



Deep dive into RPG free-form

Barbara Morris
IBM



A fully-free RPG program, starting in column 1

```
**free
ctl-opt bnddir('ACCRCV');

dcl-f custfile usage(*update);
dcl-ds custDs likerec(custRec);
dcl-f report printer;

read custfile custDs;
dow not %eof;
  if dueDate > %date(); // overdue?
    sendOverdueNotice ();
    write reportFmt;
    exec sql insert :name, :duedate into
      mylib/myfile;
  endif;
  read custfile custDs;
enddo;
inlr = '1';

dcl-proc sendOverdueNotice;
  /copy invoices
  sendInvoice (custDs : IS_OVERDUE);
end-proc;
```

First line has **FREE
All free-form statements
No fixed-form allowed

Fully free-form RPG

PTFs for 7.1 and 7.2 provide the ability to code free-form RPG starting in column 1 and going to the end of the line.

There is no practical limit on the length of a source line.

- CRTSRCPF has a limit of 32766
- IFS files have no limit

Fully free-form RPG – how long should your lines be?

Various style-guides for other languages recommend a maximum line length of 80, 132, 120 etc.

The “80” comes from IBM punch cards.

Google [maximum length of a code line] to see some of discussions about line length.

If you create your RPG source files with RCDLEN(112), then that gives you 100 characters, which is probably ideal.

Fully free-form RPG – source must start with **FREE

Any source member that contains fully-free code must have **FREE in column 1 of the first line of the source.

```
**FREE  
ctl-opt main(greeting);  
  
dcl-proc greeting;  
    dsply 'Hello';  
end-proc;
```

Fully free-form RPG

- All code in a `**FREE` source member must be free-form. If you need any fixed-form code, you can put it in a `/COPY` file
- Source lines must not begin with `**` unless they are the special directives for compile-time data, file-translation, or alternate collating sequence.
- `/FREE` and `/END-FREE` are not allowed in a `**FREE` source member

Fully free-form RPG – copy files

- Each copy file has its own source mode
- A copy file is always assumed to have column-limited source mode unless it has `**FREE` in line 1

Fully free-form RPG – RDI

RDI supports fully-free RPG code (SEU does not)

The screenshot displays the RDI interface for a file named *TEST.RPGLE. The main editor window shows the following code:

```

Line 9      Column 4      Replace 7 changes
...+....1....+....2....+....3....+....4....
000100  **free
000101
000102  dcl-s quantity int(10);
000103  dcl-s quality char(4);
000104  dcl-s price  packed(5 : 2);
000105  dcl-s total  packed(7 : 2);
000107
000108  for i = 1 to 3;
000109      q
000
  
```

The word 'quality' is highlighted in the code, and a yellow tooltip box shows the word 'quality' with a pink label 'keyw' next to it. The Outline window on the right shows the following structure:

- Global Definitions
 - Fields
 - quantity : Integer (10,0)
 - quality : Character (4)
 - price : Packed Decimal (5,2)
 - total : Packed Decimal (7,2)
- Main Procedure

Fully free-form RPG – Embedded SQL

The SQL precompiler supports fully-free RPG code

```
**FREE  
dcl-s greeting char(10);  
  
exec sql set :greeting = 'Hello';  
dsply greeting;  
return;
```

What is wrong with fixed-form code?

- Most programmers today have never seen fixed form code
- When they see RPG code like this, it looks like gibberish

```
H bnddir('ACCRCV') dftactgrp(*no)
Fcustfile  if  e                disk
Freports   o  e                printer
```

- Here's what happens when a non-RPG programmer tries to make a change

```
H bnddir('ACCRCV')
Fcustfile  if  e                disk
Freport    o  e                printer
RNF0289E Entry contains data that is not valid; only valid data is used.
RNF2013E The Device entry is not PRINTER, DISK, SEQ, WORKSTN or SPECIAL;
         defaults to DISK.
RNF2003E The File Type is not I, O, U, or C; defaults to O if File
         Designation is blank, otherwise to I.
RNF2005E The Sequence entry is not blank, A, or D; defaults to blank.
... more error messages
```

RPG is still not 100% free

There are still some areas where RPG is not free (and may never be)

- I specs and O specs must be coded in fixed-form
 - I and O specs are considered deprecated by many RPG programmers in favor of externally-described files
- Code related to the RPG cycle must be coded in fixed-form
 - The cycle is considered deprecated by many RPG programmers in favor of using SQL for scenarios where the cycle formerly shone

What does an all-free RPG mean?

- Fewer "secret codes" to remember ("E in column 19 means externally-described")
- Indented code is more maintainable
- Better token-colorization in the RDI editor, allowing programmers to have the same look-and-feel for RPG code as for other languages like Java or PHP
- New programmers will only have to learn how to use RPG, without having to struggle with how it is coded

Removal of many frustrations

- /FREE and /END-FREE in every procedure
- Two lines for many definitions in fixed-form

```
D getNextCustomer...
D                               pr
```

vs

```
dcl-pr getNextCustomer;
```

- Insufficient room in D-spec keywords for long strings

```
D HSSFCellStyle      c
D
D 'org.apache.poi.hssf.-
  userModel.HSSFCellStyle'
```

vs

```
dcl-c HSSFCellStyle 'org.apache.poi.hssf.usermodel.HSSFCellStyle';
```

More information

Documentation

- The ILE RPG Reference in the 7.2 and 7.3 Knowledge Center has all the information about free-form. The free-form information also applies to 7.1.

RPG Café wiki page with PTF information:

https://ibm.biz/rpgcafe_fullyfree_rpg

Conversion

- RDI free-form conversion does not do any conversion from H F D P to free-form.
- ARCAD has a product that converts H F D C and P specs to fully-free-form.
- Linoma's conversion tool converts H F D C and P specs to fully-free-form.

