF8TOCHAR or UTFTOWCHAR source that is invalid UTF-8
- IBM2260: flags the unsupported use of INITIAL expressions in DEFINE STRUCT

新規および変更されたプリプロセッサー・メッセージ

V4R3 で導入された新規および変更済みのメッセージを以下にリストします。 さらに詳しい総合的な説明については、「*Enterprise PL/I for z/OS* メッセージおよびコード」を参照してください。

- IBM3000: now used to issue informational-level messages from the CICS or DB2 preprocessor back end
- IBM3024: now used to issue informational-level %NOTE messages
- IBM3250: now used to issue warning-level messages from the CICS or DB2 preprocessor back end

- IBM3259: now used to issue warning-level %NOTE messages
- IBM3261: flags the incomplete syntax in the DEPRECATE option
- IBM3262: flags invalid keywords in the DEPRECATE option
- IBM3326: flags violations of the DEPRECATENEXT(INCLUDE) option
- IBM3500: now used to issue error-level messages from the CICS or DB2 preprocessor back end
- IBM3501: now used to issue error-level %NOTE messages
- IBM3659: flags violations of the STMT suboption of DEPRECATE
- IBM3750: now used to issue severe-level messages from the CICS or DB2 preprocessor back end
- IBM3771: now used to issue severe-level %NOTE messages

V4R2 で導入されたメッセージ

このトピックでは、V4R2 で導入された新規または変更されたメッセージを記載しています。これにはコンパイラー・メッセージとプリプロセッサー・メッセージの両方が含まれています。

新規および変更されたコンパイラー・メッセージ

V4R2 で導入された新規および変更済みのメッセージを以下にリストします。 これらのメッセージの多くは、特定のコンパイラー・オプションが有効である場合にのみ生成されます。 さらに詳しい総合的な説明については、「*Enterprise PL/I for z/OS メッセージおよびコード*」を参照してください。

- IBM2211: flags the lack of a closing shift code on a line
- IBM2212: flags the INDICATORS built-in function when it is applied to an element that is not a structure
- IBM2213: flags procedures and BEGIN blocks with too many label arrays
- IBM2214: flags the use of XMLATTR on parent structures
- IBM2215: flags the use of XMLATTR on unnamed elements
- IBM2216: flags the use of XMLATTR on arrays
- IBM2217: flags the use of XMLATTR on an element when the previous element at that logical level does not also have the XMLATTR attribute
- IBM2218: flags the use of XMLOMIT on non-native float elements
- IBM2219: flags the use of INONLY with ASSIGNABLE
- IBM2220: flags the use of OUTONLY with only NONASSIGNABLE
- IBM2221 - IBM2228: flag invalid non-constant extents in BASED
- IBM2230: flags invalid POPCNT arguments
- IBM2231: flags the use of XMLCHAR with a non-native character set
- IBM2232: flags multiple targets in BY DIMACROSS assignments
- IBM2233: flags non-structure targets in BY DIMACROSS assignments
- IBM2234: flags the use of arrays in BY DIMACROSS assignments
- IBM2235: flags invalid targets in BY DIMACROSS assignments

- IBM2419: flags the use of an option with an ARCH level that is too low
- IBM2449: flags violations of RULES(NOSELFASSIGN)
- IBM2820: flags options supported only on other platforms

新規および変更されたプリプロセッサー・メッセージ

V4R2 で導入された新規および変更済みのメッセージを以下にリストします。 さらに詳しい総合的な説明については、「*Enterprise PL/I for z/OS* メッセージおよびコード」を参照してください。

- IBM3024: transmits DB2 I-level messages
- IBM3259: transmits DB2 W-level messages
- IBM3314: flags host variables that must be fully qualified
- IBM3315: flags host variables that are arrays of structures
- IBM3316: flags host variables that are structures of arrays
- IBM3501: transmits DB2 E-level messages
- IBM3502: flags K constants that have too many digits
- IBM3503: flags K constants whose values are too large
- IBM3504: flags M constants that have too many digits
- IBM3505: flags M constants whose values are too large
- IBM3506: flags G constants that have too many digits
- IBM3507: flags G constants whose values are too large
- IBM3508: flags variables with a precision of zero in the DECLARE statement
- IBM3509: flags a DECLARE statement with invalid syntax
- IBM3515: flags scale factors that are greater than 127
- IBM3516: flags scale factors that are less than -128
- IBM3520: flags structure level values equal to 0
- IBM3521: flags structure levels that are too large
- IBM3528: flags a DECLARE statement with more than one precision value
- IBM3529: flags scale factors in a float declaration
- IBM3571: flags the inconsistent use of SQL and PL/I float options
- IBM3572: flags structure declarations in DECLARE statements with an initial level value greater than 1
- IBM3573: flags structure declarations with missing level values
- IBM3574: flags declarations of nameless scalars
- IBM3575: flags duplicate specifications of attributes
- IBM3576: flags empty EXEC SQL statements
- IBM3577: flags INCONLY when it is preceded by other options
- IBM3640: flags declarations with invalid level values
- IBM3641: flags declarations with an invalid LIKE attribute
- IBM3751: flags a missing reference after a colon in an EXEC SQL statement
- IBM3752: flags too many dots in a reference in an EXEC SQL statement

- IBM3753: flags an overly large length value in SQL TYPE IS
- IBM3754: flags a missing left parenthesis in SQL TYPE IS
- IBM3755: flags a missing integer in SQL TYPE IS
- IBM3756: flags a missing right parenthesis in SQL TYPE IS
- IBM3757: flags a missing left parenthesis in SQL TYPE IS XML AS
- IBM3758: flags a missing integer in SQL TYPE IS XML AS
- IBM3759: flags a missing right parenthesis in SQL TYPE IS XML AS
- IBM3766: flags declarations of structures with too many levels
- IBM3767: flags a length value of 0 in SQL TYPE IS
- IBM3771: transmits DB2 S-level messages
- IBM3782: flags a missing AS after SQL TYPE IS XML
- IBM3783: flags a missing type after SQL TYPE IS XML AS
- IBM3784: flags a missing LIKE after SQL TYPE IS TABLE
- IBM3785: flags a missing table-name after SQL TYPE IS TABLE LIKE
- IBM3786: flags a missing AS in SQL TYPE IS TABLE
- IBM3787: flags a missing LOCATOR in SQL TYPE IS TABLE
- IBM3788: flags an invalid type after SQL TYPE IS
- IBM3795: flags the lack of a closing shift code on a line
- IBM3799: flags host variables that are not declared in the SQL DECLARE SECTION
- IBM3805: flags a missing LARGE in SQL TYPE IS XML
- IBM3806: flags a missing OBJECT in SQL TYPE IS XML
- IBM3807: flags a missing LARGE in SQL TYPE IS CHARACTER
- IBM3808: flags a missing LARGE in SQL TYPE IS BINARY
- IBM3809: flags a missing OBJECT in SQL TYPE IS BINARY LARGE
- IBM3877: reports an internal error during an SQL back-end initialization
- IBM3880: flags undefined host variable references
- IBM3881: flags ambiguous host variable references
- IBM3882: flags an indicator array with more than one dimension
- IBM3883: flags an indicator array with nonconstant bounds
- IBM3884: flags an indicator reference that is not an array
- IBM3885: flags a host variable reference with more than one dimension
- IBM3886: flags a host variable array with nonconstant bounds
- IBM3887: flags a host variable array that is not CONNECTED
- IBM3888: flags a host variable reference with no corresponding DB2 type
- IBM3889: flags a host variable reference that is a union
- IBM3890: flags a host variable reference that is an array of structures
- IBM3891: flags a host variables reference that contains an array
- IBM3892: flags a host variable reference that contains structures
- IBM3893: flags a host variable reference that contains unnamed subelements

- IBM3894: flags an indicator reference that is not FIXED BIN(15)
- IBM3895: flags an indicator reference that is a scalar used with an array
- IBM3929: flags EXEC SQL statements not enclosed in a procedure
- IBM3934: flags invalid syntax in EXEC SQL INCLUDE
- IBM3935: flags failure to fetch the SQL back end
- IBM3936: flags an SQL back end that is not at the latest level
- IBM3937: flags EXEC SQL statements that are too long
- IBM3938: flags EXEC SQL statements with too many host variables

V4R1 で導入されたコンパイラー・メッセージ

このトピックには、V4R1 で導入された新しいメッセージと変更されたメッセージがあります。

これらのメッセージの多くは、特定のコンパイラー・オプションが有効である場合にのみ生成されます。 さらに詳しい総合的な説明については、「メッセージおよびコード」を参照してください。

新しいメッセージ

- IBM2210: flags invalid use of the VALUE type function
- IBM2442: flags violations of the RULES(NOPADDING) option
- IBM2443: flags violations of the RULES(NOGLOBALDO) option
- IBM2444: flags violations of the DEPRECATE(BUILTIN) option
- IBM2445: flags violations of the DEPRECATE(INCLUDE) option
- IBM2446: flags violations of the DEPRECATE(ENTRY) option
- IBM2447: flags violations of the DEPRECATE(VARIABLE) option
- IBM2640: flags assignments to the REFER object
- IBM2641: flags syntax errors in options
- IBM2642: flags the PROC(REENTRANT) statement when code might not be reentrant
- IBM3658: flags violations of the DEPRECATE(INCLUDE) option

変更されたメッセージ

- IBM1204: does not flag the use of the static label arrays, when they are declared as NONASGN
- IBM2016: flags only the use of the VALUE type function in the DEFINE STRUCTURE statement

V3R9 で導入されたコンパイラー・メッセージ

このトピックには、V3R9 で導入された新しいメッセージがあります。

これらのメッセージの多くは、特定のコンパイラー・オプションが有効である場合にのみ生成されます。 さらに詳しい総合的な説明については、「メッセージおよびコード」を参照してください。

- IBM1985: ファイル・オープンが失敗した場合の C ランタイム・メッセージが含まれる
- IBM1986: コンパイル中に発生するシステム (またはユーザー) 異常終了を報告する
- IBM2200: DFP ハードウェアがない場合の DFP 変換エラーを検出し、フラグを立てる
- IBM2201: ROUNDDEC 組み込み関数に対する無効引数を示すフラグ
- IBM2202: ARCH(7) のない MEMCU 組み込み関数の使用を示すフラグ
- IBM2203: 構造体での VALUE の無効な使用を示すフラグ
- IBM2204: 構造体での VALUE の無効な使用を示すフラグ
- IBM2205: 構造体での VALUE の無効な使用を示すフラグ
- IBM2206: 構造体での VALUE の無効な使用を示すフラグ
- IBM2207: 構造体での VALUE の無効な使用を示すフラグ
- IBM2208: 構造体での VALUE の無効な使用を示すフラグ
- IBM2209: 変数エクステントの設定された BASED を示すフラグ
- IBM2435: q が 0 未満の FIXED(p,q) 宣言を示すフラグ
- IBM2436: q が p より大きい FIXED(p,q) 宣言を示すフラグ
- IBM2437: PP(SQL) の重複呼び出しを示すフラグ
- IBM2438: RULES(NOSTOP) の違反を示すフラグ
- IBM2439: RULES(NOPROCENDONLY) の違反を示すフラグ
- IBM2440: RULES(NOLAXQUAL(STRICT)) の違反を示すフラグ
- IBM2441: RULES(NOGOTO(LOOSE)) の違反を示すフラグ
- IBM2635: q が p より大きい FIXED(p,q) を生成する演算を示すフラグ
- IBM2636: SELECT ステートメントに重複する ORDINAL がある場合を示すフラグ
- IBM2637: RETURNS なしで宣言され、関数として使用される ENTRY を示すフラグ
- IBM2638: MAXGEN 限度を超える行を示すフラグ
- IBM2639: MAXGEN 限度を超えるステートメントを示すフラグ
- IBM2815: BYVALUE の不適切な使用を示すフラグ
- IBM2816: BYVALUE の不適切な使用を示すフラグ
- IBM2817: BYVALUE の不適切な使用を示すフラグ
- IBM2818: FOFL を発生させる可能性のある FIXED DEC 加算演算を示すフラグ
- IBM2819: FOFL を発生させる可能性のある FIXED DEC 乗算演算を示すフラグ
- IBM3518: NAMEPREFIX オプションの違反を示すフラグ
- IBM3810: CICS 前処理時に 1 つのステートメントに対して多すぎるラベルがあることを示すフラグ

V3R8 で導入されたコンパイラー・メッセージ

このトピックには、V3R8 で導入された新しいメッセージがあります。

これらのメッセージの多くは、特定のコンパイラー・オプションが有効である場合にのみ生成されます。 さらに詳しい総合的な説明については、「メッセージおよびコード」を参照してください。

- IBM2189: flags arrays with bounds greater than 2G-1
- IBM2190: flags arrays with bounds less than -2G
- IBM2191: flags OR, NOT or QUOTE with no valid characters
- IBM2192: flags invalid PLISAXC event structures
- IBM2193: flags invalid PLISAXC event structures
- IBM2194: flags invalid PLISAXC event structures
- IBM2195: flags invalid PLISAXC event structures
- IBM2196: flags invalid PLISAXC event structures
- IBM2197: flags invalid arguments to some UTF functions
- IBM2198: flags invalid arguments to some UTF functions
- IBM2199: flags code generation without XPLINK(ON)
- IBM2429: flags CMPAT(V3) without LIMITS(FIXEDBIN(*,63))
- IBM2430: flags mismatches between LINESIZE and RECSIZE
- IBM2431: flags options invalid with GOFF
- IBM2432: flags INITIAL with PARAMETER
- IBM2433: flags INITIAL with DEFINED
- IBM2434: flags unprototyped ENTRYs under RULES(NOLAXENTRY)
- IBM2630: flags operations producing FIXED(p,q) with q larger than p
- IBM2631: flags built-in functions mixing FIXED DEC and FLOAT BIN
- IBM2632: flags built-in functions mixing FIXED DEC and FLOAT DEC
- IBM2633: flags POINTER or OFFSET variables based on FIXED BIN variables
- IBM2634: flags FIXED BIN variables based on POINTER or OFFSET variables
- IBM2814: flags allocations where an aggregate is mapped via a library call

