Tutorial 2 – Logic Control Statements in /Free

Now that we have a simple program going, let's throw some logic control statement in here. specifically, we will look at If, Select, and For. Ready?

IF

The If statement in /Free works much like the one in regular RPG except for the differences.

The basic key words in it are the same; IF, ELSE, and ENDIF. Just remember that there will be semicolon after each one of these.

The big difference is the way the first line of the IF is written. Rather than the Teutonic, inverted order of fixed format RPG, it will use more straightforward verbiage. For example,

```
IF Dave > *Blanks;
```

If there is more than one clause, like if you have multiple values for what you want Dave to be, then you need to repeat the field name like

```
IF Dave = '1' OR Dave = '2';       this could also be written as

IF Dave = '1' OR
   Dave = '2';
```

Note that on the second form you put the semicolon at the end of the clause, not the end of the line. Either way is OK, depending on how you like to see things structured. The main point is that there are no continuation characters required. The semicolon defines where a statement ends and another begins, not whether you switch to a new line or not. Also note that the form IF Dave = '1' OR '2'; is not valid. I am kind of sorry about that but it doesn't work and that's the fact.

You can have multiple clauses but they must be linked together by a logical operator (AND or OR), and in those cases it is best to use parenthesis to help you keep things straight.

IF (Dave = ‘1’ OR Dave=’2’) and (Tom = ‘1’ OR Tom = ‘4’);

There are some differences in the logical operators we can use. The *EQ, *LT, etc. are not supported in /Free. Instead you need to use the arithmetic symbols of >, <, =, etc. I know what you’re thinking. How then do I express a ‘not’ condition? There is no key that represents a not equal to or not greater than, etc. Let’s take an example.

IF Dave = ‘1’;

And say we want to really say if Dave is not equal to ‘1’. Then we would do this.

IF Not (Dave = ‘1’);

I know, I wish they would have given us something like ‘not=’ or ‘NE’ or something. But they didn’t. Get over it. I did.

You use an ENDIF; to end the If statement. And in between you can put in whatever execution statements you want. Just remember to put a semi-colon after each of these execution lines. And remember to indent. This is one of the most important advantages to using /Free, the ability to indent which makes it easier to see the logic flow so don’t forget to do that. The code doesn’t force you to indent, and there are no rules about how much you indent. I like four spaces but that’s just me. Now take your program and insert an IF statement and try things out.

```
0023.00 D DAVE             S              10
0023.01 D TRUE             S               1
0024.00 D
0025.00  /free
0026.00
0027.00   // Initializing Fields
0028.00     EVAL DAVE = 'Nice Guy';
0028.01     EVAL TRUE = 'N';
0028.02
0028.03   // Check value and assign Truth value.
0028.04     IF (DAVE = 'Nice Guy') OR
0028.05        (DAVE = 'OK Guy');
0028.06         EVAL TRUE = 'N';
0028.07     ELSE;
0028.08         EVAL TRUE = 'Y';
0028.09     ENDIF;
0029.00
```

Fig 1 - /Free with IF

Do Stuff

The same basic facts are true for DOW and DOU loops.

You use the same operators (DOW, DOU, and ENDDO) as you do in fixed format but you write the statement like an English sentence.

DOU Dave = '11';

Yada, yada, yada; ← see the indenting? Use it.

ENDDO;

As with the IF statement you can use parenthesis and logical operators to write more complex conditions, and you can split them up on multiple lines with a semicolon only at the end of the whole DO clause (and also after the ENDDO and any imperative statements).

```
FMT ** ... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+.
0027.00   // Initializing Fields
0028.00     EVAL DAVE = *BLANKS;
0028.02
0028.03   // LOOP THROUGH PRODUCT MASTER ASSIGNING DAVE VALUE BASED ON DEMAND
0028.04   // TYPE.
0028.05     READ MSPMP100;
0028.06
0028.07     DOW NOT (%EOF);
0028.08         IF PM_MSFLG = 'I' OR
0028.09            PM_MSFLG = 'C';
0028.10             EVAL DAVE = 'NICE GUY';
0028.11         ELSE;
0028.12             EVAL DAVE = 'GROUCHY GUY';
0028.13         ENDIF;
0028.14
0028.15         READ MSPMP100;
0028.16     ENDDO;
```

Fig 2 – DO Loop with an IF.

Select

I like the SELECT statement, it reminds me of the old Case structure command on the IBM 8100. Anyone remember that machine? It was sort of a mainframe AS/400. Simpler than CICS but an integrated development environment and very powerful. For some reason after a fast start it just sort of petered out. I think the release of the 4300 series might have had something to do with that, plus all of the minicomputers like the VAX and what not sort of declined about that time. Great machine though.

Anyway, the select statement is great when you have a number of options for your If and you want something that isn't going to indent you to death. I use it if I have more than two clauses for my IF. The format is quite simple.

```
SELECT;
    WHEN logical statement;
        imperative statement;
    WHEN logical statement;
        imperative statement;
    OTHER;
       imperative statement;
ENDSL;
```

The logical statements in the WHEN clause do not have to have anything in common. They don't need to reference the same variable (although often they do). In addition they can be as complex as you want them to be with logical operators and parenthesis.