The details

Let's look at the details

- General features
- Control (H)
- File declaration (F)
- Data declaration (D)
- Procedure (P)

Some general features

The new statements all

- Start with an "opcode"
- End with a semicolon

Just like calculation statements in RPG:

```
if duedate > today;  
    sendAngryLetter (customer);  
endif;
```

Some general features

Unlike free-form calculations, can have /IF, /ELSEIF, /ELSE, /ENDIF within a statement

```

[ dcl-s salary
  /if defined(large_vals)
    packed(13 : 3)
  /else
    packed(7 : 3)
  /endif
  ;
]
```

Some general features

Can mix fixed-form and free-form without /FREE and /END-FREE

Example: Defining the TAG for SQL "whenever"

```
    exec sql whenever sqlerror goto err;  
    ...  
    return;  
C      err          tag  
    ok = *off;  
    reportSqlError ();
```

Control statements

CTL-OPT (Control Option) statement

- Start with CTL-OPT
- Zero or more keywords
- End with semicolon

```
ctl-opt option(*srcstmt : *nodebugio)  
          dftactgrp(*no);
```

Control statements

- Can have multiple CTL-OPT statements
- The rules about not repeating keywords apply across all statements

```
ctl-opt; // no keywords
ctl-opt option(*srcstmt : *nodebugio)
           dftactgrp(*no); // two keywords
H datfmt(*iso) text('My Program')
  ctl-opt alwnull(*usrctl); // free again
```

Control statements

One little enhancement for free-form H:

If there is at least one free-form control statement, you don't need DFTACTGRP(*NO) if you have one of the ACTGRP, BNDDIR, or STGMDL keywords

File statements

DCL-F (Declare file) statement

- Start with DCL-F
- File name
- Keywords
- End with semicolon

File statements

- Only full-procedural and output – no cycle, RAF or table files
- The name can be longer than 10 as long as there's an EXTFILE keyword (and an EXTDESC keyword if externally-described)

```
dc1-f year_end_report printer  
      oflind(overflow)  
      extdesc('YERPT')  
      extfile(*extdesc);
```


File statements – the device

Device keyword or LIKEFILE must be the first keyword

DISK, PRINTER, SEQ, SPECIAL, WORKSTN

- Defaults to DISK

Externally-described: *EXT (default)

Program-described: record-length

```
dc1-f orders; // defaults to DISK(*EXT)
dc1-f qprint printer(132);
dc1-f screen workstn; // defaults to *EXT
```

File statements – the usage

USAGE keyword

*INPUT, *OUTPUT, *UPDATE, *DELETE

Equivalent of fixed-form File Type (I, O, U, C) and File-Addition

Default for USAGE depends on the device

```
dc1-f orders disk; // *INPUT
dc1-f report printer; // *OUTPUT
dc1-f screens workstn; // *INPUT : *OUTPUT
```

- SEQ and SPECIAL default to USAGE(*INPUT)

File statements – the usage

Some usage values imply other values

*UPDATE implies *INPUT

*DELETE implies *UPDATE and *INPUT

```
// USAGE(*INPUT : *UPDATE)  
dc1-f orders disk usage(*update);
```

```
// USAGE(*INPUT : *UPDATE : *DELETE)  
dc1-f arrears disk usage(*delete);
```

Can specify implied values explicitly too

```
dc1-f orders disk usage(*update : *input);
```

File statements – the usage

If you specify the USAGE keyword, the defaults are not considered

```
// output only
```

```
dc1-f f1 disk usage(*output);
```

```
// input and output
```

```
dc1-f f2 disk usage(*input : *output);
```

File statements – difference for *DELETE

In fixed form, U enables update and delete

In free form, *UPDATE does not enable delete

- *DELETE must be coded explicitly

File statements – Keyed files

For externally-described files, KEYED keyword

```
dc1-f orders disk keyed;
```

For program-described files, KEYED(*CHAR:len)

```
dc1-f generic disk(2000) keyed(*CHAR:100);
```

File statements – Program-described keyed files

Only character keys supported for program-described

For other types, use a data structure

```
dc1-f generic disk(2000) keyed(*CHAR:7);
```

```
dc1-ds key len(7) qualified;  
    item_num packed(12);  
end-ds;
```

```
key.item_num = 14;  
chain key generic;
```

File statements

F specs can be mixed with D specs (even in fixed form).

Group related items together

```
[ dcl-f orders
    usage (*update : *output) keyed;
  dcl-ds orders_dsi
    likerec (ordersR:*input);
  dcl-ds orders_dso
    likerec (ordersR:*output);
  dcl-s num_orders int(10);
```

```
[ dcl-f report printer;
  dcl-ds report_ds
    likerec (reportR:*output);
```


File-related data structures must be defined after the file

If your file has a related data structure such as an INFDS or INDDS, the data structure must be coded after the file.