V3R7 で導入されたコンパイラー・メッセージ

このトピックには、V3R7 で導入された新しいメッセージがあります。

これらのメッセージの多くは、特定のコンパイラー・オプションが有効である場合にのみ生成されます。 さらに詳しい総合的な説明については、「メッセージおよびコード」を参照してください。

- IBM2184: 多すぎる行を持つソース・ファイルを示すフラグ
- IBM2185: ISFINITE などへの無効な引数を示すフラグ
- IBM2186: SQRTF などへの DFP 引数を示すフラグ
- IBM2187: 大きすぎる指数を持つ DFP リテラルを示すフラグ
- IBM2188: 小さすぎる指数を持つ DFP リテラルを示すフラグ
- IBM2420: ARCH(7) を使用しない FLOAT(DFP) を示すフラグ

- IBM2421: ENDFILE ブロック内のファイルの CLOSE を示すフラグ
- IBM2422: FLOAT(DFP) で FLOAT DEC とともに HEX 属性が使用されたことを示すフラグ
- IBM2423: FLOAT(DFP) で FLOAT DEC とともに IEEE 属性が使用されたことを示すフラグ
- IBM2424: FLOAT 宣言のスケール因数を示すフラグ
- IBM2425: RULES(NOELSEIF) が適用される場合の ELSE-IF ステートメントを示すフラグ
- IBM2426: DO ステートメントの過度のネスティングを示すフラグ
- IBM2427: IF ステートメントの過度のネスティングを示すフラグ
- IBM2428: BEGIN/PROC ステートメントの過度のネスティングを示すフラグ
- IBM2621: ON ERROR SYSTEM から始まらない ON ERROR ブロックを示すフラグ
- IBM2622: DO ループで初期値を設定する関数の使用を示すフラグ
- IBM2623: DFP での FLOAT DEC と FIXED BIN の混合を示すフラグ
- IBM2624: DFP での FLOAT DEC と BIT の混合を示すフラグ
- IBM2625: DFP での FLOAT DEC と FLOAT BIN の混合を示すフラグ
- IBM2626: 3 番目の引数が 0 である SUBSTR を示すフラグ
- IBM2627: XINFO(XMI) でサポートされない REFER 構造体を示すフラグ
- IBM2628: 32 バイトより大きい BYVALUE パラメーターを示すフラグ
- IBM2629: シンボル・テーブル情報が生成されない変数を示すフラグ
- IBM2812: TRANSLATE および VERIFY 内のテーブルとして AUTO (および STATIC) 変数が使用されたことを示すフラグ
- IBM3325: どのデータ属性も指定されない %DECLARE を示すフラグ
- IBM3820: INCLUDE または XINCLUDE の PP(MACR0) の INCONLY サブオプションで、無効なマクロ・プロシージャー名が使用されたことを示すフラグ
- IBM3821: INCLUDE または XINCLUDE の PP(MACR0) の INCONLY サブオプションで、無効なマクロ・ステートメント・ラベルが使用されたことを示すフラグ
- IBM3822: INCLUDE または XINCLUDE の PP(MACR0) の INCONLY サブオプションで、無効なマクロ変数名が使用されたことを示すフラグ

V3R6 で導入されたコンパイラー・メッセージ

このトピックには、V3R6 で導入された新しいメッセージがあります。

これらのメッセージの多くは、特定のコンパイラー・オプションが有効である場合にのみ生成されます。 さらに詳しい総合的な説明については、「メッセージおよびコード」を参照してください。

- IBM2180: KEY/KEYFROM 文節を使用しないで KEYED DIRECT ファイルが使用されたことを示すフラグ
- IBM2181: PICSPEC への無効な最初の引数を示すフラグ
- IBM2182: PICSPEC への無効な 2 番目の引数を示すフラグ

- IBM2183: PICSPEC における引数のミスマッチを示すフラグ
- IBM2619: 参照されていない INCLUDE ファイルを示すフラグ
- IBM2620: REFER 項目を変更する構造体割り当てを示すフラグ
- IBM2811: ループ制御変数としての PICTURE の使用を示すフラグ

V3R5 で導入されたコンパイラー・メッセージ

このトピックには、V3R5 で導入された新しいメッセージがあります。

これらのメッセージの多くは、特定のコンパイラー・オプションが有効である場合にのみ生成されます。 さらに詳しい総合的な説明については、「メッセージおよびコード」を参照してください。

- IBM2177: SYSADATA データ・セットとして PDS メンバーを使用していることを示すフラグ
- IBM2178: LINEDIR オプションが有効な場合に %INCLUDE ステートメントがあることを示すフラグ
- IBM2179: マージンに対して大きすぎる %LINE ディレクティブがあることを示すフラグ
- IBM2416: TEST(SEPARATE) オプションを指定した LINEDIR オプションを使用していることを示すフラグ
- IBM2417: PRV を使用した FETCHABLE で、非 PARAMETER CONTROLLED の ALLOCATE および FREE があることを示すフラグ
- IBM2418: 参照されていない変数を示すフラグ
- IBM2615: 一回限りの DO ループを示すフラグ
- IBM2616: OPTIONS(NODESCRIPTOR) プロシージャーで、CHAR(*) NONVARYING パラメーターに対して SIZE が使用されていることを示すフラグ
- IBM2617: 非 PL/I ルーチンに、ラベルがパラメーターとして渡されていることを示すフラグ
- IBM2618: コンパイラー・サブオプションの無効なサブオプションを示すフラグ
- IBM2805: ターゲットを指名できる場合に、ライブラリー呼び出しによって実行される変換を示すフラグ
- IBM2806: パラメーターとしてラベルが渡されることを示すフラグ
- IBM2809: 8 バイト整数への暗黙的な FIXED DEC 変換を示すフラグ
- IBM2810: スケール付き FIXED BIN から FIXED DEC への変換における相違を示すフラグ

V3R4 で導入されたコンパイラー・メッセージ

このトピックには、V3R4 で導入された新しいメッセージがあります。

これらのメッセージの多くは、特定のコンパイラー・オプションが有効である場合にのみ生成されます。 さらに詳しい総合的な説明については、「メッセージおよびコード」を参照してください。

- IBM2165: n が 7 よりも大きい場合に、LIMITS(EXTNAME(n)) を NOWRITABLE(PRV) と一緒に使用していることに対するフラグ
- IBM2166: RENT と一緒に NOWRITABLE(PRV) を使用していることに対するフラグ
- IBM2167: CMPAT(LE) と一緒に NOWRITABLE(PRV) を使用していることに対するフラグ
- IBM2170: INTERNAL CONTROLLED のインスタンスが多すぎることに対するフラグ
- IBM2171: NOWRITABLE オプションが有効であるにもかかわらず、PACKAGE レベルで FETCHABLE ENTRY が宣言されたことに対するフラグ
- IBM2172: NOWRITABLE オプションが有効であるにもかかわらず、PACKAGE レベルで FILE CONSTANT が宣言されたことに対するフラグ
- IBM2173: NOWRITABLE オプションが有効であるにもかかわらず、PACKAGE レベルで CONTROLLED VARIABLE が宣言されたことに対するフラグ
- IBM2174: 結果が CHAR(32767) よりも長くなる REPLACEBY2 組み込み関数の参照があることに対するフラグ
- IBM2175: 2 番目および 3 番目の引数が制限付きの式でない REPLACEBY2 組み込み関数の参照に対するフラグ
- IBM2176: 結果が 32767 文字よりも多く必要とする HEX および HEXIMAGE 組み込み関数の参照に対するフラグ
- IBM2402: 2 番目の変数が 1 番目の変数を含めるために十分な長さでないにもかかわらず、2 番目の変数のアドレスをベースにして 1 番目の変数を宣言していることに対するフラグ
- IBM2403: *PROCESS ステートメントを使用していることに対するフラグ
- IBM2404: 2 番目の変数が入っている構造体が 1 番目の変数を含めるために十分な長さでないにもかかわらず、2 番目の変数のアドレスをベースにして 1 番目の変数を宣言していることに対するフラグ
- IBM2405: 偶数の FIXED DEC 精度を指定した、宣言および組み込み関数に対するフラグ
- IBM2406: 算術計算精度が DEFAULT ステートメントの中だが、VALUE 文節の外に指定されたことに対するフラグ
- IBM2407: ストリングの長さが DEFAULT ステートメントの中だが、VALUE 文節の外に指定されたことに対するフラグ
- IBM2408: AREA サイズが DEFAULT ステートメントの中だが、VALUE 文節の外に指定されたことに対するフラグ
- IBM2409: 関数内の RETURN; ステートメントに対するフラグ
- IBM2410: 関数内部の RETURN ステートメントが欠けていることに対するフラグ
- IBM2411: VARYING ストリングまたは NONCONNTECT 配列スライスに GRAPHIC 集合体の STRING が含まれていることに対するフラグ
- IBM2412: 含まれている PROCEDURE ステートメントに RETURNS オプションを指定していないにもかかわらず、式に RETURN ステートメントが指定されたことに対するフラグ

- IBM2413: パラメーターと離れて、記述子リストの中で CONNECTED を使用していることに対するフラグ
- IBM2604: SIZE を発生させる可能性のある FIXED DEC 代入に対するフラグ
- IBM2605: 無効の紙送り制御文字に対するフラグ
- IBM2607: SIZE を発生させる可能性のある PIC から FIXED DEC への代入に対するフラグ
- IBM2608: SIZE を発生させる可能性のある PIC から PIC への代入に対するフラグ
- IBM2609: コメント内のセミコロンに対するフラグ
- IBM2610: 1 つのオペランドが FIXED DEC で、もう 1 つのオペランドが FIXED BIN または FLOAT である、MULTIPLY、DIVIDE、ADD、および SUBTRACT 組み込み関数の参照があることに対するフラグ
- IBM2611: 重複する 2 進またはビット WHEN 値に対するフラグと重複値の表示
- IBM2612: 重複する文字 WHEN 値に対するフラグと重複値の表示
- IBM2613: ASGN BYADDR 引数として未初期化スカラーが使用された可能性があることに対するフラグ
- IBM2614: 2 つの比較の結果が比較される状況の式があることに対するフラグ
- IBM2801: 1 つのオペランドがゼロのスケール因数を持つ FIXED BIN で、もう 1 つのオペランドがゼロ以外のスケール因数を持つ FIXED DEC であり、そのためゼロ以外のスケール因数を持つ FIXED BIN を生成することになる算術演算があることに対するフラグ
- IBM2802: ライブラリー呼び出しによって行われた集合体マッピングがあることに対するフラグ
- IBM2803: GET/PUT STRING EDIT が最適化されているステートメントがあることに対するフラグ
- IBM2804: 完全に最適化されていない比較があることに対するフラグ
- IBM3270: CICS オプションが有効でない場合に EXEC CICS ステートメントがあることに対するフラグ
- IBM3271: CSPM オプションが有効でない場合に EXEC CSPM ステートメントがあることに対するフラグ
- IBM3272: DLI オプションが有効でない場合に EXEC DLI ステートメントがあることに対するフラグ

オブジェクトの互換性

VisualAge PL/I または Enterprise PL/I の以前のリリースによって生成されたコードとのオブジェクト互換性を維持する場合は、Enterprise PL/I の本リリースで、以前のコンパイラーで使用した以下のオプションの各セットからの同じ値を使用する必要があります。

- BACKREG(5) または BACKREG(11)
- BIFPREC(15) または BIFPREC(31)
- CMPAT(V2) または CMPAT(V1) または CMPAT(LE)
- CSECT または NOCSECT

- LIMITS(EXTNAME(n))
- NORENT または RENT
- WRITABLE または NOWRITABLE

APAR PQ66252 の PTF によって VisualAge PL/I 2.2.1 が変更 (また、対応する PTF によって Enterprise PL/I 3.1 と 3.2 が変更) されたため、FLOAT から FIXED DEC および PICTURE への変換の結果は、従来のコンパイラーで生成されていた結果と一致するようになりました。

これにより、一部の変換では多少相違が生じることがあります。 例えば、次のコードについて考えてみます。

```
dcl f          float dec(16);
dcl d2         dec(15,2);

f = 1.4417e+04;
f = f / 100;
d2 = f;
```

現在は、すべてのコンパイラーは 144.17 という値を *d2* に代入しますが、この PTF が適用される前であれば、新しいコンパイラーは 144.16 という値を *d2* に代入していました。

(V3R3、V3R4、および V3R5 に適用された) APAR PK17575 によって、コンパイラー生成コードは、MAIN に ON FINISH ブロックが含まれている場合に、CAA にフラグを設定します。 対応するライブラリー APAR によって、ライブラリーでこのフラグの有無が検査され、フラグがオンでない限り、FINISH は発生しません。この 2 つの変更によって、大幅なパフォーマンス上の改善が実現することがあります。ただし、このライブラリー APAR を適用した場合は、ON FINISH ブロックが含まれたすべての従来の Enterprise PL/I オブジェクトを再コンパイルする必要が生じることになります。そうしなければ、ON FINISH ブロックに入りません。

これらの変更点を除けば、次の制限事項に従っている限り、Enterprise PL/I V3R2 コンパイラーによってコンパイルされたコードと、VisualAge PL/I または Enterprise PL/I V3R1 のコンパイラーによってコンパイルされたコードとの間には、完全なオブジェクトの互換性があります。

