連載

今からでも遅くない！ RPGⅢユーザーに贈る

RPGⅣの魅力と可能性

第５回　RPGⅣのフリーフォームの特長と魅力

中村 潤
株式会社アイ・ラーニング
IT 研修本部　IBM 製品研修部
ラーニング・アドバイザー

RPG Ⅳのよさを伝えたい 年の初めに思いを新たに

この原稿は１月中旬に執筆しているので新年のご挨拶としては遅いですが、本年もよろしくお願いいたします。

今年は初詣に松陰神社へ行ってきました。最近、NHK大河ドラマの関係で吉田松陰が話題に上りますが、松陰の故郷の山口県萩市ではなく、都内にある松陰神社です。そこには松陰のお墓があります。松陰は行動力に富み、また学問の教師としても非常に優れた人でした。とりわけ教え子たちの長所を見つけることに長けていたと言われます。私たちインストラクターにとっても、松陰に見習うべき点がたくさんあると思っています。

リスト1　KINGAKUの各要素に対し、指標と同じ値を代入

```
FOR  I = 1 TO 50;
  KINGAKU(I) += 1;
ENDFOR;
```

リスト2　Xが100よりも大きくなるまで、変数URIAGEの値をGOUKEIに累積

```
DOU     X > 100;
  GOUKEI += URIAGE;
  X      += 1;
ENDDO;
```

新年から、IBM i関連の研修コースを受講される方も数多くいらっしゃいます。私はRPGⅢのコースを担当しました。RPGⅣについて連載しているのにRPGⅢを教えているのか、とお叱りを受けるかもしれませんが、現実問題として、RPGⅢのユーザーはまだ数多くおられます。

ここで誤解していただきたくないのは、RPGⅢを使い続けるのはいけないということではけっしてありません。研修を受講し、新しいRPGプログラマーが育っていくことは、それがRPGⅢであってもRPGⅣであっても、非常に喜ばしいことです。若いRPGプログラマーが後に続かず、現役RPGプログラマーがリタイアするとそこでRPGもIBM iも終わり、というのはあまりに寂しすぎるお話です。

RPGもIBM iも新しい技術に十分すぎるほどに対応しているのに、それに気づかない方が思いのほかに多いことは、本誌前号（11月号）の特集でも話題になっていました。今まで培ってきたRPGⅢ資産を基に若いプログラマーにRPGⅣを習得してもらうことは、IBM iをこれからも長く使い続ける上で非常に重要です。吉田松陰が好きな孟子の言葉、「至誠にして動かざるものは未だこれあらざるなり」に無理やり結びつけるわけではありませんが、これからもRPGⅣのよさを訴え続けていきたいと思います。

一般的なプログラミングと RPGプログラミングの違い

今回は、RPGⅢのフリーフォームについてお話ししま

す。フリーフォームについてはこれまで何度かお話ししてきましたが、あらためてそのよさを考えてみましょう。**リスト1**をご覧ください。

オープン系のプログラマーなら、このコードを見て何のことかさっぱりわからない、と言う方はまずいないでしょう。配列KINGAKUの各要素に対し、指標と同じ値を代入しています。