Bad: The DS is defined before the file

```
dc1-ds orders_infds;  
    status *status;  
end-ds;  
dc1-f orders infds/orders_infds);
```

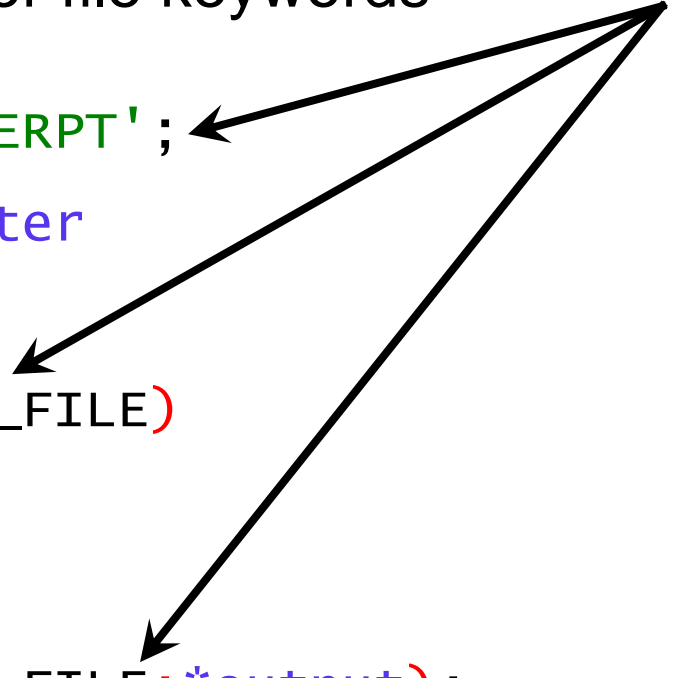
Good: The DS is defined after the file

```
dc1-f orders infds/orders_infds);  
dc1-ds orders_infds;  
    status *status;  
end-ds;
```

File statements

Named constants can be used for file keywords

```
dc1-c YEAR_END_RPT_FILE 'YERPT';  
dc1-f year_end_report printer  
oflind(overflow)  
extdesc(YEAR_END_RPT_FILE)  
extfile(*extdesc);  
dc1-ds report_ds  
extname(YEAR_END_RPT_FILE:*output);
```



Data definition statements

- Start with DCL-x
- Item name – can be *N if not named
- Keywords
- End with semicolon

```
dcl-s name like(other_name);
```

Standalone fields

The first keyword must be a data-type keyword.

```
dc1-s salary packed(9:2) inz(0);
```

If you are using the LIKE keyword, it doesn't have to be first.

```
dc1-s annual_salary inz(0)  
      like(salary : +2);
```

Data-type keywords

Some data-type keywords match the Data-Type entry exactly

CHAR, INT, POINTER . . .

Some merge the Data-Type entry with another keyword

VARCHAR = A + VARYING

DATE = D + DATFMT

OBJECT = O + CLASS

Data-type keywords – String data types

Fixed length:

- CHAR(characters)
- GRAPH(characters)
- UCS2(characters)

Varying length

- VARCHAR(characters)
- VARGRAPH(characters)
- VARUCS2(characters)

Varying length with specific prefix-size

- VARCHAR(characters : 4)
- VARGRAPH(characters : 4)
- VARUCS2(characters : 4)

Indicator

- IND

Data-type keywords – Numeric data types

Decimal types with default zero decimal positions:

- PACKED(digits)
- ZONED(digits)
- BINDEC(digits) ("BINDEC" is explained on the next slide)

Decimal types with specific decimal positions

- PACKED(digits : decimals)
- ZONED(digits : decimals)
- BINDEC(digits : decimals)

Integer, unsigned, float

- INT(digits)
- UNS(digits)
- FLOAT(bytes)

Other data types

Date, time, timestamp with default format

- DATE
- TIME
- TIMESTAMP

Date, time

- DATE(*YMD-)
- TIME(*HMS:)

Pointer and procedure pointer

- POINTER
- POINTER(*PROC)

Object

- OBJECT(*JAVA : CLASS) (parameters not needed for the prototype of a constructor)

Tip for remembering the data-type keywords

If there is a related built-in function, the data-type keyword has the same name:

%CHAR	- CHAR and VARCHAR
%GRAPH	- GRAPH and VARGRAPH
%UCS2	- UCS2 and VARUCS2
%DATE	- DATE
%TIME	- TIME
%TIMESTAMP	- TIMESTAMP
%INT	- INT
%UNS	- UNS
%FLOAT	- FLOAT

Exception: %DEC. The decimal data types are PACKED, ZONED, BINDEC.

Data structures

Data-structures end the subfield list with END-DS

- not used for LIKEDS or LIKEREK data structures

END-DS is optionally followed by the DS name

```
dcl-ds info;  
  name varchar(25);  
  price packed(4 : 2);  
end-ds info;
```

If no subfields, code END-DS on the DCL-DS line

```
dcl-ds prt_ds len(132) end-ds;
```

Data structures

DCL-DS is used to begin a data structure.

END-DS is not used if LIKERECD or LIKEDS is used (because you can't code additional subfields)

```
dcl-ds info likeds(info_t);  
dcl-ds custInDs likerec(custrec : *input);
```

END-DS is needed for an externally-described DS

```
dcl-ds custDs extname('CUSTFILE') end-ds;
```

Subfields

Subfields officially start with the DCL-SUBF opcode

The opcode is optional unless the name is the same as a free-form opcode

```
dcl-ds info;  
    name char(25);  
    dcl-subf select int(10);  
end-ds info;
```

DCL-SUBF must be used because "select" is an opcode supported in free-form

Same as the rule for EVAL and CALLP

```
name = 'sally';  
eval select = 5;
```

Subfields

The POS keyword replaces

- From-and-to positions
- OVERLAY(dsname)

```

D info      DS
D sub1      25    34A
D sub2      D    OVERLAY(info:100)
D sub3      5P 2  OVERLAY(info)

```

```

dc1-ds info;
  sub1 char(10) pos(25);
  sub2 date pos(100);
  sub3 packed(5 : 2) pos(1);
end-ds info;

```

Subfields

Free-form OVERLAY only overlays subfields

- No free-form equivalent for OVERLAY(ds:*NEXT)
- OVERLAY(ds:*NEXT) means "after all previous subfields" which is the same as not having the OVERLAY keyword at all
- SUB3 starts at position 101, after all previous subfields.

```

D info      DS
D sub1      1  100A
D sub2     11  20A
D sub3      5A  OVERLAY(info:*next)

```

Equivalent:

```

dc1-ds info;
sub1 char(100) pos(1); // 1-100
sub2 char(10) pos(11); // 11-20
sub3 char(5); // 101-105

```

Nested data structures

If you define your data structure in free-form, you can code nested data structures directly

- Define your subfield using DCL-DS and END-DS

```
dcl-ds info qualified;  
    num_employees int(10);  
    dcl-ds employees dim(100);  
        name varchar(25);  
        salary packed(6:2);  
    end-ds;  
end-ds;
```

PTF information is at
http://ibm.biz/spring_2017_rpg_enhancements

PSDS and INFDS

Use the PSDS keyword to define a program-status data structure.

