新連載

今からでも遅くない！ RPG Ⅲユーザーに贈る

RPG Ⅳの魅力と可能性

第１回　RPG ⅢとRPG Ⅳのコーディングの違い

太田 賢
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って、簡単！

RPG Ⅳをまだ使用したことのない方のために、RPG Ⅳのコーディング・スタイルについて簡単にご紹介してみたいと思います。

RPG ⅣはRPG Ⅲより言語のバージョンが1つ新しいというだけで、同じ言語です。従来のRPGとほとんど同じ感覚でプログラミングできるのはもちろんですが、よりプログラミングしやすく改良されています。RPG ⅣはRPG Ⅲと比べて構造化プログラミングを強く意識しているため、より自然で読みやすいコーディングができるようになっています。

コーディング・スタイルの変化

具体的に見てみましょう。たとえばREAD命令やCHAIN命令でファイル読み取りを行うとき、従来は結果の標識で読み取り結果を判断していました。標識の使用はコーディングを簡潔にするRPGの特長ですが、標識の多用により、プログラムの保守性が低下する問題も抱えていました。RPG Ⅳでは標識を使わずに結果を判断できます。結果の標識を1つも使わずにコーディングを行うことが可能です。**図表１**を見てください。

このコーディング例では、CHAIN命令でHINMSPというファイルの読み取りを行っていますが、結果の標識は指定していません。その代わりにIF命令で%FOUNDという組み込み関数を使用して結果を判断しています。%FOUND組み込み関数は直前CHAIN命令の結果を判断して、読み取れれば'1'(真)を返します。*IN99などの予約語も使用する必要がなくなるため、「99は何だっけ？」などと考えなくても済み、プログラムを自然に記述し読むことができます。

それでは今度は**図表２**を見てください。

「（A ＋ B ＋ C）÷ D」という演算を行うとき、従来のRPGでは**図表２**のようなコーディングを行っていたと思い

図表１　結果の標識を使わないコーディング

```
HINBAN CHAIN    HINMSP
       IF       %FOUND
       EXSR     @ARI
       ELSE
       EXSR     @NASHI
       ENDIF
```

図表２　従来のRPGの演算式

```
A      ADD   B          WORK
       ADD   C          WORK
WORK   DIV   D          RESULT
```

ます。しかしRPG Ⅳでは演算で式を使用できるため、これを**図表3**のように1ステートメントで記述できます。

式は、IF命令やDOW、DOU命令でも使用できます。式を使用すると**図表4**のように、複数の条件を簡単に記述することが可能です。

図表4は「標識03がオンか、標識12がオンになるまで繰り返す」という意味です。「*IN03」「*IN12」と標識だけを書いていますが、これはそれぞれ「*IN03がオンのとき」、「*IN12がオンのとき」という意味で、**図表4**の条件は「*IN03 = *ON　or *IN12 = *ON」と書いても同じ意味です。しかし、**図表4**の方が簡潔なコーディングだと言えるでしょう。

「標識*IN03がオフのとき」という条件であれば、「*IN03 = *OFF」と書くか、または簡潔に「NOT *IN03」とコーディングすることができます。なお、式を書いている演算仕様書の項目名は、拡張演算項目2といい、従来の演算項目2よりも記述できる桁数が長くなっています。

フリーフォーマットのコーディング

図表3でEVALという命令が出てきましたが、これは式を評価するための命令でRPG Ⅳから追加された命令です。四則演算を式で書くときには演算命令としてEVAL命令を書きます。しかし、この命令はなくてもいいような気がするかもしれません。式だけで十分意味はわかります。

そこで、EVAL命令も省略する方法があります。そのためには、フリーフォーマットのコーディングというものを行います。これは従来のRPGが定位置記入方式といって、何桁目に何を書くかが細かく決められていたのに対して、ステートメントを何桁目から書き始めてもよい、という自由な演算仕様書のコーディング方法です。プログラマーが見やすいようにインデントしてコーディングできる方法です。

フリーフォーマットのコーディングでは、まず最初に演算命令を書きます。その後に演算項目1、演算項目2、結果のフィールド、という順序でコーディングします。**図表5**を見てください。

図表5は、**図表1**のコーディングをフリーフォーマットでコーディングしたものです。CHAIN命令が最初に記述されていること、演算項目1のHINBANがCHAIN命令の後にあることに注意してください。また、各ステートメントの終わりにはセミコロン(;)が記述されています。IF命令の中のコーディングがインデントされていることにも注意してください。

図表3　RPG Ⅳの演算式

```
EVAL   RESULT = ( A+B+C ) / D
```

図表4　複数の条件式を使用したコーディング

```
DOU *IN03  or *IN12
```

図表5　フリーフォーマットによるコーディング

```
CHAIN HINBAN HINMSP;
IF %FOUND;
  EXSR @ARI;
ELSE;
  EXSR @NASHI;
ENDIF;
```

図表6　フリーフォーマットによる図表3 演算式のコーディング

```
RESULT = ( A+B+C ) / D;
```

図表7　従来のRPGのコーディング

```
*INOF   IFEQ    *ON
        WRITE   MIDASHI
```

図表8　オーバーフロー標識に任意の名前を付けられる

```
IF OVERFLOW;
  WRITE MIDASHI;
ENDIF;
```

このフリーフォーマットを使用するとき、EVAL命令は省略してもよいという決まりがあります。**図表3**のコーディングは**図表6**のようになります。

もう1つ例を見てみましょう。**図表7**を見てください。

オーバーフロー標識OFがオンだったら見出しを書き出す、というコーディングです。次に**図表8**を見てください。

OVERFLOWという記述が出てきましたが、これは名前付き標識です。RPG Ⅳではオーバーフロー標識に任意の名前を付けることができます。ここでは省略しますがファ

イル仕様書でオーバーフロー標識の名前を自由に指定できるのです。標識OFを使うよりも、自然な言語としてプログラムを読めるのではないでしょうか。

図表9は、フリーフォーマットでコーディングした対話型プログラムの演算仕様書の例です。インデントが効果的に使用されている点や、コメントが演算命令と同じ行に入れられる点などに注意してください。また、「NOTFOUND」という記述がありますが、これは名前付き標識です。ここでは省略していますが、定義仕様書（D仕様書）でエラーを画面に表示するために表示装置ファイルのDDSで定義した標識に名前を付けています。