- 異なる CMPAT オプションを使用してコンパイルされたコードを混合してはいけません。
- RENT コードおよび NORENT コードを混合する場合は、次に示す以前と同じ制約事項に従う必要があります。
  - RENT を指定してコンパイルされたコードは、 EXTERNAL STATIC 変数を共用している場合、NORENT でコンパイルされたコードと混在できません。
  - RENT を指定してコンパイルされたコードは、NORENT でコンパイルされたコード内の ENTRY VARIABLE セットを呼び出すことができません。
  - RENT を指定してコンパイルされたコードは、NORENT でコンパイルされたコードで FETCH された ENTRY CONSTANT を呼び出すことができません。

- RENT を指定してコンパイルされたコードは、以下の条件のいずれかが当てはまれば、NORENT でコンパイルされたコードを含むモジュールをフェッチできます。
  - フェッチされたモジュールのすべてのコードが NORENT でコンパイルされている。
  - モジュールへのエントリー・ポイントを含んでいるコードが RENT でコンパイルされている。
- NORENT コードを指定してコンパイルされたコードは、RENT でコンパイルされた任意のコードを含むモジュールを FETCH できません。
- NORENT WRITABLE を指定してコンパイルされたコードは、任意の外部 CONTROLLED 変数または任意の外部 FILE を共用している場合、NORENT NOWRITABLE でコンパイルされたコードと混在できません。

すべてのコードは、RENT/NORENT オプションおよび WRITABLE/NOWRITABLE オプションを同じように設定してコンパイルすることをお勧めします。

ランタイムの変更

コードの動作に影響を与える唯一のランタイムへの変更点 (バグ修正およびパフォーマンス向上を除く) を取り上げます。その変更内容は、SIZE 条件が、処理されない場合でも ERROR 条件にプロモートされなくなったというものです。しかし、V3R5 以降のリリースに加えられたコンパイラーへの変更点の中には、対応するライブラリーの変更が必要なものがあります。

したがって、それらのリリースによってコンパイルされたコードを使用している場合には、こうした APAR 用の PTF がインストールされていなければなりません。

- PQ97843 - TEST(NOHOOK) のサポート用
- PQ98938 - REFER の小規模コードのサポート用
- PK03093 - MAIN 後に開始される DebugTool のサポート用
- PK04110 - PLITABS のサポート用
- PK11161 - FIXED DEC(31) 演算における代替のパック 10 進記号のサポート用
- PK12504 - DB2 日時パターンのサポート用
- PK12833 - TEST(SEPARATE) のサポート用
- PK50199 - トルコ語コード・ページのサポート用
- PK50714 - ONOFFSET 組み込み関数のサポート用
- PK50715 - DFP 組み込み関数 PRED、SUCC などのサポート用
- PK50717 - ライブラリー・コードによる DFP 変換のサポート用
- PK50718 - PLIDUMP 出力での DFP 設定のレポートのため
- PK68704 - PLISAXC および ONLINE 組み込み関数のサポート用
- PK68705 - UTF を処理するための組み込み関数のサポート用
- PK68708 - UTF を処理するための組み込み関数のサポート用
- PK72146 - DFP 数学組み込み関数および CMPAT(V3) のサポート用
- PK36059 - LRECL=X のサポート用

- PK74015 - 動的ファイル割り振りのサポート用
- PK86153 - 動的ファイル割り振りのサポート用
- PK90903 - IPT に対する SYSPRINT のオープンのサポート用
- PM11241 - 新しい PLISAXD 組み込み関数のサポート用
- PM13775 - DebugTool での AUTOMON の拡張サポート用
- PM19445 - GONUMBER(SEPARATE) のサポート用
- PM19344 - 新しい ONAREA 組み込み関数のサポート用
- PM18574 - DB2 V8 および V9 システムでの DB2 コプロセッサー XREF オプションの許容度用

第 5 部 サブシステムおよびその他の言語に関する考慮事項

© Copyright IBM Corp. 1999, 2016

第 20 章 PL/I アプリケーションに関するアセンブラーの考慮事項

この章では、混合された PL/I プログラムとアセンブラー・プログラムが含まれたアプリケーションについて説明します。

ここには、以下の情報があります。

- PL/I メイン・プロシージャーを模倣したアセンブラー・プログラムに関する考慮事項
- アセンブラーおよび言語環境プログラムに準拠したアセンブラーからの PL/I の呼び出し
- 条件処理およびアセンブラー・プログラム
- アセンブラー・ユーザー出口の使用に関する考慮事項

新しいコンパイラーは、生成されたコード内で、従来のコンパイラーとは若干異なる内部制御ブロックを使用します。 そうした制御ブロックのレイアウトや意義を把握している従来のアセンブラー・コードがある場合には、そのようなコードはほとんど作動しない可能性が高いので、おそらく変更しなければなりません。 以下のような場合、こうした違いによってコード変更が必要になります。

- PL/I ラベル変数のレイアウトを把握していているアセンブラー・コードで、それをアセンブラーから PL/I コードに再びブランチするために使用する場合
- PL/I ファイル変数および関連するファイルの制御ブロックのレイアウトを把握しているアセンブラー・コードで、それをファイルの DCB を取得するために使用する場合

PL/I メイン・プロシージャーを模倣したアセンブラー・プログラムに関する考慮事項

PL/I の MAIN プロシージャーを模倣したアセンブラー・プログラムがある場合は、このアセンブラー・プログラムを言語環境プログラムに準拠したアセンブラー・プログラムである MAIN に変換する必要があります。

言語環境プログラムに準拠していないアセンブラー・プログラムは、(PL/I MAIN プロシージャーから呼び出されたのでない限り) 非 MAIN PL/I プロシージャーを呼び出すことはできません。

詳細については、*z/OS* 言語環境プログラム プログラミング・ガイド を参照してください。

アセンブラーおよび言語環境プログラムに準拠したアセンブラーからの PL/I の呼び出し

言語環境プログラムの環境で、PL/I ルーチンを呼び出すアセンブラー・プログラムは、言語環境プログラムによって定義される呼び出し規則に準拠している必要があります。

© Copyright IBM Corp. 1999, 2016

例えば、レジスター 13 が保管域を指していること、保管域の反復チェーニングが正しく行われていること、保管域の最初のワードが 0 であることなどが必要です。詳しくは、「*z/OS* 言語環境プログラム プログラミング・ガイド」を参照してください。

PL/I メインプログラムがアセンブラー・プログラムによって呼び出される場合に、言語環境プログラム準拠のアセンブラーを使用するためにアセンブラー・プログラムを変換するには、以下のどちらかを実行する必要があります。

- 新しい PL/I コンパイラーで OPTIONS(MAIN) を使用せずに PL/I プログラムを再コンパイルする。
- または制御を受け取るエントリー・ポイントが PL/I プログラムの真のエントリー・ポイントであることを確認する。

どちらの場合にも、呼び出し先の PL/I プログラムはサブルーチンとして扱われます。両方の場合とも、呼び出し先のプログラムは同じ言語環境プログラムの別プログラムの下で実行されます。この別プログラムでは、アセンブラー・プログラムがメインプログラムで、呼び出し先の PL/I プログラムがサブルーチンです。

言語環境プログラム準拠のアセンブラーが Enterprise PL/I サブルーチンに制御を渡す方法には、3 とおりがあります。

1. 静的にリンクされた PL/I サブルーチンに分岐する。
2. 言語環境プログラムのマクロ CEEFETCH を使用して、別個にリンクした Enterprise PL/I サブルーチンにブランチする。
3. LOAD や BALR などのアセンブラー命令を使用して、別個にリンクした Enterprise PL/I サブルーチンにブランチする。

1 つめまたは 2 つめの方法を使用する PL/I サブルーチンを Enterprise PL/I で再コンパイルする場合、アセンブラー・プログラムに CEESG011 をインクルードする必要はありません。アセンブラー・プログラムが 3 つ目の方法の命令を使用する場合は、 Enterprise PL/I で PL/I サブルーチンを再コンパイルする場合でも、必ずアセンブラー・プログラムに CEESG011 をインクルードする必要があります。

条件処理およびアセンブラー・プログラム

アセンブラーからの LINK の条件処理動作が明確に定義されました。

詳しくは、「*z/OS* 言語環境プログラム プログラミング・ガイド」を参照してください。

アセンブラー・ユーザー出口の使用に関する考慮事項

Enterprise PL/I によってサポートされるアセンブラー・ユーザー出口は、言語環境プログラムのユーザー出口 CEEBXITA だけです。

IBMBXITA と IBMFXITA はサポートされません。

特定の考慮事項:

- ダンプ出力用の PL1DUMP、PLIDUMP、または CEEDUMP ファイルは処理リソースとして扱われるので、エンクレーブ終了時にクリアしてはなりません。

- OS PL/I 異常終了出口 IBMBEER は、言語環境プログラムの環境では無視されます。 言語環境プログラムの環境で異常終了を強制する方法については、 40 ページの『条件処理における相違点』を参照してください。

CEEBXITA のパラメーターの詳しい説明と、アセンブラー言語ユーザー出口に関する詳細については、「*OS/390* 言語環境プログラム プログラミング・ガイド」を参照してください。

第 21 章 PL/I アプリケーションに関する CICS の考慮事項

この章では、CICS 環境で実行されるプログラムのソース言語に関する考慮事項について説明します。

また、CICS ソースまたは Enterprise PL/I ソースを使用するアプリケーションに対して必要な処置についても説明しており、次の内容が含まれます。

- CICS に関する一般的な考慮事項
- CICS 環境で実行されるプログラム用のコンパイラー・オプション
- CICS アプリケーションのリンクとランタイムに関する考慮事項
- 統合 CICS プリプロセッサーへのマイグレーション

CICS に関する一般的な考慮事項

CICS ストレージ保護機能は、CICS 3.3 で導入されました。この機能により、アプリケーション・プログラムや特に CICS 領域全体のデータ保全性とセキュリティーが強化されます。 この新機能が原因で、正常に実行されていた PL/I アプリケーションのいくつかが、ASRA(0C4) 異常終了と CICS メッセージ DFHSR0622 を出力して正常に実行されなくなる可能性があります。

PL/I アプリケーションでこの問題が発生した場合は、 CICS システム初期化パラメーター RENTPGM=NOPROTECT を設定します。 このパラメーターは、ユーザー・キーによるユーザー・プログラムの保護を設定します。 RENTPGM のデフォルトは PROTECT です。

Enterprise PL/I CICS アプリケーション内で PUT ステートメントを使用している場合 (特に `PUT DATA` ステートメント) に、上記のエラーが発生する可能性があります。

また、CICS プログラム内で、これらの PUT ステートメントはデバッグ用途にだけ使用してください。 これらのステートメントはパフォーマンスに悪影響を及ぼすため、運用プログラム内では使用しないようお勧めします。

CICS の環境で新旧のオブジェクト・コードを混合して使用する場合は、 149 ページの『オブジェクト・モジュールおよびロード・モジュールに関する考慮事項』で説明した規則と制限をすべて順守する必要があります。

CICS システム定義 (CSD) ファイルの更新

言語環境プログラムの環境で CICS 領域を起動する場合は、言語環境プログラム CEECCSD にリストされているモジュール名が、CSD 内で定義されていることを確認する必要があります。

CEECCSD は SCEESAMP 内にあります。 CICS 第 4 版の自動インストール機能を使用する場合は、CSD 内で言語環境プログラムのモジュールを手動で定義する必要はありません。

© Copyright IBM Corp. 1999, 2016

Enterprise PL/I CICS アプリケーションを実行するには、Enterprise PL/I メンバーのイベント・ハンドラー CEEEV011 を CICS CSD 定義テーブル内で定義する必要があります。

```
DEFINE PROGRAM(CEEEV011) GROUP(CEE) LANGUAGE(ASSEMBLER)
DEFINE PROGRAM(IBMPAM24) GROUP(CEE) LANGUAGE(ASSEMBLER)
```

デバッグ・ツールを使用して PL/I トランザクションをデバッグするには、次のように、CICS CSD 定義テーブル内でデバッグ・ツールの API を定義する必要があります。

```
DEFINE PROGRAM(IBMPDAPI) GROUP(CEE) LANGUAGE(ASSEMBLER)
```

マクロ・レベル・インターフェース

CICS マクロ・レベル・インターフェースはサポートされません。

CICS 環境で実行されるプログラム用のコンパイラー・オプション

PL/I の MAIN である CICS プログラムをコンパイルする際には、SYSTEM(CICS) オプションまたは SYSTEM(MVS) オプションを使用する必要があります。

CICS プログラムを再入可能にする場合 (ほとんどの CICS プログラムは再入可能にしてください)、および CONTROLLED 変数または FILE を使用する場合は、NOWRITABLE オプションも指定してコンパイルする必要があります。

CICS アプリケーションのリンクとランタイムに関する考慮事項

CICS のもとで Enterprise PL/I オブジェクト・モジュールをリンクする場合、一般的には、特別なアクションは必要なくなりました。ただし、FETCH 対象のルーチンの場合は、リンケージ・エディターの ENTRY ステートメントをコーディングして、実際のエントリー・ポイントを指示する必要があります。

PDSE は、CICS Transaction Server 1.3 以降でサポートされています。 PDSE についての説明と、前提条件として必要な APAR フィックスのリストについては、「*CICS Transaction Server for OS/390* リリース・ガイド」(資料番号: GC88-8662) を参照してください。

エラー処理

言語環境プログラムでは、PL/I ON ユニットまたは PL/I ON ユニットに呼び出されるコードで、一部の EXEC CICS コマンドを使用することが禁止されています。

以下の EXEC CICS コマンドは許可されていません。

- EXEC CICS ABEND
- EXEC CICS HANDLE AID
- EXEC CICS HANDLE ABEND
- EXEC CICS HANDLE CONDITION
- EXEC CICS IGNORE CONDITION
- EXEC CICS POP HANDLE

- EXEC CICS PUSH HANDLE

他のすべての EXEC CICS コマンドは ON ユニット内で許可されています。 しかし、それらは NOHANDLE オプション、RESP オプション、または、RESP2 オプションを使用してコーディングされていなければなりません。