Use values like *STATUS to define the special PSDS or INFDS subfields.

```
dc1-ds statusDs PSDS;  
    moduleStatus *STATUS;  
end-ds;
```

```
dc1-f myfile INFDS(myfileInfds);
```

```
dc1-ds myfileInfds;  
    myfileStatus *STATUS;  
end-ds;
```

Prototypes and procedure interfaces

Prototypes and procedure interfaces are similar

```
dc1-pr qcmdexc extpgm;  
    cmd char(3000);  
    cmd_len packed(15 : 5);  
end-pr;
```

Bonus feature:
EXTPGM parameter
is optional

```
dc1-pr init end-pr; // no parameters
```

```
dc1-pr init;  
end-pr; // can be a separate statement
```

```
dc1-pi *n varchar(25); // name not needed  
    id int(10);  
end-pi;
```

When do you need a name for the procedure interface?

A "cycle-main" procedure is the procedure coded before any subprocedures are coded.

If you need a prototype for your main procedure (probably in a /COPY file)

- Then you need to give a name for the PI so the RPG compiler knows which prototype to use

```
ctl-opt dftactgrp(*yes);  
dcl-pr subproc end-pr;  
dcl-pr mypgm end-pr; // "Main" prototype  
dcl-pi mypgm end-pi; // "Main" PI
```

```
subproc(); // These are the calculations  
return; // for the main procedure
```

... Subprocedures follow the main procedure ...

When do you need a name for the procedure interface?

If your program is **never** going to be called from another RPG program or module

- If it is always called from a CL program
- If it is the command-processing program for a command

Then you don't need a prototype for the program

(Otherwise, you do need a prototype, and it should be in a /copy file)

If you don't need a prototype, just code *N as the name for the procedure interface

```
ctl-opt dftactgrp(*yes);  
dcl-pr subproc end-pr;  
dcl-pi *N end-pi; // "Main" PI (no PR)
```

*DCLCASE for external procedure names

A common bug:

- EXTPROC is needed for the mixed-case name
- The programmer uses copy-paste and forgets one change

Bug!

```
D Qc3EncryptData...
D           pr      extproc('Qc3EncryptData')
D Qc3DecryptData...
D           pr      extproc('Qc3EncryptData')
```

Use *DCLCASE to avoid retyping the name:

```
dc1-pr Qc3EncryptData extproc(*dc1case);
dc1-pr Qc3DecryptData extproc(*dc1case);
```

- Less error prone when coding
- Easier for code reviewers to see that it's correct

Parameters

Parameters officially start with DCL-PARM

DCL-PARM is optional. Same rule as for subfields

```
dcl-pr proc;  
    name char(25) const;  
    dcl-parm clear ind value;  
end-pr;
```

Procedure statements

Begin a procedure

- DCL-PROC
- Procedure name
- Keywords
- End with semicolon

```
dcl-proc myProc export;
```

End a procedure

- END-PROC
- Optional procedure name
- End with semicolon

```
end-proc myProc;
```

or

```
end-proc;
```

Procedure example

```
dcl-proc getCurUser export;  
  dcl-pi *n char(10) end-pi;  
  
  dcl-s curUser char(10) inz(*user);  
  
  return curUser;  
end-proc;
```

- The PI uses the place-holder *N for the name
- END-PI is specified as a keyword at the end of the DCL-PI statement

Can use named constants for keywords

```
ddl-c SYS_NAME_LEN 10;  
  
ddl-ds sys_obj qualified;  
      obj char(SYS_NAME_LEN);  
      lib char(SYS_NAME_LEN);  
end-ds;
```

Can use named constants for keywords

In fixed form, some keywords allow literals to be specified without quotes: DTAARA, EXTNAME, EXTFLD

What data area is used for fld1?

D fld1 S 10A DTAARA(dta1)

What about fld2?

D dta2 C 'MYLIB/DTAARA2'
D fld2 S 10A DTAARA(dta2)

DTAARA keyword difference

In free-form, an unquoted name is always a variable or named constant

```
D dta1      C      'MYLIB/DTAARA1'
```

```
D fld1a     S      10A  DTAARA(dta1)
dc1-s fld1b char(10) dtaara('DTA1');
```

```
*LIBL/DTA1
```

```
dc1-s fld1c char(10) dtaara(dta1);
```

```
MYLIB/DTAARA1
```

```
D fld2a     S      10A  DTAARA(*VAR:nameFld)
dc1-s fld2b char(10) dtaara(nameFld);
```

```
Value of nameFld
```

Gotchas

- Update does not imply delete
- END-DS, END-PR, END-PI are needed at the end of a subfield or parameter list (even when there are no subfields or parameters)
- Keywords like DTAARA and EXTNAME that assume unquoted names are named constants or variables

(These have already been discussed)

Another gotcha

If you are in the habit of using ellipsis at the end of D and P spec names

```
D customerName...
```

```
D          S          50A
```

That will not work for free-form declarations

```
dc1-s customerName...
      char(50);
```