演算命令の桁数の変更

RPG Ⅲでは演算命令が5桁だったため、演算命令のもとになる英語単語が不自然に短く縮められていました。RPG Ⅳでは桁数が10桁になったため、**図表10**のように自然な命令になりました。

RPG制限の変更

桁数が増えたのは演算命令だけではありません。**図表11**

図表9　フリーフォーマットのコーディング例

```
/FREE
 // メイン ルーチン
 DOW  NOT EXIT;

      WRITE  RCD03;        // 指示画面の表示
      EXFMT  RCD01;
      IF   EXIT;      //F3キーの検査
          EXSR @LAST;
      ENDIF;

      CHAIN HINBAN HINMSP;    //   品目マスターの読み取りと処理

      IF %FOUND;             // レコードが見つかったときの処理

           NOTFOUND=*OFF;
         WRITE   RCD01;
           EXFMT   RCD02;

           IF EXIT;            // F3キーの検査
               EXSR @LAST;
           ENDIF;

       ELSE;                 //レコードが見つからなかったときの処理
           NOTFOUND=*ON;
       ENDIF;

 ENDDO;
 // サブルーチン

 BEGSR  @LAST;
     *INLR=*ON;
     RETURN;
 ENDSR;
 /END-FREE
```

のように扱うことのできるファイル名、レコード様式名、フィールド名なども桁数が拡張されています。

ILE RPGとRPG Ⅳは同じもの?

「ILE RPGとRPG Ⅳは同じものですか?」という質問をよくいただきます。この質問には「同じものです」と答えるようにしています。しかし、厳密に言えば「ILEに対応したRPGのバージョンがRPG Ⅳである」ということで、意味するところは少し違います。

ILE RPGとは、ILE(統合言語環境)という言語環境に対応したRPGを意味しています。IBM iはILEという言語環境を持っていて、この環境に対応した言語では、異なる言語で書かれたモジュールを組み合わせて、1つのプログラムとして実行できます。ILE RPG、ILE COBOL、ILE C、ILE CLがILEに対応した言語です。さまざまな言語のプログラマー、あるいは、さまざまなベンダーが開発したプログラムを、1つのアプリケーションで同時に容易に利用できる、便利で柔軟で強力な環境がILEです。

RPG ⅣのⅣ(フォー)とは、RPGという言語のバージョンです。ⅢよりⅣの方がバージョンが新しいので、機能が多く、またコーディングの仕方が今回一部をご紹介したように異なるところがあります。

今回はRPG Ⅳの部分に焦点を当てました。ILEについては、また機会がありましたらご紹介させていただきたいと思います。なお、ILEに対して従来の環境をOPM(オリジナル・プログラム・モデル)と呼んでいます。ILEとOPMはともに1つのIBM iシステムに共存できるので、RPG Ⅳのプログラムも、RPG Ⅲのプログラムも、同時に1つのシステムで使用できます。一度にすべてのアプリケーションをRPG Ⅳに変更する必要はなく、新しいアプリケーションから徐々にRPG Ⅳで開発していくことが可能です。

まとめ

ここまで見てきた例で、従来の書き方と新しい書き方のどちらがコーディングしやすく、読みやすいと思われたでしょうか。RPGのベテランプログラマーの方は、従来の書き方が馴染み深く、好まれる方も多いかもしれません。しかしRPGを初めて学ぶ方であれば、新しい書き方のほうが簡単で、わかりやすいと思われるでしょう。そしてRPGのベテランの方でも、慣れると新しい書き方のほうが書きやすくなってくるのではないでしょうか。

つまりRPG Ⅳのコーディングについてまず言えることは、コーディングは難しくなくわかりやすいということです。RPG Ⅳにはこのほかにもさまざまな新しい機能があり、使いこなすことで、新しいアプリケーション開発の世界が広がります。ぜひRPG Ⅳのコーディングを学んでみてください。

手前味噌で恐縮ですが、株式会社アイ・ラーイングではRPG Ⅲをご存じの方のためのRPG Ⅳの入門書『RPG Ⅲ経験者のためのRPG Ⅳ—自習方式』をご提供しています(税別価格7500円。http://bit.ly/1fHhTd0)。スキルアップにご活用いただければ幸いです。

図表10 RPG ⅢとⅣの演算命令

RPG Ⅲ	RPG Ⅳ
EXCPT	EXCEPT
LOKUP	LOOKUP
SETOF	SETOFF
RETRN	RETURN
DELET	DELETE

図表11 RPG制限の変更

項目	RPG Ⅲ	RPG Ⅳ(IBM i 7.1)
フィールド/配列名	6	4096
データ・ストラクチャー名	6	4096
レコード様式名	8	10
ファイル名	8	10
最大使用可能ファイル数	50	制限なし
最大文字フィールド長	256	16773104
名前付き固定情報	256	16380
データ・ストラクチャーのサイズ	9999	16773104
小数点以下の桁数	9	63
配列の要素の数	9999	16773104
サブルーチンの数	254	制限なし
プログラム・サイズ	制限あり	制限なし