PL/I MAIN プロシージャーの FETCH

CICS は、PL/I による PL/I MAIN プロシージャーの FETCH をサポートしません。

ランタイム出力

プログラムを DISPLAY(STD) を指定してコンパイルすると、すべてのランタイムの出力は、CICS 一時データ・キュー CESE に送信されます。

プログラムを DISPLAY(WTO) を指定してコンパイルすると、DISPLAY の出力は、CICS JESLOG に経路指定されます。 すべての他のランタイムの出力は、CICS 一時データ・キュー CESE に送信されます。

言語環境プログラムは、CICS 環境における MSGFILE オプションを無視します。出力データ・キューのフォーマットは、図 2に示されています。

ASA	端末 ID	トランザクション ID	B	日時 YYYYMMDDHHMMSS	B	データ

図 *2. CESE* の出力データ・キュー

また、PL/I 一時キュー CPLI および CPLD は、使用されなくなりました。 このため、CICS 宛先管理テーブル (DCT) 内で CPLI と CPLD のエントリーを指定する必要はありません。

CICS 環境で PL/I によって使用される異常終了コード

OS PL/I バージョン 2 の環境で発行されていた APL*x* 異常終了コードは、発行されなくなりました。 その代わり、言語環境プログラムの定義による異常終了コードが発行されます。

言語環境プログラムの異常終了コードについて詳しくは、「*z/OS* 言語環境プログラム ランタイム・メッセージ」を参照してください。

統合 CICS プリプロセッサーへのマイグレーション

このトピックでは、CICS プログラムを開発およびコンパイルする場合、また CICS プリプロセッサーを使用する場合に考慮しておかなければならない、いくつかの重要なポイントに焦点を当てます。

CICS の環境で実行するプログラムを開発する場合は、次の 2 つの方法のどちらかで EXEC CICS コマンドをすべて変換する必要があります。

- PL/I コンパイルの前のジョブ・ステップで、CICS 提供のコマンド言語変換プログラムを使用する
- PL/I コンパイル時に、PL/I CICS プリプロセッサーを使用する (CICS TS 2.2 以降が必要)

CICS プリプロセッサーを使用するには、さらに PP(CICS) コンパイル時オプションも指定する必要があります。

CICS プログラムが MAIN プロシージャーである場合は、SYSTEM(CICS) オプションも指定してコンパイルする必要があります。 このオプションにより NOEXECOPS が暗黙指定され、MAIN プロシージャーに渡されるすべてのパラメーターは POINTER である必要があります。

CICS プログラムで、EXEC CICS ステートメントが含まれたファイルがインクルードされているか、またはこのステートメントが含まれたマクロが使用されている場合は、コードを変換 (上記のいずれかの方法で) する前にマクロ・プリプロセッサーも実行する必要があります。 CICS プリプロセッサーを使用する場合、次の例に示すような PP オプション 1 つを使用してこのプリプロセッサーを指定できます。

```
pp (macro(...) cics(...) )
```

最後に、CICS プリプロセッサーを使用するためには、PL/I コンパイラー用の STEPLIB DD に CICS SDFHLOAD データ・セットが含まれている必要があります。

統合 PL/I CICS プリプロセッサーについて詳しくは、「*Enterprise PL/I for z/OS* プログラミング・ガイド」を参照してください。

第 22 章 PL/I アプリケーションに関する IMS の考慮事項

この章では、言語環境プログラムの環境で IMS を使用する Enterprise PL/I プログラムを実行する場合の考慮事項について説明します。

IMS へのインターフェース

言語環境プログラムは、PL/I ルーチンからの PLITDLI、ASMTDLI、および EXEC DLI の各インターフェースをサポートしています。 また、IMS/ESA 第 4 版の環境で実行される Enterprise PL/I ルーチンからの CEETDLI インターフェースもサポートしています。

言語環境プログラムでは、CEETDLI が推奨インターフェースです。CEETDLI は、アプリケーション・インターフェース・ブロック (AIB)、またはプログラム連絡ブロック (PCB) を使用する呼び出しをサポートしています。 AIB の詳細、および CEETDLI インターフェースの詳しい説明については、「*IMS/ESA Version 4 Application Programming Guide*」を参照してください。

SYSTEM(IMS) コンパイル時オプション

IMS から呼び出される PL/I MAIN プログラムをコンパイルする際には、必ず SYSTEM(IMS) オプションを使用する必要があります。

Enterprise PL/I を使用してメイン・プロシージャーを再コンパイルする場合、オブジェクト・モジュールはパラメーターが BYVALUE として渡されることを前提にしています。 必要に応じ、言語環境プログラムがパラメーターを自動的に BYVALUE 形式に変換するので、パラメーターは常に正しく渡されます。

IMS MAIN ルーチンへのパラメーターとして BYADDR 属性が明示的または暗黙に指定されている場合に、Enterprise PL/I を使用してメイン・プロシージャーをコンパイルすると、エラー・メッセージが表示され、コンパイラーは代わりに BYVALUE 属性を適用します。

SYSTEM(IMS) コンパイル時オプションについて詳しくは、「*Enterprise PL/I for z/OS プログラミング・ガイド*」を参照してください。

IMS での PLICALLA サポート

PL/I PLICALLA エントリー・ポイントは、言語環境プログラムの環境でサポートされています。

詳しくは、 42 ページの『PLICALLA に関する考慮事項』を参照してください。

© Copyright IBM Corp. 1999, 2016

サポートされている PSB 言語オプション

言語環境プログラムは、IMS のサポート対象のリリースで、次の PSBGEN LANG オプションを使用する PL/I アプリケーションをサポートします。

IMS/ESA バージョン 4

表 12 は、IMS/ESA バージョン 4 リリース 1 以降の PSB LANG オプションのサポートを示しています。

表 12. IMS/ESA バージョン 4 リリース 1 以降の PSB LANG オプション

SYSTEM オプション	エントリー・ポイント	**LANG=**
IMS	CEESTART	PLI、または PASCAL を除くその他の値
IMS	PLICALLA	PLI
MVS	PLICALLA	PLI
MVS	CEESTART	PLI
その他	- -	無効

ストレージの使用に関する考慮事項

IMS/ESA 第 3 版リリース 1 以降では、IMS インターフェースに渡されるパラメーターは、16M ラインより下のエリアに制限されなくなりました。

以下の規則を順守すれば、パラメーターは 16M ラインより上にきます。

- IMS に渡すパラメーターが CONTROLLED または BASED ストレージ内にある場合は、HEAP ランタイム・オプションに ANYWHERE サブオプションを指定します。
- IMS に渡すパラメーターが AUTOMATIC ストレージ内にある場合は、STACK ランタイム・オプションに ANYWHERE サブオプションを指定します。
- IMS に渡すパラメーターが STATIC ストレージ内にある場合は、 AMODE(31) 属性を指定してロード・モジュールをリンクします。

IMS 環境での調整条件処理

言語環境プログラムと IMS の条件処理は調整を取って行われます。つまり、アプリケーションが IMS 環境で実行されているときにプログラム割り込みまたは異常終了が発生した場合は、問題の発生個所がアプリケーションまたは IMS のどちらであるかが、言語環境プログラムの条件マネージャーに対して通知されます。

問題が IMS 内で発生した場合、言語環境プログラム (および呼び出された HLL 固有の条件処理ルーチンと同様に) は条件を IMS に戻してパーコレートします。

言語環境プログラムのランタイム・オプション TRAP(ON) を使用すると、PL/I ルーチンから呼び出した PLITDLI および ASMTDLI の両インターフェースに対する調整条件処理が、継続して言語環境プログラムによってサポートされます。

IMS/ESA バージョン 3 (PTF UN4928 適用済み)、または IMS/ESA バージョン 4 を使用する場合、言語環境プログラムでは、CEETDLI、C ルーチンからの

CTDLI、COBOL プログラムからの CBLTDLI、PL/I プログラムからの AIBTDLI、および非 PL/I プログラムからの ASMTDLI の調整条件処理もサポートされます。

IMS の外部のアプリケーション内でプログラム割り込みまたは異常終了が発生した場合、または重大度 2 以上のソフトウェア条件が IMS の外部で発生した場合、言語環境プログラムの条件マネージャーは、「*z/OS* 言語環境プログラム プログラミング・ガイド」に記述されている通常の条件処理を実行します。 この場合、IMS がデータベース・ロールバックを実行できるようにするためには、次のいずれかの処置を行う必要があります。

- アプリケーションが処理を続行できるように、エラーを完全に解決する。
- IMS に対してロールバック呼び出しを実行してから、アプリケーションを終了する。
- IMS ロールバックを引き起こすために、すべての異常終了をシステム異常終了に変換する ABTERMENC(ABEND) ランタイム・オプションを使用して、アプリケーションを異常終了させる。
- IMS ロールバックを引き起こすために、すべての異常終了をシステム異常終了に変換するように変更したアセンブラー・ユーザー出口 (CEEBXITA) を提供して、アプリケーションを異常終了させる。

  ユーザー提供のアセンブラー・ユーザー出口は、戻りコードと理由コードまたは CEEAUE_ABTERM ビットを検査し、該当する場合には CEEAUE_ABND フラグを ON にして異常終了を要求する必要があります。詳しくは、「*z/OS* 言語環境プログラム プログラミング・ガイド」を参照してください。

ライブラリー保存 (LRR) によるパフォーマンスの向上

LRR を使用すると、パフォーマンスを高めることができます。

詳しくは、 145 ページの『CPU 使用効率の向上』を参照してください。

第 23 章 PL/I アプリケーションに関する DB2 の考慮事項

この章では、DB2 とともに実行されるプログラムのソース言語に関する考慮事項について説明します。

DB2 に関する一般的な考慮事項

ユーザー定義関数を PL/I で作成した場合、DB2 はストリング・ロケーター記述子を PL/I プロシージャーに渡します。

こうしたプログラムを Enterprise PL/I の環境で正しく実行するには、CMPAT(V1) または CMPAT(V2) オプションを指定してプログラムをコンパイルする必要があります。

注: CMPAT(V1) を使用する場合、BLOB、CLOB、または DBCLOB のサイズは 32 K 未満でなければなりません。

統合 SQL プリプロセッサーへのマイグレーション

統合 PL/I SQL プリプロセッサーを使用することにより、SQL ステートメントが含まれた PL/I プログラムで、DB2 プリコンパイラーを使用した別個のプリコンパイル・ステップが不要になります。

注: SQL プリプロセッサーを使用するには、DB2 for z/OS バージョン 9 リリース 1 以降が必要です。

プログラミングとコンパイルに関する考慮事項

PL/I SQL プリプロセッサーを使用した場合は、PL/I SQL コンパイラーによって、組み込み SQL ステートメントが含まれたソース・プログラムがコンパイル時に処理され、別個のプリコンパイル・ステップを使用する必要はなくなります。 別個のプリコンパイル・ステップの使用も引き続きサポートされますが、PL/I SQL プリプロセッサーを使用することをお勧めします。 PL/I SQL プリプロセッサーを使用すると、デバッグ中に SQL ステートメントだけが表示されるように (生成された PL/I ソースは表示されない)、デバッグ・ツールによる対話式デバッグが強化されています。

さらに、PL/I SQL プリプロセッサーを使用すると、SQL プログラムに対する DB2 プリコンパイラーの制限が一部解消されます。 PL/I SQL プリプロセッサーを使用して SQL ステートメントを処理すると、次のことが可能になります。

- 構造化ホスト変数の完全修飾名を使用する。
- トップレベルのソース・ファイル内だけでなく、ネストされた PL/I プログラムの任意のレベルで SQL ステートメントを組み込む。
- ネストされた SQL INCLUDE ステートメントを使用する。

© Copyright IBM Corp. 1999, 2016

PL/I コンパイラー・リストには、PL/I SQL プリプロセッサーが生成したエラー診断情報 (SQL ステートメントの構文エラーなど) が含まれています。 EXEC SQL ステートメントのリストは、オリジナル・ソースに似た、読みやすい形式で表示されます。

PL/I SQL プリプロセッサーを使用するには、以下の手順に従ってください。

1. プログラムのコンパイル時に次のオプションを指定する。

   ```
   PP(SQL('options'))
   ```

   このコンパイラー・オプションを指定すると、コンパイラーにより PL/I SQL プリプロセッサーが呼び出されます。 SQL キーワードの後に、SQL 処理オプションのリストを括弧で囲んで指定します。 オプションはコンマまたはスペースで区切ることができますが、単一引用符または二重引用符で囲む必要があります。

   例えば `PP(SQL('DATE(USA),TIME(USA)')` は、 DATE および TIME の両データ・タイプに対して USA フォーマットを使用するようにプリプロセッサーに指示します。

   また、LOB サポートを使用するには次のオプションを指定する必要があります。

   ```
   LIMITS( FIXEDBIN(31,63)  FIXEDDEC(15,31) )
   ```

2. コンパイル・ステップ用の JCL に、次のデータ・セットに対する DD ステートメントを組み込む。

   - DB2 ロード・ライブラリー (プレフィックス.SDSNLOAD)

     PL/I SQL プリプロセッサーは、SQL ステートメントの処理を行うために DB2 モジュールを呼び出します。 このため、DB2 ロード・ライブラリーのデータ・セット名を、コンパイル・ステップ用の STEPLIB 連結に組み込む必要があります。

   - SQL INCLUDE ステートメント用のライブラリー

     ソース・プログラムへの 2 次入力を指定する `SQL INCLUDE member-name` ステートメントがプログラムにある場合は、*member-name* を含むデータ・セットの名前を、コンパイル・ステップ用の SYSLIB 連結に組み込む必要があります。

   - DBRM ライブラリー

     PL/I プログラムをコンパイルすると、DB2 データベース要求モジュール (DBRM) が生成されるので、DBRM を書き込む先のデータ・セットを指定するために、DBRMLIB DD ステートメントが必要です。