The OTHER keyword is used alone (no logical statement) and it is a catchall for things that don't fit into one of the previous WHENs.

```
SELECT;
    WHEN Dave = '1';
        do something
    WHEN Dave = '1' AND Tom = '4';
        do something else
    WHEN OTHER;
        do a third thing
ENDSL;
```

```
0029.02   // LOOP THROUGH PRODUCT MASTER ASSIGNING DAVE VALUE BASED ON DEMAND
0029.03   // TYPE.
0029.04     READ MSPMP100;
0029.06     DOW NOT (%EOF);
0029.07         SELECT;
0029.08             WHEN PM_MSFLG = 'I';
0029.09                 EVAL DAVE = 'NICE GUY';
0029.10             WHEN PM_MSFLG = 'C';
0029.11                 EVAL DAVE = 'NICE GUY';
0029.12             WHEN PM_MSFLG = 'C';
0029.13                 EVAL DAVE = 'GROUCHY GUY';
0029.14             OTHER;
0029.15                 EVAL DAVE = *BLANKS;
0029.16         ENDSL;
0029.23
0029.24         READ MSPMP100;
0029.25     ENDDO;
```

Fig 3 – Do Loop with Select.

FOR

There is also a FOR statement which allows you to do a loop x number of times. This seems very FORTRAN to me and that is OK, but I don't seem to have much call to do a loop a given number of times so I am not going to deal with this. It's there though if you need it.

FOR i = 1 to 5;

your indented logic statements

ENDFOR;

The from and to limits can also be variables.

(PS – strangely enough, almost as soon as I had written this statement, I found a spot where using the FOR was perfect. One thing I will mention about it, is that You do not need to increment the i within the loop. That is done automatically by the FOR. Only an idiot would set one up with an i increment embedded in it and I would hate to see you make the same mistake I did.)

```
0022.00 D
0023.00 D DAVE            S             10
0024.00 D DAVEA           S              1    DIM(5)
0025.00 D I               S              2  0
0026.00  /free
0027.00
0028.00   // CHECK VALUES IN DAVE ARRAY
0029.00     EVAL I = 1;
0030.00     FOR I = 1 TO 5;
0031.00         IF DAVEA(I) = 'Y';
0032.00             EVAL DAVE = 'NICE GUY';
0033.00         ELSE;
0034.00             EVAL DAVE = *BLANKS;
0035.00         ENDIF;
0036.00     ENDFOR
0037.00
```

Fig 4 – For loop. Do you see the syntax error that I included in here? I did that on purpose. No, seriously, I did. I did. On purpose.

(I forgot the semicolon on the ENDFOR. On purpose.)

Leaving the Loop

Sometimes you are in an If statement or a loop and you decide you just want to get out. What we generally do is set a flag and fudge the structure of the loop so that when this flag is set you can exit. Of course, back in the olden days we would just do a GOTO and get out but that was abused and has rightfully been banished.

Fortunately, /Free supports the LEAVE and ITER op codes. LEAVE will take you to the next statement after the end of the loop you are currently in. ITER cause you to jump to the ENDDO (or ENDFOR) and perform the normal checking that is done to see if you should get out or do another iteration. In other words, LEAVE does just that, it kicks you out of the loop you are in. ITER bypasses all of the logic between you and the end of the loop but keeps you in the loop and lets it do the normal checking to see if you should be set free or if you have another ticket to ride.

Both of these op codes work only if you are in a DO or FOR loop. You can code it inside an IF or SELECT statement within such a loop and that is OK, but if you just put it in a standalone If or SELECT, you will get a compile error. I have no idea.

Indent

While we are talking about logic statements, I want to say just a word about indenting your code.

Do it!

Indenting is something RPG programmers generally don't think about too much (since you can't do it in fixed format mode). But indenting is probably the single most powerful structure tool a programmer has and anyone who has struggled to match up IF's and ENDIF's in a big logic section has to appreciate the ability to be able to indent code and so easily see which ENDIF matches up with which IF.

So get in the habit right now. I like to indent four spaces, some people like more or less. The important point is to start to build indenting into your coding style with /Free right from the start. You won't regret it.

The End

So how are you feeling now? It's pretty simple, really. This stuff at least. Try adding some logic statements of each type to your simple /Free program. Don't be afraid of a few complier errors. They teach you things. Right.

Anyway, the important thing is that now it's time to march into your bosses office and tell him you finally want to be paid what you're worth. Ready?

What now? IO statements? Good grief, Linus. At this rate this is going to take forever. OK, I have to remember the old Computer Associates motto; 'The Damn Customer is always right'. So let's go on to Tutorial 3 and learn about IO.

Questions or Comments?

For more information on how /Free can help you, contact Shirey Consulting Services at 616-452-6398 or support@shireyllc.com.