また、**リスト2**はいかがでしょうか。

リスト3 一般的な報告書印刷

```
0001.00 H DATEDIT(*YMD)
0002.00 FTOKMSP    IP   E           K DISK
0003.00 FQPRINT    O    F  132        PRINTER OFLIND(*INOF)
0004.00
0005.00 ITOKREC        01
0006.00
0007.00 C     TKGEND         SUB        TKUZAN         PRRSAG             9 0
0008.00 C                    ADD        1              TOTKEN             5 0
0009.00 C                    ADD        TKGEND         TOTEND             9 0
0010.00 C                    ADD        TKUZAN         TOTZAN             9 0
0011.00 C                    ADD        PRRSAG         TOTSAG             9 0
0012.00
0013.00 OQPRINT    H    1P                        02
0014.00 O         OR    OF
0015.00 O                                           6 'ILE01R'
0016.00 O                                          14 'EDXXX'
0017.00 O                                          53 ' **得意先マスター +
0018.00 O                                              一覧表 ** '
0019.00 O                                          75 ' 日付 '
0020.00 O                        *DATE         Y   86
0021.00 O                        PAGE               94
0022.00 O                                          99 ' 頁 '
0023.00 O          H    1P                     04
0024.00 O         OR    OF
0025.00 O                                           8 ' 得意先 '
0026.00 O                                          26 ' 得意先名ー漢字 '
0027.00 O                                          37 ' 郵便 '
0028.00 O                                          56 ' 信用限度額 '
0029.00 O                                          69 ' 売掛金残高 '
0030.00 O                                          83 ' 差額 '
0031.00 O          H    1P                     05 07
0032.00 O         OR    OF
0033.00 O                                           8 ' 番 号 '
0034.00 O                                          37 ' 番号 '
0035.00 O          D    01                      2
0036.00 O                        TKBANG             6
0037.00 O                        TKNAKJ            30
0038.00 O                        TKPOST            38
0039.00 O                        TKGEND        J   55
0040.00 O                        TKUZAN        J   69
0041.00 O                        PRRSAG        J   83
0042.00 O          T    LR                      2
0043.00 O                                          10 ' 最終計 '
0044.00 O                                          29 ' レコード件数 '
0045.00 O                        TOTKEN        1   35
0046.00 O                        TOTEND        J   55
0047.00 O                        TOTZAN        J   69
0048.00 O                        TOTSAG        J   83
```

これも、もうおわかりですね。DOU命令が見慣れないかもしれませんが、DOUNTILのこと、と言えば納得していただけると思います。変数Xの値が100よりも大きくなるまで処理を繰り返し、変数URIAGEの値をGOUKEIに累積します。

RPGのフリーフォームでは、このようにJavaやVBなどとさほど大きな違いがなく、そちらの経験者であればまったく違和感なく取り入れられる言語となっています。そもそもRPGは、Report Program Generatorの略称で、報告書を作成するための言語でした。RPGにはRPG内部論理があり、それを使用すればCOBOLよりもはるかに少ないステップでプログラムを書くことができ、COBOLと比べものにならないくらい開発効率がよいというものでした。RPG内部論理を使ったプログラミングはRPGⅢでも使われ、RPGⅣでもRPGⅢから移行してきた場合はお使いの方が多いと思います。今回は少々逆戻りの感がありますが、この項でもRPGⅣでRPG内部論理を使ったプログラミングをご紹介します。

リスト3は、データベース・ファイルを1件読み取っては1件印刷する、を繰り返す、報告書印刷でよく見かけ

リスト4　RPG内部論理を使ったプログラミング　～報告書印刷

```
0001.00 H DATEDIT(*YMD)
0002.00 FJUMEIL01  IP   E           K DISK
0003.00 FQPRINT    O    F  132        PRINTER OFLIND(*INOF)
0004.00
0005.00 IJUMEIR        01
0006.00 I                                      JDTOKB         L2
0007.00 I                                      JDCHUB         L1
0008.00
0009.00 C   L2              Z-ADD     *ZERO          L2@KING           9 0
0010.00 C   L1              Z-ADD     *ZERO          L1@KING           9 0
0011.00 C                   ADD       1              COUNT             5 0
0012.00 C                   ADD       JDKING         L1@KING           9 0
0013.00 CL1                 ADD       L1@KING        L2@KING           9 0
0014.00 CL2                 ADD       L2@KING        LR@KING           9 0
0015.00
0016.00 OQPRINT    H    L2                        02 04
0017.00 O          OR   OFNL2
0018.00 O                                       6 'ILE02R'
0019.00 O                                      53 ' 得意先別受 +
0020.00 O                                          一覧表 '
0021.00 O                                      70 ' 得意先   ='
0022.00 O                      JDTOKB          75
0023.00 O           D    01                  2
0024.00 O                L1    JDCHUB          10
0025.00 O                      JDKING       J  22
0026.00 O           T    L1                   2
0027.00 O                                      10 ' 受注計 '
0028.00 O                      L1@KING      J  22
0029.00 O           T    L2                   2
0030.00 O                                      10 ' 得意先計 '
0031.00 O                      L1@KING      J  22
0032.00 O           T    LR                   2
0033.00 O                                      10 ' 最終計 '
0034.00 O                      LR@KING      J  22
```