   例えば、JCL には次のような行を指定します。

   ```
   //STEPLIB  DD  DSN=DSNA10.SDSNLOAD,DISP=SHR
   //SYSLIB   DD  DSN=PAYROLL.MONTHLY.INCLUDE,DISP=SHR
   //DBRMLIB  DD  DSN=PAYROLL.MONTHLY.DBRMLIB.DATA(MASTER),DISP=SHR
   ```

統合 PL/I SQL プリプロセッサーについて詳しくは、「*Enterprise PL/I for z/OS* プログラミング・ガイド」を参照してください。

FOR BIT DATA への代入に関する注記

旧版 DB2 プリコンパイラー・サービスでは、ホスト変数の CCSID 値に対応していませんでした。 このため、FOR BIT DATA カラムを CHARACTER データで更新することができました。 DB2 V9.1 以降の新しい DB2 プリコンパイラー・サービスは、CCSID 値に対応しており、デフォルトの CCSID 値を使用してこの値をホスト変数に代入します。

これにより、FOR BIT DATA カラムを CHARACTER データで更新するコードが存在する場合、問題が発生します。 統合 PL/I SQL プリプロセッサーでは、このようなケースを扱うために新しいオプション CCSID0 / NOCCSID0 が用意されました。 デフォルトの CCSID0 オプションを使用すると、ホスト変数へ CCSID 値の 0 が代入され、FOR BIT DATA データベース列への CHARACTER 変数の代入が可能になります。

前提条件 DB2 APAR

このトピックには、Enterprise PL/I V4R1 以降のリリースにマイグレーションするときに、インストールする必要がある APAR の PTF がリストされています。

- PM18574 - DB2 V9 および V8 システムでの DB2 コプロセッサー XREF オプションの許容度用。

第 6 部 付録

© Copyright IBM Corp. 1999, 2016

付録 A. 変換とマイグレーションの支援機能

この付録では、実際の変換とマイグレーションの作業時に役立つ変換とマイグレーションのツールについて説明します。

以下のツールが有効です。

- OS PL/I ルーチン置換ツール
- OS PL/I V1R5.1 メイン・ロード・モジュール ZAP
- OS PL/I 共用ライブラリー置換ツール
- OS PL/I オブジェクト・モジュール再リンク・ツール - APAR PN69803
- EDGE Portfolio Analyzer
- ベンダー製品

OS PL/I ルーチン置換ツール

言語環境プログラムでは、OS PL/I バージョン 1 リリース 3.0 から 5.0 のロード・モジュールはサポートされていません。これらのロード・モジュールについては、次のどちらかを実行できます。

- 言語環境プログラムを使用して直接オブジェクト・モジュールを再リンクする
- ロード・モジュール内のライブラリー・ルーチンを言語環境プログラムのスタブに置換する

言語環境プログラムでは、OS PL/I バージョン 1 リリース 3.0 から 5.1 およびバージョン 2 のロード・モジュール内のライブラリー・ルーチンを、対応する言語環境プログラムのスタブに置換するための 2 つのサンプル (SCEESAMP 内に配置) が用意されています。 これらのサンプルには、ロード・モジュール内の各ライブラリー・ルーチンを言語環境プログラム内の対応するスタブに置換するための、リンケージ・エディター REPLACE 制御ステートメントのリストが含まれており、詳しくは次のとおりです。

- IBMWRLK は、MVS の非 CICS および VM 用です。

  MVS の非 **CICS** については、IBMWRLK を使用して、OS PL/I V1R3.0 から V1R5.1 および V2 のロード・モジュールを置換します (マルチタスキングと非マルチタスキングの両方)。 IBMWRLK には、OS PL/I HLL のユーザー出口 IBMBINT の名前を CEEBINT に変更するための CHANGE ステートメントが含まれています。
- IBMWRLKC は CICS 用です。

  IBMWRLKC を使用して、OS PL/I V1R3.0 から V1R5.1 および V2 のロード・モジュールを置換します。 IBMWRLKC には、OS PL/I HLL のユーザー出口 IBMBINT と PLIMAIN の名前をそれぞれ CEEBINT と CEEMAIN に変更するための CHANGE ステートメントが含まれています。 また、ロード・モジュールが CICS 環境で動作できるようにするための INCLUDE ステートメントも含まれています。

© Copyright IBM Corp. 1999, 2016

CICS マクロ言語はサポートされていません。

次に示す MVS JCL のサンプルでは、ユーザー・ロード・モジュールのランタイム・ライブラリー・ルーチンが置換される一方で、ユーザー・オブジェクト・モジュールが保持されます。この例では、MYPDS.LOAD は、MYLMOD という名前のロード・モジュールが含まれたロード・モジュール・ライブラリーのデータ・セット名です。

```
//RELINK    EXEC PGM=IEWL,PARM='LIST,MAP,XREF,SIZE(3072K,4K)',REGION=5M
//SYSPRINT  DD   SYSOUT=A
//SYSLIB    DD   DSN=CEE.V1R4M0.SCEELKED,DISP=SHR
//SAMPLIB   DD   DSN=CEE.V1R4M0.SCEESAMP,DISP=SHR
//SYSUT1    DD   UNIT=SYSDA,SPACE=(1024,(200,200))
//SYSLMOD   DD   DSN=MYPDS.LOAD,DISP=OLD
//SYSLIN    DD   *
  INCLUDE   SAMPLIB(IBMWRLK)
  INCLUDE   SYSLMOD(MYLMOD)
  NAME      MYLMOD(R)
```

CICS 環境でロード・モジュールを置換する場合は、SYSLIB で CICS SDFHLOAD データ・セットを指定する必要があります。

OS PL/I バージョン 1 リリース 5.1 メイン・ロード・モジュール ZAP

言語環境プログラムでは、次の制限のもとで OS PL/I バージョン 1 リリース 5.1 のメイン・ロード・モジュールがサポートされます。

- メイン・ロード・モジュールが、MVS の非共用ライブラリー、非 CICS、および非マルチタスキング用、または VM 用の場合は、まずこのメイン・ロード・モジュールを、言語環境プログラムの SCEESAMP に配置されている言語環境プログラム付属サンプルのいずれかを使用して ZAP する必要があります。 ZAP の使用について詳しくは、IBMRZAPM と IBMRZAPV を参照してください。次に各サンプルについて説明します。
  - MVS の非共用ライブラリー、非 CICS、非マルチタスキング用の IBMRZAPM

    ZAP されたメイン・ロード・モジュール (OS PL/I の高速初期化/終了機能を備えたものを含む) は、引き続き OS PL/I バージョン 1 リリース 5.1 および バージョン 2 の環境で動作します。 ZAP されたメイン・ロード・モジュールは、OS PL/I の高速初期化/終了機能を備えている場合、OS PL/I のランタイム初期化ルーチン IBMBPIIA を必ず動的に 1 回ロードします。IBMBPIIA は、この作業が終了するまで削除されません。 この一回限りの IBMBPIIA のロードにより、アプリケーションのパフォーマンスが影響を受けることがあります。 IBMBPIIA を LPA 内に配置すると、パフォーマンスへの影響を最小限に抑えることができます。

    ZAP されたメイン・ロード・モジュールは言語環境プログラムでサポートされますが、このロード・モジュールが OS PL/I の高速初期化/終了機能を備えている場合はサポートされません。 言語環境プログラムでは、初期化ルーチンと終了ルーチンは必ず動的にロードされます。 「*z/OS Language Environment Installation and Customization under OS/390*」および「*z/OS Language Environment* カスタマイズ」で推奨されているように、言語環境プ

ログラムのライブラリー・ルーチンと CEEBLIIA を LPA(E) に配置すると、パフォーマンスへの影響を最小限に抑えることができます。

– VM 用の IBMRZAPV

ZAP されたメイン・ロード・モジュールは、OS PL/I バージョン 1 リリース 5.1 およびバージョン 2 の環境ではサポートされていません。 これは、言語環境プログラムでのみサポートされています。

メイン・ロード・モジュールを ZAP しない場合は、 203 ページの『OS PL/I ルーチン置換ツール』を参照して、他にできることを確認してください。 アプリケーションを Enterprise PL/I または OS PL/I バージョン 2 で再コンパイルすることもできます。 言語環境プログラムでの OS PL/I のオブジェクト・モジュールとロード・モジュールのサポートについて詳しくは、 53 ページの『第 7 章 オブジェクト・モジュールおよびロード・モジュールに関する考慮事項』を参照してください。

サンプル ZAP は、言語環境プログラムを所有していないが言語環境プログラムへのマイグレーション準備をご希望のお客様のために IBM サポートで提供されています。

OS PL/I 共用ライブラリー置換ツール

共用ライブラリーを使用している OS PL/I バージョン 1 リリース 5.1 および バージョン 2 のロード・モジュールをサポートするには、この共用ライブラリー内のライブラリー・モジュールを言語環境プログラムのスタブに置換する必要があります。

言語環境プログラムでは、共用ライブラリーを置換するための次の 2 つのサンプル JCL が、SCEESAMP 内に用意されています。

- OS PL/I バージョン 1 リリース 5.1 MVS の CICS またはマルチタスキング、および OS PL/I バージョン 2 の共用ライブラリー用の IBMRLSLA
- OS PL/I バージョン 1 リリース 5.1 MVS の非 CICS 非マルチタスキングの共用ライブラリー用の IBMRLSLB

JCL を使用する前に、言語環境プログラムでの OS PL/I 共用ライブラリーのサポートについて確認しておく必要があります。

OS PL/I オブジェクト・モジュール再リンク・ツール - APAR PN69803

OS PL/I バージョン 2 リリース 3 には、PL/I-COBOL ILC アプリケーションと PLISRTx アプリケーションをマイグレーションするために役立つ APAR PN69803 が用意されています。

ILC アプリケーション

言語環境プログラムでは、OS PL/I-COBOL ILC アプリケーションはサポートされていません。 PL/I-COBOL ILC アプリケーション内の OS PL/I オブジェクト・モジュールは、すべて再リンクする必要があります。

言語環境プログラムでの ILC のサポートについては、 49 ページの『言語間通信 (ILC) のサポートの相違点』を参照してください。

ただし、PL/I-COBOL ILC アプリケーション内の OS PL/I オブジェクト・モジュールを PN69803 を使用して再リンクした場合は、得られたロード・モジュールは言語環境プログラムでサポートされます。 PN69803 により、OS PL/I バージョン 2 リリース 3 の使用中に、PL/I-COBOL ILC の再リンクを準備できるという柔軟性が得られます。 再リンクを完了したら、準備ができ次第言語環境プログラムに切り替えることができます。

PL/I-COBOL ILC アプリケーションを PN69803 を使用して再リンクする前に、まず次の PL/I-COBOL ILC APAR を PL/I と COBOL に適用する必要があります。

OS PL/I V2R3 共通ライブラリー: PN36844
VS COBOL II V1R3.0 ライブラリー: PN13459
VS COBOL II V1R3.1 ライブラリー: PN04721
VS COBOL II V1R3.2 ライブラリー: PN09732

注**:** VS COBOL II V1R4.0 のベース・コードには、上記の COBOL APAR が含まれています。

上記の APAR をまだ適用していない場合は、PN69803 は機能しません。アプリケーションに PL/I-COBOL ILC が含まれていない場合は、上記の APAR は不要です。

PL/I-COBOL ILC アプリケーションを PN69803 を使用して再リンクしても、これらのアプリケーションが、本書または「*COBOL for OS/390 & VM Migration Guide*」で説明されている再リンクが必要な機能を備えている場合は、言語環境プログラムを使用してこれらのアプリケーションをさらにリンクすることが必要になることがあります。 例えば、アプリケーションに COBOL NORES が含まれているか、またはロード・モジュールに言語環境プログラムでサポートされていない OS PL/I オブジェクト・モジュールが含まれている場合は、アプリケーションをさらに再リンクする必要があります。 後者の場合は、OS PL/I オブジェクト・モジュールを Enterprise PL/I または OS PL/I バージョン 2 で再コンパイルする必要があります。

PLISRTx アプリケーション

PLISRTx を使用する OS PL/I アプリケーションは、Language Environment for OS/390 & VM リリース 1.4 以降でサポートされていますが、PLISRTx を使用するアプリケーションを再リンクすることをお勧めします。

その理由については、 46 ページの『PLISRTx のサポートの相違点』を参照してください。 この再リンクは、言語環境プログラムを使用して行うか、または OS PL/I バージョン 2 リリース 3 上の PN69803 を使用して行うことができます。どちらの場合でも、ロード・モジュールで、言語環境プログラム DFSORT インターフェースのサポートを利用できるという利点が得られます。

EDGE Portfolio Analyzer

Edge Portfolio Analyzer は、既存の OS PL/I と PL/I for MVS & VM のロード・モジュールのインベントリーを作成するのに役立ちます。 Edge Portfolio Analyzer は以下のタスクを実行できます。

- ロード・モジュールを作成した OS PL/I コンパイラーまたは PL/I for MVS & VM コンパイラーのバージョンとリリースを確認する。
- ロード・モジュールのコンパイル時に指定されたコンパイラー・オプションを確認する。
- 現在のシステム日付が必要なロード・モジュールを特定する。
- 置換する必要のある CSECT を特定する。

注: Edge Portfolio Analyzer は、現在は IBM からは販売されていませんが、Edge Information Group 社から直接購入することができます。詳しくは、同社の Web サイト www.edge-information.com をご覧ください。

ベンダー製品

ソース・プログラムを Enterprise PL/I プログラムにアップグレードして言語環境プログラムに移行するのに役立つ、いくつかの IBM 以外の変換ツールが提供されています。 IBM では、言語環境プログラムおよび Enterprise PL/I で使用できるベンダー製品のリストを、「*Language Environment Enabled Vendor Tools and Application Packages*」という文書にまとめています。この情報を入手するには、Web 上で `http://www.ibm.com/s390/le` にアクセスし、「Library」リンクに移動してください。

