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８回　若い人の声に耳を傾けるべき時がきた

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

ILE COBOLコースも必要に?

１年を振り返るには、まだ少し早い気もしますが、10月ともなると１年もあと２カ月少々、今年の目標に向かってスパートをかけていると、あっという間に年末になってしまいます。私も数カ月おきにこの連載を書いていると、前回からそんなに日にちが経ったのかと驚くことが多くあります。

この１年を振り返ると、今年はCOBOL研修を担当する機会があり、何年かぶりかでCOBOLを勉強し直すということがありました。私が担当するCOBOLですので、当然ながらIBM iのCOBOLですが、他社マシンからIBM iに乗り換えるユーザーが、COBOLのままIBM iに入ってくる事例が数多く見られます。こういう事例が多くあると、ILE COBOLのコース開催が必要かもしれないと、考えさせられたりもしました。RPGだけでなく、COBOLも言語の選択肢の一つであるという、IBM iの多様性を物語る話ですが、RPGⅣを語る身としては、やはりRPGの話にも耳を傾けてほしく思います。

JavaやCとの隔たり

これまでRPGⅣの素晴らしさを連載してきましたが、RPGⅢ人気はまだまだ根強くあります。RPGⅣはこれだけ素晴らしい言語なのに、なぜRPGⅢを使い続けているのですか？と、当方としては訴えているつもりですが、RPGⅢユーザーからすれば「RPGⅢで問題なくプログラムもシステムも動いているのに、なぜわざわざRPGⅣで作り直さなければいけないのか」が正直な気持ちと思います。まったくその通り

図表１　Javaのコーディング法

```
Class  Iteration1  {
      public  static  void  main(String[]  args)  {
          for  ( int i  =   0;  I  <  5;  i++)  {
              System.out.println( “Hello!” );
          }
      }
 }
```

図表２　Cのコーディング法

```
  int i;
  for (i = 1; i <= 5; i++)
printf( “Hello!\n” );
```

です。変える必要もないのに、わざわざ変えるシステム担当者はあまりお見かけしません。しかし、これまで何度かお話ししてきたように、一番大きな問題は後継者問題です。

RPGは、発表当初はツールに近い簡易言語で、帳票印刷プログラムを作成することに特化した言語でしたが、バージョンを重ねるごとに機能が強化され、データベースを使用したバッチ処理、対話型処理にも優れた言語となりました。

RPG仕様書に記入する方式で、必要な場所に必要なことを書いていく、いわば穴埋め方式のコーディングでプログラムが書けるので、他の言語より習得が速いことが特徴でした。一昔前に専門学校や大学で教えられてきた言語はCOBOLであり、Fortranであり、アセンブラでしたが、COBOLを勉強した人は、ほぼ間違いなくRPGに対応できました。COBOLは固定位置記入の言語ではありませんが、使っている命令がREADであったりWRITEであったり同じ名前のものが多く、ファイル処理を行える言語としても、共通部分が数多くあったためです。

しかし近年は、若い人たちが勉強する言語はJava**（図表1）**やC**（図表2）**などオープン系言語が中心です。これらの言語は、命令というより構文でプログラムを構築し、ステートメントの固定位置に決まった情報を記述するRPGとはスタイルがまったく違うため、RPGは他の言語と共通部分が少ない孤立した言語、というイメージが強くなりました。若いプログラマーのRPG離れが余儀なくされる状況になり、当社の受講者の中には「今度、RPGを担当することになってしまった」と、RPGについて否定的な言い方をされる方がいます。もちろん、担当になったからには真剣に講義を聴き、一生懸命に勉強されるのですが……。

上司・先輩がフリーフォームを嫌がる

若いプログラマーにとって一番わかりやすいRPGのスタイルは、フリーフォームのRPGであることは言うまでもありません。見た目が、JavaやCと違和感がまったくありません**（図表3）**。

私も長年、RPGプログラミングの研修を担当してきましたが、RPGの初級者研修では、**図表3**のような「Hello!World」的なプログラムは書かないのです。初級者研修の初日から、オーバーフロー標識を使った帳票印刷プログラムを作成します。プログラミング経験のない方が悪戦苦闘することは言うまでもありませんが、固定位置記入で「Hello!World」的なプログラムを書いても、次のステップにつなげることができず、やっても仕方がないのが大きな理由です。RPGプログラムでは、命令をたくさん覚えることが重要なのです。

しかし、フリーフォームプログラミングでは、「命令を覚える」のではなく、JavaやCと同じく「構文」でプログラムを作成でき、JavaやCを知っている人であればすぐに理解でき、初めてプログラムを勉強する人にとっても難解ではありません。

アイ・ラーニングの研修でも「RPGⅣバッチ・プログラム演習」「RPGⅣ対話型プログラム演習」で、フリーフォームのプログラミングを解説していますが、若いプログラマーは圧倒的にこのスタイルに興味をもちます。私もその言葉を聞くたびにうれしく思うのですが、次に皆さんから出てくる言葉が「上司、先輩がフリーフォームを嫌がる」というもので、これを聞くと考え込まざるを得なくなります。長年私たちが使いこなしてきた固定位置記入のRPGが異端児扱いされるのは寂しい話ですが、若い人たちの声に耳を傾けなければならない時が来ているようです。

RPGⅢ技術者にも簡単なフリーフォームRPG

ベテランのRPGプログラマーにはフリーフォームのRPGがRPGとはまったく別の言語のように見えるようです。その気持ちは十分に理解できます。何よりも私自身がそうだったのですから。しかしフリーフォームのRPGをよく見ると、今までのRPGⅢプログラマーにもなじみのスタイルであることに気づくはずです。**図表4**のプログラムをご覧ください。