The name is customerNamechar, and "(50)" is found where the compiler expects to find the data type.

```
dc1-s customerName
      char(50);
```

Colorization in RDI

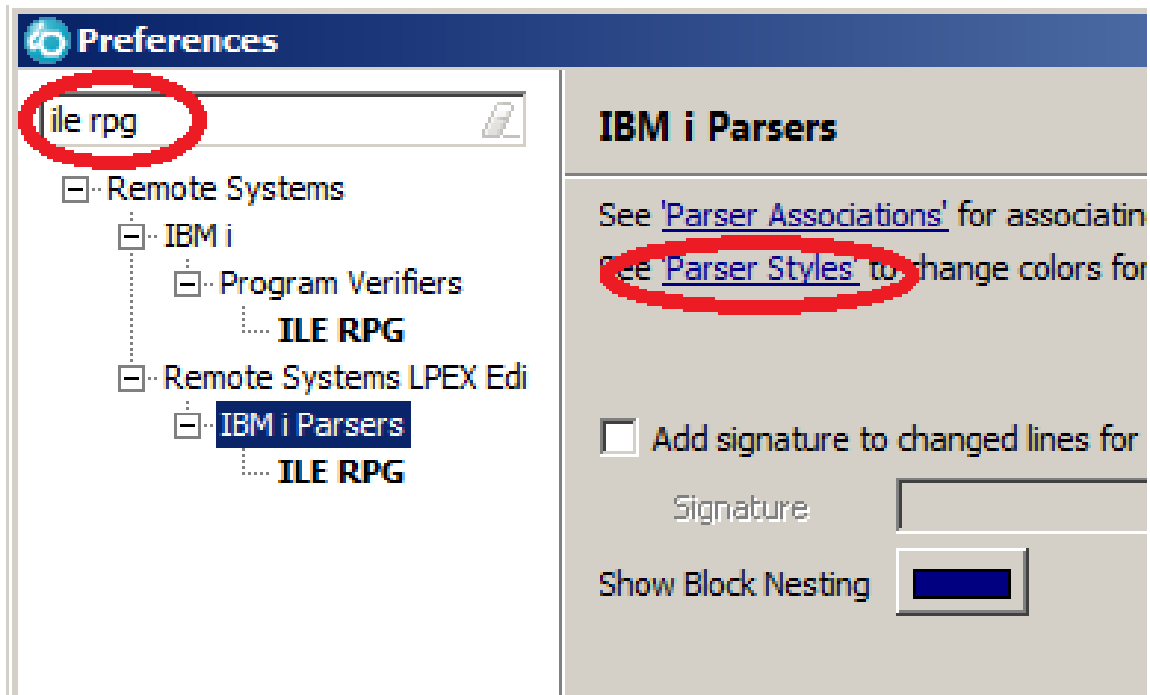
Much more control for colorizing your code

Here is some code with the default colors

```
000101  
000102     dcl-f custfile usage(*update);  
000103  
000104     dcl-ds myDs likerec(custrec : *input);  
000105     /if defined(debug)  
000106         dcl-s debugMsg varchar(100);  
000107     /endif  
000108  
000109     read custfile myDs;  
000110     if myDs.duedate > %date();  
000111         handleOverdue (myDs);  
000112     endif;
```

