14 章　言語間連絡の拡張

この章では、言語間連絡を主題として、連絡機能の拡張について説明します。

14.1 連絡機能の拡張

次に COBOL2002 での連絡機能の拡張を、項目毎に見て行きます。

（１）PROCEDURE DIVISION や CALL 命令で RETURNING の指定が可能になりました。前者は組み込み関数の様な値を返却するプログラム、関数を COBOL で記述出来る様になったという事です。また後者もその様な関数タイプの手続きを CALL 命令で呼び出す事が出来ます。

（２）PROCEDURE DIVISION や CALL 命令のパラメータ受け渡し方式として、BY VALUE の指定が可能になりました。従来の規格では BY REFERENCE と BY CONTENT だけでしたが、この２方式は両方、アドレス渡し方式です。従来規格ではポインタデータも無いし、アドレスを渡している意識は特に無かったかも知れません。REFERENCE と CONTENT の差は前者が呼び出し側のデータそのものを渡すのに対し、後者はデータのコピーを渡します。
これらを纏めると次の表になります。

	REFERENCE	CONTENT	VALUE
渡し方	アドレス	アドレス	データ値
渡す物	オリジナル	コピー	コピー

表 14.1　パラメータの受け渡し方式

（３）PROCEDURE DIVISION や CALL 命令で各パラメータの前に OPTIONAL 指定が出来、パラメータが省略可能である事を表現出来ます。また引数が省略されているかどうかを下記形式の「引数省略条件」で判定出来ます。

データ名　IS　[NOT]　OMITTED

例えば次の様に、パラメータを最大３つ渡せるが、必須なのは先頭パラメータだけである様なプログラムを作る事が出来ます。

```
PROCEDURE   DIVISION   USING
                SYORI-SYUBETU
                OPTIONAL   PARAM-2
                OPTIONAL   PARAM-3.
    IF   PARAM-2   IS   OMITTED
        第２パラメータ省略時の処理
    END-IF   …
```

（４）コンパイル指示子 CALL-CONVENTION が指定出来ます。これはプログラムやメソッドの相互作用の詳細を決定するものです。つまりプログラム呼び出し規約の種類を指定出来ると言う事です。詳細については処理系作成者定義になります。

（５）GOBACK 命令が新設されました。これは EXIT PROGRAM と STOP RUN の機能を併せ持つ働きをします。つまり呼ばれたプログラム内では EXIT PROGRAM と同様の動作を行い、メインプログラムでは STOP RUN と同じになります。

（６）プログラム・プロトタイピング（原型）機能が利用出来ます。環境部・構成節の「REPOSITORY」段落でプロトタイプ宣言を行う事によって、下記の様なプログラムの性質をチェック可能になります。

パラメータの渡し方(BY VALUE 他)
パラメータの省略可否
パラメータの型
呼び出し規約

＜使用例＞

```
IDENTIFICATION  DIVISION.
PROGRAM-ID. proto-program-name-1  IS  PROTOTYPE.
DATA  DIVISION.
LINKAGE  SECTION.
01  param-1    PIC  ...
01  param-2    PIC  ...
01  return-val    PIC  ...
PROCEDURE  DIVISION   USING  BY  VALUE  param-1,  OPTIONAL   param-2
                              RETURNING  return-val.
END PROGRAM  proto-program-name-1.

IDENTIFICATION  DIVISION.
PROGRAM-ID. main-program-name-1.
ENVIRONMENT  DIVISION.
CONFIGURATION  SECTION.
REPOSITORY.
      PROGRAM  proto-program-name-1.
PROCEDURE  DIVISION.
      CALL  "proto-program-name-1"  USING  BY VALUE argument-1,  OMITTED
                                    RETURNING return-code.
```

＜説明＞

上記の翻訳グループは２つの翻訳単位で構成されています。第１の翻訳単位はプロプラム原型定義であり、第２は従来通りのプログラム定義(メイン・プログラム)です。メイン・プログラムの REPOSITORY 段落では、何処かで定義されたプロプラム原型情報を参照する事を宣言し、このプログラム内で記述されている CALL 命令でのプログラム性質のチェックを可能にしています。

（７）PROGRAM-POINTER 型データを定義出来ます。このデータ項目は CALL, INVOKE 等で使用でき、主にオブジェクト指向機能と結びついています。
OBJECT REFERENCE 型データも同じ目的のものです。

（８）連絡関係の例外
EC-PROGRAM... EC-ARGUMENT... EC-FLOW など。

14.2 言語間連絡の拡張について

ここで説明している連絡機能の殆どは、COBOL 言語内だけの機能にとどまりません。COBOL 間での連絡機能も勿論であり、その機能範囲が広がった事も重要な点ですが、他言語との連携の可能性を広げたというポイントも見逃せません。特に BY　VALUE 方式によるパラメータ渡しや、呼び出し規約の指定は現在幾つかの他言語で利用されている手法であり、COBOL にこの様な機能が取り込まれた結果として、呼び出される側、または呼び出す側としても COBOL 以外の言語で作ったモジュールを想定する事が容易になります。また既に一定のレベルで実装し始めている処理系も有ります。

またオブジェクト指向機能の詳細についてはここでは説明しませんが、この機能も他言語連絡も見据えて来ています。当然仕様策定の上では既存のオブジェクト指向言語(C++,Java 等)の仕様も踏まえ、同等の機能や考え方が多く取り入れられている事は言うまでも有りません。これは仕様的な整合性を取るという意味ばかりでなく、実装に当たって相互の連携利用の道を開くものと言えます。つまり C++から COBOL で作ったオブジェクトを呼び出したり、COBOL プログラムから Java　Beans の機能を利用したりする可能性が予測されます。

特に CORBA(Common Object Request Broker Architecture)や「.net」等最近の業界動向の中でフレームワーク（枠組み、構成）の統合、連携の方向性が強く出てきており、これら枠組みの中に COBOL という言語開発手法も浸透して行く可能性を予感させるものであると言う事も出来ます。