付録 B. コンパイラー・エレメントの比較

希望に応じて OS PL/I または PL/I for MVS & VM と同じ SMP/E ゾーンにインストールできるように、Enterprise PL/I のパーツ名が変更されました。

各製品のエレメントの識別に役立つように、次の表に名前の違いをリストします。

表 *13. PL/I* のエレメント名

OS PL/I	PL/I for MVS & VM	Enterprise PL/I
IEL0AA	IEL1AA	IBMZPLI
IKJEN00n	IEL1IKJn	
IEL0nn	IEL1nn	IBMZnn
PLInnnnn	IEL1Mnnn	IBMZMnnn
PLIXnnn	IEL1nnn	IBMZnnn
PLIHELP	IEL1PLIH	--

© Copyright IBM Corp. 1999, 2016

付録 C. コンパイラーの制限の比較

このセクションには、OS PL/I、PL/I for MVS & VM、VisualAge PL/I、および Enterprise PL/I におけるコンパイラーの実装の制限がリストされています。

表 *14.* 言語エレメントの制限

言語エレメント	制限内容	**OS PL/I**	**PL/I for MVS & VM**	**VisualAge PL/I**	**Enterprise PL/I**
配列	配列の次元の最大数	15	15	15	15
	下限の最小値	-2147483648	-2147483648	-2147483648	-2147483648
	上限の最大値	+2147483647	+2147483647	+2147483647	+2147483647
構造体	構造体内のレベルの最大数	15	15	15	15
	構造体内の最大レベル番号	255	255	255	255
算術精度	FIXED DEC の最大精度	15	15	31	31
	FIXED BINARY の最大精度	31	31	63	63
	FLOAT DEC の最大精度	33	33	33	33
	FLOAT BINARY の最大精度	109	109	109	109
	FIXED データの最大スケール因数	127	127	127	127
	FIXED データの最小スケール因数	-128	-128	-128	-128
ストリング変数/定数と AREA 変数/定数	CHARACTER の最大の長さ	32767	32767	32767	32767
	BIT の最大の長さ	32767	32767	32767	32767
	GRAPHIC の最大の長さ	16383	16383	16383	16383
	WIDECHAR の最大の長さ	なし	なし	16383	32767
	AREA の最大サイズ	2147483647	2147483647	2147483647	2147483647
組み込み関数	IAND、IOR、 MAX、および MIN 関数への引数の最大個数	64	64	64	64

© Copyright IBM Corp. 1999, 2016

表 *14.* 言語エレメントの制限 *(*続き*)*

言語エレメント	制限内容	**OS PL/I**	**PL/I for MVS & VM**	**VisualAge PL/I**	**Enterprise PL/I**
プログラム・サイズ	ID の最大の長さ	31	31	100	100
	プログラム内のプロシージャーの最大数	255	255	255	255
	ブロック内の DEFAULT ステートメントの最大数	31	31	31	31
	%INCLUDE ステートメントの最大ネスト	8	8	2046	2046
	ソース・ファイル内の最大行数	65,535	65,535	1048575	1048575
	ステートメントの最大数	32,767	32,767	16777215	16777215
	ブロック内の LIKE 属性の最大数	63	63	63	63
	データ・リスト内の出力式の最大数	60	60	60	60
	データ・リスト内の反復 DO 指定の最大数	25	25	50	50
プログラム・サイズ	位置合わせされていないビットを含まないデータ集合の最大サイズ	2147483648	2147483648	2147483647	2147483647
	位置合わせされていないビットを含むデータ集合の最大サイズ	268435455	268435455	268435455	268435455
	CALL または関数参照内の引数の最大数	64	64	255	255
	プロシージャーの最大パラメーター数	64	63	4095	4095
	分配された属性の最大ネスト	15	15	15	15
	BEGIN および PROCEDURE ステートメントの最大ネスト	42	42	30	30
	DO グループの最大ネスト	38	38	49	49
	IF ステートメントの最大ネスト	80	80	49	49
	SELECT ステートメントの最大ネスト	50	50	49	49
	%NOTE メッセージの最大の長さ	256	256	32767	32767

表 14. 言語エレメントの制限 (続き)

言語エレメント	制限内容	OS PL/I	PL/I for MVS & VM	VisualAge PL/I	Enterprise PL/I
その他	文字ピクチャー内のピクチャー文字の最大数	511	511	511	511
	数値ピクチャー内の最大バイト数	256	256	253	253
	数値ピクチャー内の数値ピクチャー文字の最大数	15	15	31	31
	KEYTO 文字ストリングの最大の長さ	120	120	120	120
	KEYTO グラフィックまたはワイド文字ストリングの最大長	60	60	60	60
	KEY の最大の長さ	8	8	32763	32763
	LINESIZE の最大行サイズ	32,000	32,000	32,000	F フォーマットあるいは U フォーマットでは 32,759、V フォーマットでは 32,751
	LINESIZE の最小行サイズ	10	10	1	1
	PAGESIZE の最大ページ・サイズ	32,000	32,000	32,767	32,767
その他	PAGESIZE の最小ページ・サイズ	1	1	1	1
	DISPLAY 文字ストリングの最大サイズ	126	126	126	126
	最大の DISPLAY 応答メッセージ	72	72	72	72

付録 D. バッチ処理のサンプル

次のコード・サンプルは、Enterprise PL/I で「バッチ・コンパイラー」を実装する方法を示したものです。

```
batch: proc options(main);

   dcl eof     bit(1);
   dcl rc      fixed bin(15);
   dcl system  builtin;
   dcl source  char(80);
   dcl sysutz  file output record sequential
               env( fb,recsize(80) );

   dcl compin  file input record sequential;

   dcl plixopt ext static char(40) var
               init('errcount(0),heap(2m,1m,any,free)');

   open file(compin);

   rc = 0;
   eof = '0'b;
   data_read = '0'b;
   on endfile(compin) eof = '1'b;

   data_read = '0'b;
   open file(sysutz);
   read file(compin) into(source);
   do while( eof = '0'b );
     if substr(source,1,8) = '*PROCESS' then
       if data_read then
         do;
           close file(sysutz);

           rc = max( rc, system('ibmzpli @dd:options') );

           data_read = '0'b;
           open file(sysutz);
         end;
       else;
     else
       data_read = '1'b;
     write file(sysutz) from(source);
     read file(compin) into(source);
   end;

   close file(sysutz);

   rc = max( rc, system('ibmzpli @dd:options') );
   call pliretc(rc );
 end;
```

このプログラムをコンパイルおよびリンクして、ランタイムに次の JCL を使用することにより、このプログラムを「バッチ・コンパイラー」として使用することができます。

```
//SYSPRINT  DD SYSOUT=*
//OPTIONS   DD  *
 dd(*,sysutz) name
 limits(extname(7)) norent cmpat(v2)
```

© Copyright IBM Corp. 1999, 2016

```
//COMPIN    DD  *
*PROCESS X(F);
  x: proc;
    dcl a ext char(80);
  end;
*PROCESS NORENT;
  y: proc;
    dcl b ext char(40);
  end;
//SYSLIN   DD DSN=...,DISP=(MOD)
//SYSUT1    DD DSN=&&SYSUT1,UNIT=SYSDA,
//             SPACE=(1024,(200,50),,CONTIG,ROUND),DCB=BLKSIZE=1024
//SYSUTZ  DD DSN=&&SOURCE,DISP=(NEW),UNIT=SYSSQ,
//             SPACE=(CYL,(3,1))
```

OPTIONS DD の最初の行は、DD(*,SYSUTZ) と NAME を指定しており、このプログラムをバッチ・コンパイラーとして動作させるために必要です。 2 行目は単なる例として使用されています。

付録 E. デバッグ・ツールの比較

デバッグ・ツールは、言語環境プログラム内で実行されるプログラム・アナライザーであり、 Enterprise PL/I などの多くの高水準言語をサポートしています。

Enterprise PL/I では、デバッグ・ツールはコンパイラーの機能の一部として注文することができます。

デバッグ・ツール間の相違点

IBM Debug Tool は、PL/I と言語環境プログラムをサポートした対話式デバッガーです。 Debug Tool は、PLITEST と同等の機能を備えています。 ただし、Debug Toolでは、一部の PLITEST コマンドの名前が変更されており、使用できません。これらのコマンドのリストを表 15 に示しています。

OS PL/I アプリケーションで Debug Tool を使用するには、システムに Language Environment for OS/390 & VM リリース 4 以降がインストールされている必要があります。

表 *15. PLITEST* コマンドおよびこれらに対応する *Debug Tool* コマンド

PLITEST コマンド	対応する **Debug Tool** コマンド
CLEAR ON	CLEAR AT OCCURENCE
LIST %FPRS	LIST SHORT FLOATING
LIST %LPRS	LIST LONG FLOATING
LIST %GPRS	LIST REGISTERS
LIST SNAP	LIST CALLS
MOVECURS	CURSER
ON	AT OCCURENCE
QUERY AT	LIST AT
QUERY ATTRIBUTES	DESCRIBE ATTRIBUTES
QUERY BEARINGS	QUERY LOCATION
QUERY ENVIRONMENT	DESCRIBE ENVIRONMENT
QUERY MONITOR	LIST MONITOR
QUERY NAMES 'パターン'	LIST NAMES 'パターン'
QUERY NAMES PROCEDURE	LIST PROCEDURE
QUERY PROGRAM	DESCRIBE PROGRAM
QUERY STATEMENT NUMBERS	LIST STATEMENT NUMBERS
SEARCH	FIND
SET GRAPHIC	SET DBCS
SET LANGUAGE	SET NATIONAL LANGUAGE
SET LAST n	SET HISTORY n
SET FILE	SET LOG
SIGNAL (ON cond) PROGRAM	TRIGGER (ON cond)

© Copyright IBM Corp. 1999, 2016

表 *15. PLITEST* コマンドおよびこれらに対応する *Debug Tool* コマンド *(*続き*)*

PLITEST コマンド	対応する **Debug Tool** コマンド
SIGNAL (ON cond) TEST	TRIGGER AT OCCURENCE (ON cond)
SIGNAL (AT cond) TEST	TRIGGER AT (AT cond)
VTRACE	STEP
WINDOWS	LAYOUT

特記事項

本書は米国 IBM が提供する製品およびサービスについて作成したものです。本書に記載の製品、サービス、または機能が日本においては提供されていない場合があります。日本で利用可能な製品、サービス、および機能については、日本 IBM の営業担当員にお尋ねください。 本書で IBM 製品、プログラム、またはサービスに言及していても、その IBM 製品、プログラム、またはサービスのみが使用可能であることを意味するものではありません。 これらに代えて、IBM の知的所有権を侵害することのない、機能的に同等の製品、プログラム、またはサービスを使用することができます。 ただし、IBM 以外の製品とプログラムの操作またはサービスの評価および検証は、お客様の責任で行っていただきます。

IBM は、本書に記載されている内容に関して特許権 (特許出願中のものを含む) を保有している場合があります。 本書の提供は、お客様にこれらの特許権について実施権を許諾することを意味するものではありません。 実施権についてのお問い合わせは、書面にて下記宛先にお送りください。

〒103-8510
東京都中央区日本橋箱崎町19番21号
日本アイ・ビー・エム株式会社
法務・知的財産
知的財産権ライセンス渉外

以下の保証は、国または地域の法律に沿わない場合は、適用されません。

IBM およびその直接または間接の子会社は、本書を特定物として現存するままの状態で提供し、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任を負わないものとします。 国または地域によっては、法律の強行規定により、保証責任の制限が禁じられる場合、強行規定の制限を受けるものとします。

この情報には、技術的に不適切な記述や誤植を含む場合があります。 本書は定期的に見直され、必要な変更は本書の次版に組み込まれます。 IBM は予告なしに、随時、この文書に記載されている製品またはプログラムに対して、改良または変更を行うことがあります。

本プログラムのライセンス保持者で、(i) 独自に作成したプログラムとその他のプログラム (本プログラムを含む) との間での情報交換、および (ii) 交換された情報の相互利用を可能にすることを目的として、本プログラムに関する情報を必要とする方は、下記に連絡してください。

IBM Corporation
J74/G4
555 Bailey Avenue
P.O. Box 49023
San Jose, CA 95161-9023
U.S.A.