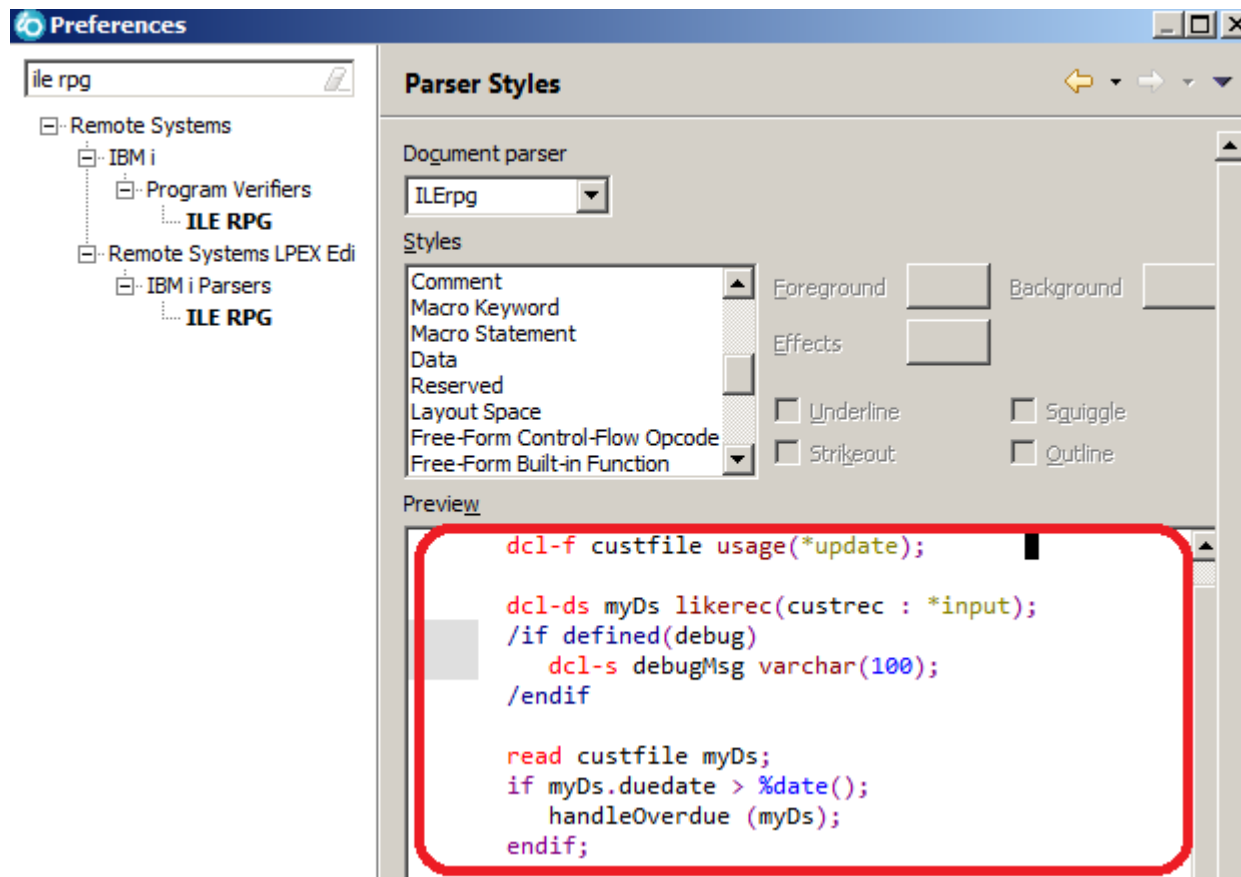
Navigate to the color preferences

- Window > Preferences
- Search for ILE RPG
- Click on Parser Styles



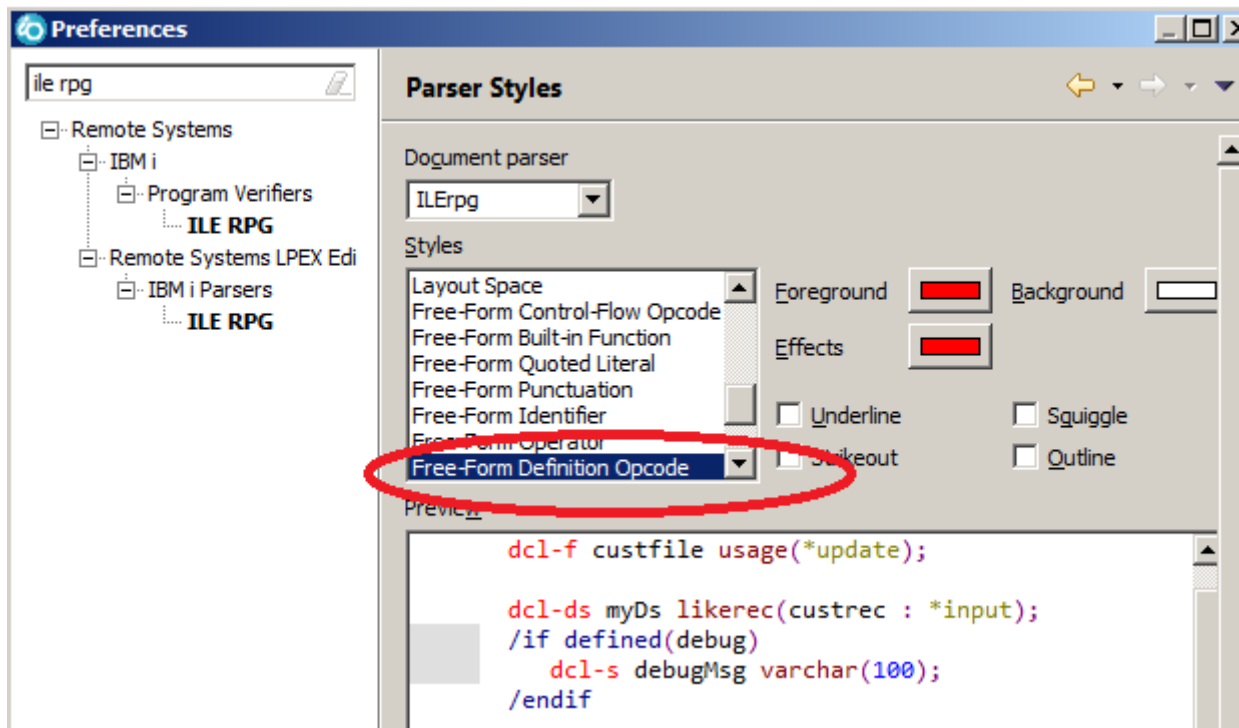
You can change the code to work with

- In the code section, I like to paste in a bit of my own code at the top



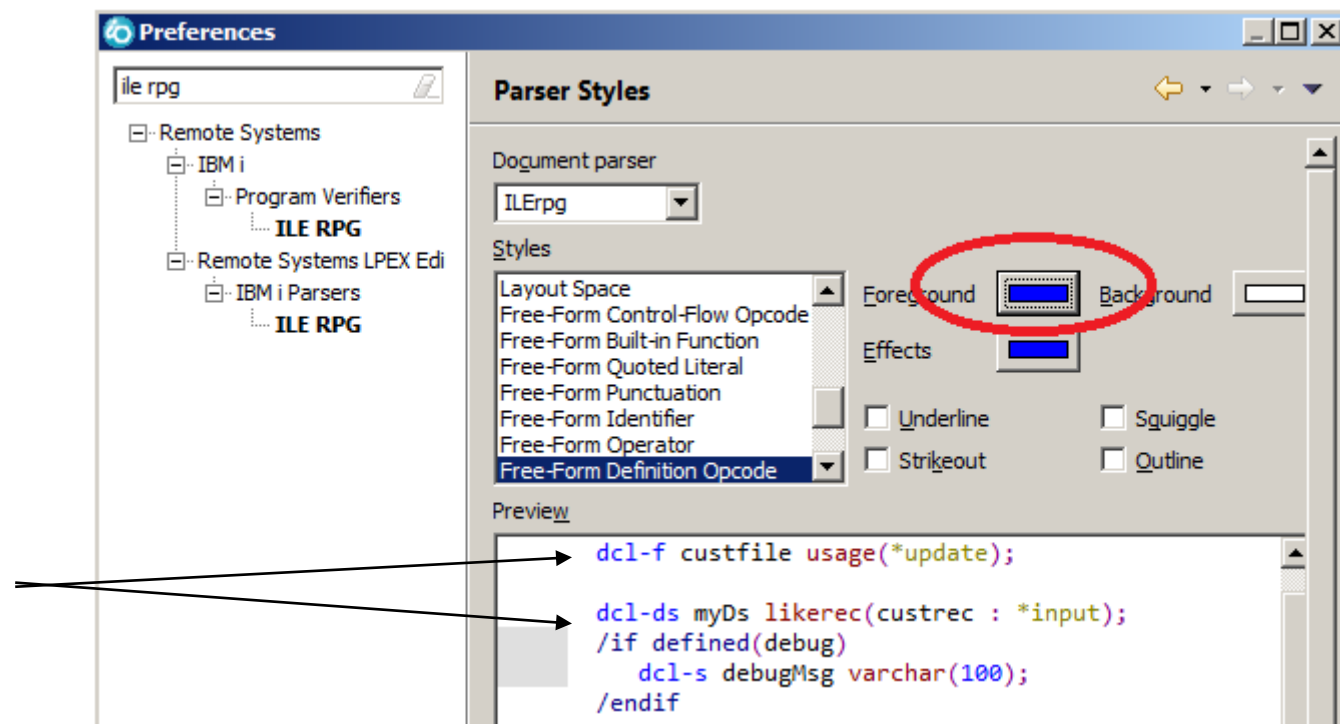
Choose which style you want to change

- Then click on the code you want to change the color for
- The top section will automatically position to the relevant style



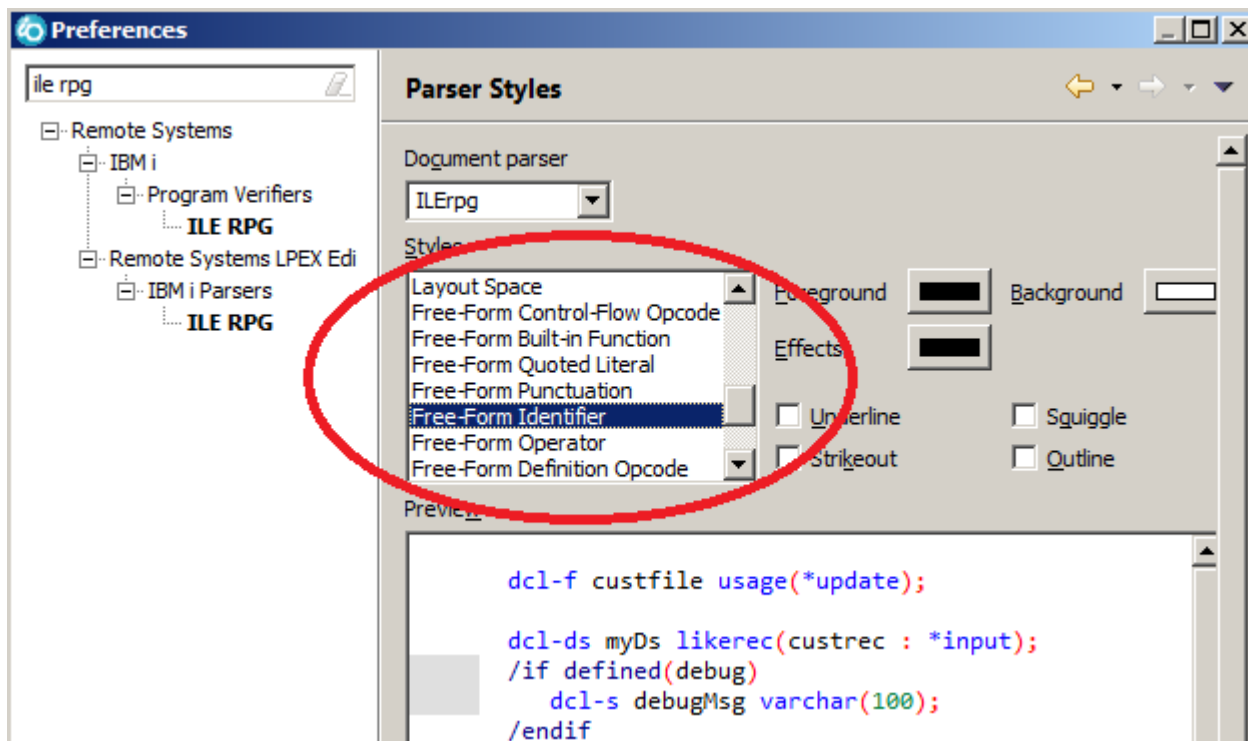
Customize your colors

- Choose the color you want
- It will automatically be colored in the code section so you can see the effect it has



Another way to choose the style

- For most free-form code, the styles are listed together
- You can select them one-by-one, adjusting the colors



Here's how I like it

```
000101
000102     dcl-f custfile usage(*update);
000103
000104     dcl-ds myDs likerec(custrec : *input);
000105     /if defined(debug)
000106         dcl-s debugMsg varchar(100);
000107     /endif
000108
000109     read custfile myDs;
000110     if myDs.duedate > %date();
000111         handleOverdue (myDs);
000112     endif;
```

Summary

We had two goals when designing the new free-form syntax:

- Easy for non-RPG programmers to learn
- Easy for existing RPG programmers to learn

We have a few years of evidence that we have indeed accomplished those goals!



www.ibm.com/software/rational

© Copyright IBM Corporation 2011. All rights reserved. The information contained in these materials is provided for informational purposes only, and is provided AS IS without warranty of any kind, express or implied. IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, these materials. Nothing contained in these materials is intended to, nor shall have the effect of, creating any warranties or representations from IBM or its suppliers or licensors, or altering the terms and conditions of the applicable license agreement governing the use of IBM software. References in these materials to IBM products, programs, or services do not imply that they will be available in all countries in which IBM operates. Product release dates and/or capabilities referenced in these materials may change at any time at IBM's sole discretion based on market opportunities or other factors, and are not intended to be a commitment to future product or feature availability in any way. IBM, the IBM logo, Rational, the Rational logo, Telelogic, the Telelogic logo, and other IBM products and services are trademarks of the International Business Machines Corporation, in the United States, other countries or both. Other company, product, or service names may be trademarks or service marks of others.