るパターンです。報告書印刷では、「ファイルを読む」「印刷をする」、命令で言うと、READ、EXCEPTの書き出し命令、DOなどの繰り返し命令を使うのは当たり前ですが、その命令を使わなくてもいいようにRPGはできています。つまり演算仕様書には、読んでから書き出すまでの演算を記入すればよいだけです。

また、制御グループの処理（ブレイク処理）の場合も、制御レベルのフィールドにL1、L2などの制御レベル標識を指定すれば、その標識のオン／オフで、制御の切れ目（ブレイク）が判断できました。次の例では、入力仕様書（仕様書コード：Ⅰ）で中計にJDTOKB、小計にJDCHUBを指定し、それぞれ制御レベルの標識L2、L1を使用しています。L2、L1がオンになった時がブレイクが起こった時で、演算仕様書では条件付けの標識にL2、L1のついたコードが同一グループの先頭行での処理で、ステートメント7、8桁目の制御レベルにL2、L1のついたコードが、同一グループの合計処理になります**（リスト4）**。

マッチングの場合も同様に、マッチングフィールドを指定すればよいだけです。マッチングの場合は、プライマリーファイルとセカンダリーファイルを指定します。下記の例では、プライマリーファイルのJUMEIL01のフィールドJDTOKBと、セカンダリーファイルのTOKMSPのフィールドTKBANGを付き合わせ、JDTOKBとTKBANGの値がマッチングしていればMR標識をオンにします。セカンダリーファイルのTOKMSPを読み込んだ時にMR標識がオンであれば、JUMEIL01の情報と、TOKMSPの情報を、JUMIDPに書き出しています**（リスト5）**。

プログラム論理や全体の組み立てを演算仕様書で考慮しなくていい

ここまでは、懐かしのRPGプログラミングです。いえ、今も多くの方がこの方法でRPGプログラムを書いているはずですので、懐かしの、なんて言い方をすると非常に失礼なのですが、このRPG内部論理を使ったプログラミングは、プログラム論理や全体の組み立てを演算仕様書で考慮する必要のないことが魅力です。

RPG内部論理を使えばEXCEPT命令を使う必要がありません。EXCEPTとは「例外」という意味ですが、書き

リスト5　RPG内部論理を使ったプログラミング　～マッチングの場合

```
0001.00 H DATEDIT(*YMD)
0002.00 FJUMEIL01  IP   E           K DISK
0003.00 FTOKMSP    IS   E           K DISK
0004.00 FJUMIDP    O    E             DISK
0005.00
0006.00 IJUMEIR        01
0007.00 I                                        JDTOKB          M1
0008.00 ITOKMSR        02
0009.00 I                                        TKBANG          M1
0010.00
0011.00 C                   IF        *IN01
0012.00 C                   ADD       JDKING         JHKING
0013.00 C                   ADD       1              JHGYOS
0014.00 C                   ENDIF
0015.00
0016.00 C                   IF        *IN02 AND *INMR
0017.00 C                   MOVE      JDTOKB         JHTOKB
0018.00 C                   MOVE      JDCHUB         JHCHUB
0019.00 C                   MOVE      TKTIKU         JHTIKU
0020.00 C                   WRITE     JUMIDR
0021.00 C                   CLEAR                    JUMIDR
0022.00 C                   ENDIF
```