© Copyright IBM Corp. 1999, 2016

本プログラムに関する上記の情報は、適切な使用条件の下で使用することができますが、有償の場合もあります。

本書で説明されているライセンス・プログラムまたはその他のライセンス資料は、IBM 所定のプログラム契約の契約条項、IBM プログラムのご使用条件、またはそれと同等の条項に基づいて、IBM より提供されます。

著作権使用許諾:

本書には、様々なオペレーティング・プラットフォームでのプログラミング手法を例示するサンプル・アプリケーション・プログラムがソース言語で掲載されています。 お客様は、サンプル・プログラムが書かれているオペレーティング・プラットフォームのアプリケーション・プログラミング・インターフェースに準拠したアプリケーション・プログラムの開発、使用、販売、配布を目的として、いかなる形式においても、IBM に対価を支払うことなくこれを複製し、改変し、配布することができます。 このサンプル・プログラムは、あらゆる条件下における完全なテストを経ていません。従って IBM は、これらのサンプル・プログラムについて信頼性、利便性もしくは機能性があることをほのめかしたり、保証することはできません。

この情報をソフトコピーでご覧になっている場合は、写真やカラーの図表は表示されない場合があります。

プログラミング・インターフェース情報

本書は、お客様が PL/I 前リリースから、Enterprise PL/I および z/OS 言語環境プログラムへマイグレーションする際に役立ちます。 本書には、プログラムを作成するユーザーが Enterprise PL/I のサービスを使用するためのプログラミング・インターフェースが記述されています。

商標

IBM、IBM ロゴおよび ibm.com は、世界の多くの国で登録された International Business Machines Corp. の商標です。他の製品名およびサービス名等は、それぞれ IBM または各社の商標である場合があります。現時点での IBM の商標リストについては、www.ibm.com/legal/copytrade.shtml をご覧ください。

Windows は、Microsoft Corporation の米国およびその他の国における商標です。

参考文献

PL/I の資料

Enterprise PL/I for z/OS

「プログラミング・ガイド」、GI88-4249
「言語解説書」、SA88-4235
「メッセージおよびコード」、GA88-4237
「コンパイラーおよびランタイム 移行ガイド」、GA88-4236

PL/I for MVS & VM

「インストールとカストマイズ」、SC88-7221
「言語解説書」、SC88-7219
「コンパイル時メッセージおよびコード」、SC88-7224
「診断の手引き」、SC88-7223
「移行の手引き」、SC88-7220
「プログラミングの手引き」、SC88-7218
「参照要約」、SX88-7011

PL/I for AIX

「プログラミング・ガイド」、SA88-4427
「言語解説書」、SA88-4429
「メッセージおよびコード」、GA88-4430
「インストール・ガイド」、GA88-4428

関連資料

DB2 UDB for z/OS

「管理ガイド」、SA88-4204

「アプリケーション・プログラミングおよび *SQL* 解説書」、SA88-4205

「コマンド解説書」、SA88-4208

「メッセージ」、GA88-4213

「コード」、GA88-4207

「*SQL* 解説書」、SA88-4217

インフォメーション・センター (publib.boulder.ibm.com/infocenter/dzichelp/v2r2/topic/com.ibm.db2z10.doc/src/alltoc/db2z_10_prodhome.htm) も参照してください。

DFSORT

「アプリケーション・プログラミングの手引き」、SC88-7061

「導入およびカスタマイズ」、SC88-7163

© Copyright IBM Corp. 1999, 2016

IMS/ESA

「アプリケーション・プログラミング: データベース管理プログラム」、SC88-7552

「*Application Programming: Database Manager Summary*」、SC26-8037

「アプリケーション・プログラミング: 設計の手引き」、SC88-7542

「アプリケーション・プログラミング: トランザクション管理プログラム」、SC88-7553

「*Application Programming: Transaction Manager Summary*」、SC26-8038

「アプリケーション・プログラミング: *EXEC DL/I* コマンド *(CICS* および *IMS™)*」、SC88-7554

「*Application Programming: EXEC DL/I Commands for CICS and IMS Summary*」、SC26-8036

TXSeries for Multiplatforms

「*Encina* 管理の手引き 第 *2* 巻：サーバー管理」、SD88-7403

「*Encina SFS Programming Guide*」、SC09-4483

インフォメーション・センター (publib.boulder.ibm.com/infocenter/txformp/v7r1/index.jsp) も参照してください。

z/Architecture

「解説書」、SA88-8773

z/OS 言語環境プログラム

「概念」、SA88-8555

「デバッグ・ガイド」、GA88-8548

「ランタイム・メッセージ」、SA88-8554

「カスタマイズ」、SA88-8552

「プログラミング・ガイド」、SA88-8549

「プログラミング・リファレンス」、SA88-8550

「ランタイム・アプリケーション マイグレーション・ガイド」、GA88-8553

「*ILC* (言語間通信) アプリケーションの作成」、SA88-8551

z/OS MVS

「*JCL* 解説書」、SA88-8569

「*JCL* ユーザーズ・ガイド」、SA88-8570

「システム・コマンド」、SA88-8593

z/OS TSO/E

「コマンド解説書」、SA88-8628

「ユーザーズ・ガイド」、SA88-8638

z/OS UNIX システム・サービス

「*z/OS UNIX* システム・サービス コマンド解説書」、SA88-8641

「*z/OS UNIX* システム・サービス・プログラミング*:* アセンブラー呼び出し可能サービス 解説書」、SA88-8642

「*z/OS UNIX* システム・サービス ユーザーズ・ガイド」、SA88-8640

Unicode および文字表現

「*OS/390 Unicode* サポート*:* 変換サービスの使用法」、SD88-6163

索引

日本語, 数字, 英字, 特殊文字の順に配列されています。なお, 濁音と半濁音は清音と同等に扱われています。

[ア行]

アクセシビリティー
  支援技術 xvi
  Enterprise PL/I for z/OS の xv
アセンブラー・サポート
  コンパイラーの起動 76
  メイン・パラメーター・リスト 51
  CEESG011 をリンクする必要がある 140
  IMS に関する考慮事項 64, 194
  PLIMAIN エントリー・ポイント 51
  PLISTART エントリー・ポイント 51
  PL/I の呼び出し 51, 140, 186
アプリケーションの再チューニング
  ストレージ使用効率、向上 146
  CPU 使用効率 145
  IMS 環境、向上 147
異常終了コード
  CICS に関する考慮事項 63, 191
インストール
  言語環境プログラム 15
エラー処理、CICS に関する考慮事項 62
置き換え、OS PL/I 共用ライブラリーの
  サンプル置換エイド 205
オブジェクト・モジュール
  一般的な考慮事項 53
  言語環境プログラム・サポート
    リリース 3.0 より前の OS PL/I バージョン 1 56
    OS PL/I バージョン 1.3.0 ～ 1.4.0 55
    OS PL/I バージョン 1.5.0 55
    OS PL/I バージョン 1.5.1 53
    OS PL/I バージョン 2 56
  ILC マイグレーション・エイド 205
オブジェクト・モジュールおよびロード・モジュールに関する考慮事項 53, 56

[カ行]

概要
  PL/I ランタイム環境 8
疑似変数の制限 72
教育
  言語環境プログラム 16
  Enterprise PL/I 26
共用ライブラリー、OS PL/I
  サンプル置換エイド 205
共用ライブラリー置換エイド
  IBMRLSLx の使用 205
共用ライブラリーのサポート 46
組み込み関数
  数学 142
  スケールされた FIXED BIN を持つ 133
  図形型から文字型への変換に使用 134
  制限付き 72
  100 以上の新規 8
  DATE/TIME 46
言語環境プログラム
  移行の計画 15
  既存アプリケーションの実行 33
  既存の CICS アプリケーションの起動 34
  既存の非 CICS アプリケーションの起動 33
  プログラマーの教育 16
  Enterprise PL/I 前提条件レベル 15
言語環境プログラム・ライブラリー xiv
言語間通信 (ILC)
  使用可能な言語 49
  相違点 49
構造式の制約事項 71
考慮事項
  インストール
    製品構成 209
    製品構成、SCEELKED 7
    製品構成、SCEERUN 7
    Enterprise PL/I 25
    OS/390 要件 7
  サブシステム
    CICS 61, 189
    DB2 66, 197
    IMS 63, 193
  マイグレーション前 37
    アセンブラー 185
    事前初期設定済みプログラム 45
    条件処理 40
    使用ストレージの再チューニング 145
    ストレージ報告書 49
    ソート・プログラムの使用 46
    デバッグ・ツール 217
考慮事項 (続き)
  マイグレーション前 (続き)
    パフォーマンスの再チューニング 145
    ユーザー戻りコード 47
    ランタイム・オプション 37
    ランタイム・メッセージ 47, 141, 142
    DATE/TIME 組み込み関数 46
    ILC の相違点 49
    PLIDUMP 48
  リンク・エディット
    シンボル・テーブル 59
    数学ルーチン 60
    CHANGE カード 139
    NCAL リンケージ・エディター・オプション 60
    PLICALLA および PLICALLB 137
互換性
  考慮事項
    PLICALLA エントリー・ポイント 42
    PLICALLB エントリー・ポイント 43
  コンパイラー・オプション
    BIFPREC(15) 81
    CMPAT(V*) 81
    DFT(LINKAGE(SYSTEM)) 81, 86
    DFT(NOBIN1ARG) 85
    DFT(OVERLAP) 81, 86
    EXTRN(FULL) 81, 83
    LIMITS(EXTNAME(7)) 81, 83
    NOREDUCE 81, 86
    NORENT 81, 84
    NORESEXP 81, 87
    NOWRITABLE 87
    RULES(LAXCTL) 81, 87
  必要なランタイム・オプション
    ABTERMENC(RETCODE) 139
    DEPTHCONDLMT(0) 139
    ERRCOUNT(0) 140
    TRAP(ON) 140
    XUFLOW(ON) 140
コンパイラーの制約事項
  疑似変数 72
  組み込み関数 72
  構造式 71
  自動変数の範囲 72
  配列式 71
  マクロ・プリプロセッサー 74
  マルチタスキング機能 46

© Copyright IBM Corp. 1999, 2016

コンパイラーの制約事項 (続き)
DBCS 73
DEFAULT ステートメント 72
GRAPHIC および POSITION 73
iSUB 定義 73
LABEL 配列 73
OPTIONS(REENTRANT) 73
RECORD I/O 70
STREAM I/O 71
VM 10
コンパイラーの利点
新しい組み込み関数 8
統合プリプロセッサー 8
マルチスレッド化、サポート 8
FETCH 8
コンパイラー・オプション
サポートされない 75
制限付き 74
BACKREG 81
BIFPREC 82
CMPAT 83, 149
DEFAULT
LINKAGE 86, 149
NOBIN1ARG 85
NONASGN 89
NONCONNECTED 89
NOOVERLAP 90
OVERLAP 86, 149
REORDER 89, 90
EXTRN 83, 149
GONUMBER 96
LIMITS
EXTNAME 83, 149
NOREDUCE 86, 90
NORENT 84, 91, 149, 180
NOWRITABLE 84, 87, 181
OPTIMIZE 89, 90
PREFIX 96
REDUCE 90
RENT 84, 180
RULES
LAXCTL 87
LAXSTRZ 95
NOLAXCTL 91
NOLAXDCL 92
NOLAXIF 93
NOLAXMARGINS 94
NOMULTICLOSE 95
SYSTEM 85
TEST 96
WRITABLE 84, 181
コンパイラー・オプションの制限
INCLUDE 74
LANGLVL 74
LIST 74
STMT 75
コンパイラー・オプションの制限 (続き)
SYSTEM 75
コンパイラー・サポートの廃止
マルチタスキング 69
無効なコード 69
CHARSET(48) 69
CHECK 69
EGCS 69
Fortran 69
VM 10
コンパイラー・メッセージ
2603 101
EXIT オプション 112
IBM1044 97
IBM1053 97, 134
IBM1063 126
IBM1065 98
IBM1089 120
IBM1091 98
IBM1099 98
IBM1181 100
IBM1196 132
IBM1206 100
IBM1208 101
IBM1215 102
IBM1216 102
IBM1220 103, 120
IBM1927 103
IBM1936 77
IBM1948 104
IBM2063 92, 104
IBM2402 104
IBM2409 105
IBM2410 105
IBM2412 106
IBM2421 106
IBM2610 106
IBM2611 107
IBM2617 108
IBM2621 108
IBM2622 108
IBM2626 109
IBM2628 109
IBM2801 109
IBM2804 110
IBM2810 110
IBM2811 111
IBM2812 111
IBM5002 77
コンパイル単位の定義 48

[サ行]

サービス
CICS サポート 6
OS PL/I 6
再コンパイル
必要性 3
再リンク、OS PL/I-COBOL ILC の
再リンク・ツールの使用 205
サブシステムに関する考慮事項
CICS 61, 189
DB2 66, 197
IMS 63, 193
サブシステムのパフォーマンス、向上 147
サポートされない ENVIRONMENT のオプション 70
サポートされないコンパイラー・オプション 75
CONTROL 75
COUNT 75
DECK 75
ESD 75
FLOW 75
HGPR 75
IMPRECISE 75
LMESSAGE 75
SEQUENCE 75
SIZE 75
SMESSAGE 75
サポートされないファイル属性
BACKWARDS 70
EXCLUSIVE 70
TRANSIENT 70
支援技術 xvi
事前初期設定済みプログラム 45
条件
ERROR 40
FIXEDOVERFLOW 117, 129
OVERFLOW 127
UNDERFLOW 38
ZERODIVIDE 127
条件処理
重大度の相違点 42
相違点 40
タイミングの相違点 40
未処理条件の相違点 41
IBMBXITA および IBMBEER の相違点 41
IMS に関する考慮事項 65, 194
U4039 の相違点 42
シンボル・テーブル
考慮事項 59
CSECT 59
数学組み込み関数
相違点 142
数学ルーチン、OS PL/I の使用 60
ストレージ
仮想要件 16
使用法
再チューニング 145
IMS に関する考慮事項 64, 194

ストレージ (続き)
DASD 要件 16
Enterprise PL/I 要件 25
ストレージ使用効率、向上 146
ストレージ報告書の相違点 49, 143
製品間の関係
デバッグ・ツール 7
製品構成
説明 209
データ・セット
新規 7
OS/390 7

[タ行]

ダンプに関する相違点 143
データ・セット
新規、OS/390 7
ロード・モジュールに関する考慮事項 53
デバッグ・ツール
製品間の関係 7
PLITEST との比較 217
デバッグ・ツール間の相違点 217
特記事項 219

[ハ行]

配列式の制約事項 71
バッチ・コンパイル
制約事項 75
例 215
パフォーマンス
コンパイラー・オプション
DFT(NONASGN) 89
DFT(NONCONNECTED) 89
DFT(NOOVERLAP) 90
DFT(REORDER) 89, 90
NORENT 91
OPTIMIZE(2) 89, 90
REDUCE 90
RULES(NOLAXCTL) 91
再チューニング 145
ストレージ使用効率 146
CICS 環境、向上 147
CPU 使用効率 145
FIXED BIN(15) をループ制御変数として使用 126
FIXED DEC をループ制御変数として使用 126
IMS 環境、向上 147
TOTAL 環境オプション 127
プリプロセッサー
CICS プリプロセッサー 191
SQL プリプロセッサー 197
プログラム、事前初期設定済み 45

[マ行]