Special notices

This document was developed for IBM offerings in the United States as of the date of publication. IBM may not make these offerings available in other countries, and the information is subject to change without notice. Consult your local IBM business contact for information on the IBM offerings available in your area.

Information in this document concerning non-IBM products was obtained from the suppliers of these products or other public sources. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. Send license inquires, in writing, to IBM Director of Licensing, IBM Corporation, New Castle Drive, Armonk, NY 10504-1785 USA.

All statements regarding IBM future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.

The information contained in this document has not been submitted to any formal IBM test and is provided "AS IS" with no warranties or guarantees either expressed or implied.

All examples cited or described in this document are presented as illustrations of the manner in which some IBM products can be used and the results that may be achieved. Actual environmental costs and performance characteristics will vary depending on individual client configurations and conditions.

IBM Global Financing offerings are provided through IBM Credit Corporation in the United States and other IBM subsidiaries and divisions worldwide to qualified commercial and government clients. Rates are based on a client's credit rating, financing terms, offering type, equipment type and options, and may vary by country. Other restrictions may apply. Rates and offerings are subject to change, extension or withdrawal without notice.

IBM is not responsible for printing errors in this document that result in pricing or information inaccuracies.

All prices shown are IBM's United States suggested list prices and are subject to change without notice; reseller prices may vary.

IBM hardware products are manufactured from new parts, or new and serviceable used parts. Regardless, our warranty terms apply.

Any performance data contained in this document was determined in a controlled environment. Actual results may vary significantly and are dependent on many factors including system hardware configuration and software design and configuration. Some measurements quoted in this document may have been made on development-level systems. There is no guarantee these measurements will be the same on generally-available systems. Some measurements quoted in this document may have been estimated through extrapolation. Users of this document should verify the applicable data for their specific environment.

Revised September 26, 2006

Special notices (cont.)

IBM, the IBM logo, ibm.com AIX, AIX (logo), AIX 5L, AIX 6 (logo), AS/400, BladeCenter, Blue Gene, ClusterProven, DB2, ESCON, i5/OS, i5/OS (logo), IBM Business Partner (logo), IntelliStation, LoadLeveler, Lotus, Lotus Notes, Notes, Operating System/400, OS/400, PartnerLink, PartnerWorld, PowerPC, pSeries, Rational, RISC System/6000, RS/6000, THINK, Tivoli, Tivoli (logo), Tivoli Management Environment, WebSphere, xSeries, z/OS, zSeries, Active Memory, Balanced Warehouse, CacheFlow, Cool Blue, IBM Systems Director VMControl, pureScale, TurboCore, Chiphopper, Cloudscape, DB2 Universal Database, DS4000, DS6000, DS8000, EnergyScale, Enterprise Workload Manager, General Parallel File System, , GPFS, HACMP, HACMP/6000, HASM, IBM Systems Director Active Energy Manager, iSeries, Micro-Partitioning, POWER, PowerExecutive, PowerVM, PowerVM (logo), PowerHA, Power Architecture, Power Everywhere, Power Family, POWER Hypervisor, Power Systems, Power Systems (logo), Power Systems Software, Power Systems Software (logo), POWER2, POWER3, POWER4, POWER4+, POWER5, POWER5+, POWER6, POWER6+, POWER7, System i, System p, System p5, System Storage, System z, TME 10, Workload Partitions Manager and X-Architecture are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries.

A full list of U.S. trademarks owned by IBM may be found at: <http://www.ibm.com/legal/copytrade.shtml>.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

AltiVec is a trademark of Freescale Semiconductor, Inc.

AMD Opteron is a trademark of Advanced Micro Devices, Inc.

InfiniBand, InfiniBand Trade Association and the InfiniBand design marks are trademarks and/or service marks of the InfiniBand Trade Association.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

IT Infrastructure Library is a registered trademark of the Central Computer and Telecommunications Agency which is now part of the Office of Government Commerce.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Linear Tape-Open, LTO, the LTO Logo, Ultrium, and the Ultrium logo are trademarks of HP, IBM Corp. and Quantum in the U.S. and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries or both.

Microsoft, Windows and the Windows logo are registered trademarks of Microsoft Corporation in the United States, other countries or both.

NetBench is a registered trademark of Ziff Davis Media in the United States, other countries or both.

SPECint, SPECfp, SPECjbb, SPECweb, SPECjAppServer, SPEC OMP, SPECviewperf, SPECcapc, SPECchpc, SPECjvm, SPECmail, SPECimap and SPECsfs are trademarks of the Standard Performance Evaluation Corp (SPEC).

The Power Architecture and Power.org wordmarks and the Power and Power.org logos and related marks are trademarks and service marks licensed by Power.org.

TPC-C and TPC-H are trademarks of the Transaction Performance Processing Council (TPPC).

UNIX is a registered trademark of The Open Group in the United States, other countries or both.

Other company, product and service names may be trademarks or service marks of others.

Revised December 2, 2010