フリーフォームRPGは、V7R1 TR7以降、制御仕様書、ファ

図表1　Javaのコーディング法

```
Dcl-s  i  packed(1:0);
For   i  =   1  by  1  to   5;
       Dsply  ('Hello!');
Endfor;
*inlr = *on;
```

イル仕様書、定義仕様書、演算仕様書がフリーフォーマット化され、完全なフリーフォーム化が実現しました。演算仕様書だけでも難解だったのに、完全フリーフォーム？ 全部？と思われる方も多いと思いますが、しかしよくよく見ると、見覚えのあるようなロジックではありませんか。

2、3行目のDCL-F文は、ファイル仕様書がフリーフォームになったものです。しかしDCL-Fは今まで、CLプログラムのファイル宣言でもDCLFコマンドを使っていました。それと同じ感覚で書くことができます。DCL-Fの直後に、ファイル名、その次に装置名というルールは存在しますが、省略時の値はDISK(*EXT)で、外部記述データベースファイルになります**（図表5）**。

また、変数定義も、DCL-S文で行えるようになりました。先のDCL-Fと同じく、宣言はすべてDCL-XXで行い、DCL-C：名前付き定数定義、DCL-PARM：パラメーター定義、DCL-DS:データ構造となります。定義部はすべてDCL-XXで統一され、E仕様書、I仕様書、C仕様書のあちこちで定義を行う必要があったRPGⅢとは、大きく様変わりしています。

さらに、演算命令の構文も、命令、演算項目1、演算項目2、結果のフィールドの順番に記入するようになり、最後にセミコロン「;」で終わります。最初は戸惑うかもしれませんが、RPGを知っている人であれば、慣れるのに時間はかかりません**（図表6）**。

この文法を見ていくと、完全フリーフォームであっても、解読不可能ということは絶対にないはずです。もちろんフリーフォームに乗り換えるには、まずRPGⅢからRPGⅣに乗り換える必要がありますが、それだけでも多くのメリットがあります。

フィールド名が6文字までの不自由さ

RPGⅢからRPGⅣへ移行するメリットを挙げると、次の

図表4　フリーフォームRPG

```
0001.00   CTL-OPT DATEDIT(*YMD);
0002.00   DCL-F   TOKMSP;
0003.00   DCL-F   QPRINT  PRINTER(132) OFLIND(*INOF)
0004.00
0005.00   DCL-S   PPRSAG  ZONED(9:0);
0005.01   DCL-S   TOTKEN  ZONED(5:0);
0007.00   DCL-S   TOGEND  LIKE(TKGEND);
0008.00   DCL-S   TOUZAN  LIKE(TKUZAN);
0009.00   DCL-S   TOTSAG  LIKE(TKGEND);
0010.00
0011.00   /FREE
0012.00
0013.00        EXCEPT  MIDASI;
0014.00        READ    TOKMSP;
0015.00
0016.00        DOW NOT %EOF;
0017.00           PPRSAG =  TKGEND  -  TKUZAN;
0018.00            TOGEND += TKGEND;
0019.00            TOUZAN += TKUZAN;
0020.00
0021.00            IF     *INOF;
0022.00                EXCEPT    MIDASI;
0023.00            ENDIF;
0024.00            EXCEPT    MEISAI;
0025.00           READ      TOKMSP;
0026.00
0027.00       ENDDO;
0028.00
0029.00        TOTSAG =  TOGEND  -  TOUZAN;
0030.00        EXCEPT  GOUKEI;
0031.00        *INLR   = *ON;
0032.00        RETURN;
```

ような点が指摘できます。

・関数を使用し、より生産性向上（日付計算は命令で可能）。
・サブプロシージャ、サービスプログラムの使用によって、パフォーマンスの向上
・フィールド名が最大4096文字で命名可能になり、6文字命名の制約からの解放

こうして書き並べると、RPG Ⅲでフィールド名が6文字までしか命名できないことが非常に窮屈で、若いプログラマーに一番敬遠される要因ではないかと思えます。関数の使い方などは、自らコーディングすることである程度解決できるでしょうが、これだけはいくら努力してもどうにもなりません。

6文字しか使えないことで、配列などは1文字か2文字くらいの名前にしないと指標を使えません。ネーミングルールを決めるにしても、かなりの制約があると思います。RPG Ⅳであればフィールドの命名は4096文字まで可能で、さすがにそんなに長い名前を付けることはないにしても、20文字程度の名前がつけられるのは、自由度がまるで違うように感じられると思います。

RPG Ⅳプログラミングの自由度は、若いプログラマーだけでなく、ベテランプログラマーにももたらされることで、RPG ⅢからRPG Ⅳへ移行して問題となることは何ひとつないと断言できます。若い人たちに、RPG Ⅳ、ひいてはフリーフォームRPGの門をぜひ開いてあげていただきたく思います。

お知らせ

アイ・ラーニングでは現在、IBM i関連の研修コースを東京と大阪で実施していますが、これ以外の都市での開催を検討しています。詳細は、決定次第、Webでお知らせしますので、ご期待ください。

図表5 ファイル仕様書の記述例

```
DCL-F   TOKMSP;
DCL-F  TOKMSP   DISK(*EXT) USAGE(*INPUT);
```

※上記2行はまったく同じ意味です。

```
DCL-F  TOKMSP   KEYED;
```

※キー順アクセスパスを使用。

図表6 演算命令のフリーフォーマット構文例

```
CHAIN   JHTOKB   TOKMSP;
IF   NOT   %FOUND(TOKMSP);
     MSG   =   ‘ 得意先番号エラー ‘;
ENDIF;
```