マイグレーション
新しいコンパイラーへの 25
アプリケーションのインベントリーの作成 17, 26
一般的作業 11
オブジェクト・モジュール再リンク・ツール 205
共用ライブラリー置換用エイド 205
コンパイラー、基本事項 4
再リンク・エイド、使用 205
実動への切り換え 24
ツールと支援機能 203
メイン・ロード・モジュールを再リンクするためのサンプル ZAP 204
ライブラリー・ルーチン置換ツール 203
ランタイム、基本事項 4
レグレッション・テスト 23
ILC に関する考慮事項 19
LE の段階的導入 19
LNKLST の使用 19
PLISRTx モジュールの再リンク 205
PL/I アプリケーションの移行 28
PL/I アプリケーションの優先順位 28
STEPLIB の使用 20
STEPLIB の例 21
マクロ・レベル・インターフェース、CICS に関する考慮事項 62, 190
マルチスレッド化
サポート 8
マルチタスキング機能
サポート 10
メイン・ロード・モジュール、ユーザー
サンプル ZAP 再リンク・エイド 204
メイン・ロード・モジュール再リンク・エイド
サンプル ZAP の使用 204
メッセージ
2603 101
EXIT オプション 112
IBM1044 97
IBM1053 97
IBM1065 98
IBM1091 98
IBM1099 98
IBM1181 100
IBM1206 100
IBM1208 101
IBM1215 102
IBM1216 102
IBM1220 103
IBM1927 103
メッセージ (続き)
IBM1948 104
IBM2063 92, 104
IBM2402 104
IBM2409 105
IBM2410 105
IBM2412 106
IBM2421 106
IBM2610 106
IBM2611 107
IBM2617 108
IBM2621 108
IBM2622 108
IBM2626 109
IBM2628 109
IBM2801 109
IBM2804 110
IBM2810 110
IBM2811 111
IBM2812 111
戻りコード 140

[ヤ行]

ユーザー情報 xiii
ユーザー出口
インストールの注意点 186
ユーザー出口 186
CEEBINT 186
CEEBXITA 186
IBMBEER 186
IBMBXITA 186
IBMFXITA 186
ユーザー戻りコードの相違点 47
ユーザー・メイン・ロード・モジュール
サンプル ZAP 再リンク・エイド 204
ユーザー・メイン・ロード・モジュールの再リンク
サンプル ZAP 再リンク・エイド 204

[ラ行]

ライブラリー・ルーチン置換ツール
IBMWRLKx の使用 203
ライブラリー・ルーチンの置換
IBMWRLKx の使用 203
ランタイム環境
PL/I の 8
ランタイムの出力、CICS に関する考慮事項 63, 191
ランタイム・オプション
相違点 37
ABTERMENC 38, 139
ALL31 38, 139, 140, 146
COUNT 37, 139

ランタイム・オプション (続き)
DEPTHCONDLMT 38, 139
ERRCOUNT 38, 139, 140
FLOW 37, 139
HEAP 37, 139, 147
ISASIZE 37
LANGUAGE 37
MSGFILE 48, 139, 142
NATLANG 37, 139
REPORT 37
RPTSTG 10, 37, 139, 145
SPIE 37
STACK 37, 139, 140, 147
STAE 37
STORAGE 139
TRAP 37, 139, 140
XUFLOW 38, 139, 140
ランタイム・メッセージ 141, 142
ランタイム・メッセージの相違点 47
リンク・エディット
および CHANGE 139
シンボル・テーブル 59
説明 59
CSECT 59
数学ルーチン、使用 60
AMODE(24) に関する考慮事項 137
LE を使用して既存アプリケーションを 34
LIMITS の影響 137
NCAL オプション 60
NCAL オプションの使用 60
PLICALLA 137
PLICALLB 137
RENT/NORENT の影響 137
ループ
無限 120
ロード・モジュール
一般的な考慮事項 53
言語環境プログラム・サポート
リリース 3.0 より前の OS PL/I バージョン 1 56
OS PL/I バージョン 1.3.0 ～ 1.4.0 55
OS PL/I バージョン 1.5.0 55
OS PL/I バージョン 1.5.1 53
OS PL/I バージョン 2 56
考慮事項
データ・セット 53
OS PL/I バージョン 2 56
IDR 情報 18, 207
PL/I バージョンの識別 18, 207

A

ABTERMENC ランタイム・オプション 38, 139
ADDBUFF 環境オプション 70
ALL31 ランタイム・オプション 38, 139, 140, 146
AMODE(24)
サポート 78
リンク・エディットに関する考慮事項 137
31 ビット・データとの混合 28
AMODE(31)
ALL31 についての考慮事項 38
APAR
前提条件として必要な言語環境 143
AREA
INITIAL 属性を持つ 131
ASCII 環境オプション 70
ASMTDLI IMS インターフェース 63, 193

B

BACKWARDS ファイル属性 70
BUFFERS 環境オプション 70
BUFOFF 環境オプション 70

C

CEEBXITA ユーザー出口 186
CEESTART
使用 51
PLICALLA 42
CEEUOPT
ランタイム・オプション 38
PLICALLB 43
CICS 環境でのアプリケーションのリンク 190
CICS に関する考慮事項
エラー処理 62
既存の CICS アプリケーションの起動 34
説明 61, 189
統合プリプロセッサー 8, 191
マクロ・レベル・インターフェース 62, 190
ランタイム出力 63, 191
CSD ファイル、更新 61, 189
Enterprise PL/I アプリケーションのリンク 190
OS PL/I に対するサポートの廃止 6
PL/I によって使用される異常終了コード 63, 191
STACK ランタイム・オプション、使用 62
SYSTEM コンパイラー・オプション 190
CMPAT コンパイラー・オプション 83, 149
DB2 に関する考慮事項 197
CONTROL コンパイラー・オプション 75
COUNT コンパイラー・オプション 75
COUNT ランタイム・オプション 37, 139
CPU 使用効率、向上 145
CSD ファイル、更新 61, 189
CSECT
シンボル・テーブル 59
IDR 情報 18, 207

D

DATE/TIME 組み込み関数 46
DB2 に関する考慮事項 66, 197
DBCS の制約事項 73
SQL プリプロセッサー 197
Debug Tool 217
DECK コンパイラー・オプション 75
DEFAULT コンパイラー・オプション
LINKAGE 86, 149
NOBIN1ARG 85
NONASGN 89
NONCONNECTED 89
NOOVERLAP 90
OVERLAP 86, 121, 149
REORDER 89, 90
DEFAULT ステートメントの制約事項 72
DEPTHCONDLMT ランタイム・オプション 38, 139
DFSORT、使用 46

E

Enterprise PL/I for z/OS
アクセシビリティー xv
Enterprise PL/I for z/OS ライブラリー xiii
Enterprise PL/I ライブラリー xiii
ERRCOUNT ランタイム・オプション 38, 139, 140
ERROR 条件 40
ESD コンパイラー・オプション 75
EXCLUSIVE ファイル属性 70
EXEC DLI インターフェース 63, 193
EXEC SQL ステートメント 197
EXTERNAL
初期化されていない STATIC 124
EXTRN コンパイラー・オプション 83, 149

F

FETCH
  コンパイラーの利点 8
FIXED BIN
  精度 7 以下の 130
  FIXEDOVERFLOW 129
FIXEDOVERFLOW
  FIXED DEC に限定 129
  SIZE 117
FLOAT
  float への代入 122
FLOW コンパイラー・オプション 75
FLOW ランタイム・オプション 37, 139

G

GONUMBER コンパイラー・オプション 96
GRAPHIC および POSITION の制約事項 73

H

HEAP ランタイム・オプション 37, 139, 147
HGPR コンパイラー・オプション 75

I

IBMBEER ユーザー出口
  インストールの注意点 186
  相違点 41
IBMBXITA ユーザー出口 186
  相違点 41
IBMFXITA ユーザー出口 186
IBMRLSLx、共用ライブラリーの置換 205
IBMWRLKx、ライブラリー・ルーチンの置換 203
IEEE 浮動小数点
  サポート 8
ILC (言語間通信)
  使用可能な言語 49
  相違点 49
  マイグレーションに関する考慮事項 19
  PLIXOPT についての考慮事項 38
IMPRECISE コンパイラー・オプション 75
IMS に関する考慮事項
  アセンブラー言語オプションのサポート 64, 194
  インターフェース 63, 193
  条件処理 65, 194
  ストレージの使用 64, 194
  説明 63, 193
IMS に関する考慮事項 (続き)
  IMS へのインターフェース 63, 193
  PLICALLA サポート 64, 193
  PSB 言語オプション 64, 194
  STEPLIB の使用と LE 21
  SYSTEM コンパイラー・オプション 64, 193
INCLUDE コンパイラー・オプション 74
INDEXAREA 環境オプション 70
ISAINC ランタイム・オプション 139
ISASIZE ランタイム・オプション 37, 139
iSUB 定義の制約事項 73

L

LABEL 配列の制約事項 73
LANGLVL コンパイラー・オプション 74
LANGUAGE ランタイム・オプション 37, 139
LIMITS コンパイラー・オプション
  EXTNAME 83, 149
    リンク・エディットに関する考慮事項 137
LIST コンパイラー・オプション 74
LMESSAGE コンパイラー・オプション 75
LNKLST
  マイグレーションでの使用 19
  SCEERUN 4, 7
LPALST
  マイグレーションでの使用 19
LRECL
  コンパイラー SYSPRINT 77

M

MSGFILE ランタイム・オプション 48, 139, 142

N

NATLANG ランタイム・オプション 37, 139
NCAL リンケージ・エディター・オプション 60
NCP 環境オプション 70
NOREDUCE コンパイラー・オプション 86, 90
NORENT コンパイラー・オプション 84, 91, 149, 180
  リンク・エディットに関する考慮事項 137
NOWRITABLE コンパイラー・オプション 84, 87, 181
NOWRITE 環境オプション 70

O

OPTIMIZE コンパイラー・オプション 89, 90
OS PL/I
  サービス 6
  バージョン 2 のロード・モジュール 56
OVERFLOW 条件 127

P

PLICALLA エントリー・ポイント
  サポート 42
  IMS に関する考慮事項 64, 193
PLICALLB エントリー・ポイント
  サポート 43
PLIDUMP
  生成される出力 48
  相違点 48, 143
PLIMAIN エントリー・ポイント 51
PLISRTx モジュール再リンク・ツール 205
PLISRTx、使用 46
PLISTART
  エントリー・ポイント 51
  PLICALLA 42
PLITDLI IMS インターフェース 63, 193
PLITEST
  デバッグ・ツールとの比較 217
PLIXHD 79, 143
PLIXOPT
  ランタイム・オプション 38
  PLICALLB 43
PREFIX コンパイラー・オプション 96
PSB 言語オプション、IMS に関する考慮事項 64, 194
PTF
  前提条件として必要な言語環境 143

R

RECORD I/O
  制約事項 70
REDUCE コンパイラー・オプション 90
REENTRANT プロシージャー・オプション 73
REGIONAL 環境オプション 70
RENT コンパイラー・オプション 84, 180
  リンク・エディットに関する考慮事項 137
REPORT ランタイム・オプション 37, 139

RPTSTG ランタイム・オプション 37, 139
ストレージの調整に使用 16, 145
RULES コンパイラー・オプション
ANS 99
LAXCTL 87
LAXSTRZ 95
NOLAXCTL 91
NOLAXDCL 92
NOLAXIF 93
NOLAXMARGINS 94
NOMULTICLOSE 95

S

SCEELKED
構成 7
非 IBM 名 34
SCEERUN
構成 7
LNKLST の 4
STEPLIB または JOBLIB 内 59
SEQUENCE コンパイラー・オプション 75
SIS 環境オプション 70
SIZE
FIXEDOVERFLOW 117
SIZE コンパイラー・オプション 75
SKIP 環境オプション 70
SMESSAGE コンパイラー・オプション 75
SPIE ランタイム・オプション 37, 139
SQL プリプロセッサー
解除された制限 197
使用 197
新規、統合 8
EXEC SQL ステートメント 197
STACK ランタイム・オプション 37, 62, 139, 140, 147
STAE ランタイム・オプション 37, 139
STATIC
書き込み可能な再入可能 8
使用されない INTERNAL の保持 117
初期化されていない EXTERNAL 124
STEPLIB
マイグレーションでの使用 20
マイグレーションの例 21
STMT コンパイラー・オプション 75
STORAGE ランタイム・オプション 139
STREAM I/O
印刷不能な文字 124
制約事項 71
SYSLIN DD
制約事項 77
SYSPRINT
新旧の PL/I 間での共用 150
SYSPRINT (続き)
LRECL 値 77
MSGFILE(SYSPRINT) に対するサポート 48, 142
SYSTEM コンパイラー・オプション 75, 85
CICS に関する考慮事項 190
IMS に関する考慮事項 64, 193

T

TEST コンパイラー・オプション 96
TOTAL 環境オプション 70, 127
TP 環境オプション 70
TRANSIENT ファイル属性 70
TRAP ランタイム・オプション 37, 139, 140
TRKOFL 環境オプション 70
TSO 76

U

U4039 ABEND 42
UNDERFLOW 条件 38
UNICODE
サポート 8
UNLOCK ステートメント 70

V

VM
サポート 10

W

WRITABLE コンパイラー・オプション 84, 181

X

XUFLOW ランタイム・オプション 38, 139, 140

Z

ZAP、メイン・ロード・モジュール再リンク・エイド 204
ZERODIVIDE 条件 127

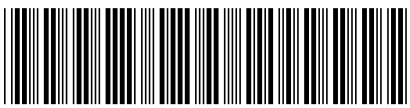
[特殊文字]

(NO)GOSTMT 75
(NO)GOSTMT コンパイラー・オプション 75

プログラム番号: 5655-PL5

Printed in Japan

GC43-3420-00



日本アイ・ビー・エム株式会社
〒103-8510 東京都中央区日本橋箱崎町19-21