出し命令を使って書き出し処理を行うことは、RPGの世界では「例外」だったということです。私も昔は上司から「EXCEPTを使ってプログラムを書くやつがあるか」と叱られたものです。このように必要最小限の命令でプログラムは書けました。

RPGプログラマーはRPGの内部論理の仕組みをしっかり理解していなければなりません。全体的なステップ数は減らせても、案外と覚えることは多いのです。それを覚えてしまえばプログラミングは至って簡単なのですが、今度は他のプログラマーから「何の処理をしているかさっぱりわからない」と言われてしまいます。

そうした背景もあり、年を経るごとに、新規のRPGプログラマーは、この内部論理を使ったプログラミングから遠ざかっていきます。結果的にはCOBOLとよく似た形のコードができ上がってしまい、初期のRPGプログラミングのスタイルとは似て非なるものになってしまったような印象を受けます。

RPG Ⅳのフリーフォームのプログラミング

そしてRPG Ⅳにおけるフリーフォームのプログラミングで、RPGはまったく姿を変えたと言ってよいでしょう。バージョン7.1 TR7（Technology Refresh 7）からは演算仕様書だけでなく、制御仕様書、ファイル仕様書、定義仕様書もフリーフォームが使えるようになりました。もはやステートメント6桁目の仕様書コードも姿を消してしまいました。次のコードをご覧ください。制御仕様書、ファイル仕様書、定義仕様書のフリーフォームです（リスト6）。

制御仕様書はRPG Ⅳになってからキーワード記入形式になったので、さほど違和感はないですね。ファイル仕様書は少々驚きかと思います。DCL-Fとファイル名、たったこれだけでよいのです。ファイル仕様書のデフォルトは、入力ファイル、全手順ファイル、キー順アクセス、データベースとなっていますので、ほとんどのプログラムでファイル仕様書はファイル名のコーディングだけでよいのです。定義仕様書もCLプログラムのDCLコマンドと似たような形で指定します。DCL-Sが独立フィールドの指定、DCL-Cは固定情報の指定です。数値フィールドの場合、DCL-S でフィールド名を指定し、BINDECキーワードで桁数、小数点以下の桁数を指定します。

ここで注意しなければならないのは、フリーフォームは入力仕様書と出力仕様書をサポートしていないことです。対話型プログラムで使用する表示装置ファイルの扱いにはさほど影響はありませんが、出力仕様書を使えないということは、報告書印刷プログラムには外部記述印刷装置ファイルを使用するか、出力仕様書のみは従来どおりの定位置記入形式にする必要があります。またファイル仕様書ではデータベース・ファイルは全手順でのアクセスのみ可能で、RPG内部論理を使用してのプライマリーファイル、セカンダリーファイルの指定ができないので、フリーフォームではRPG内部論理は使用できないことになります。RPG内部論理と分離したことにより、RPGは今までとまったく違った言語に生まれ変わり、JavaやVBと同じようなコードを羅列しながら、データベース・ファイルを読み取る、出力装置に書き出す、というプログラム論理を構築していく言語になっているのです。

だからと言って不安になる必要はありません。RPGの面白いところは、フリーフォームと今までの定位置記入の仕様書が混在するプログラムがあっても大丈夫なことで、プログラム中のほかのすべてがフリーフォームに変わっても、出力仕様書はそのまま残しておくこともできます。前回にも申し上げたことですが、一度に全部変える必要はないのです。ゆっくりフリーフォームに移行していきながら、定位置記入とフリーフォームの両方を書けるプログラマーを育てていくやり方もあると思います。オープン系のプログラマーであればフリーフォームから入ったほうが習得が早いのは明確ですから、そこを入り口として定位置記入を習得するというやり方も効果的です。今年こそ、RPG ⅢからRPG Ⅳへ一歩踏み出してみませんか。

リスト6　拡張演算項目2でIF命令を使用

```
CTL-OPT DATFMT(*ISO) TIMFMT(*ISO);
DCL-F   TOKMSP;
DCL-S   URIAGE BINDEC(9:0)
```