RPG IV Programming Fundamentals Workshop for IBM i

(Course code AS06)

Student Exercises

ERC 7.0

Trademarks

IBM® is a registered trademark of International Business Machines Corporation.

The following are trademarks of International Business Machines Corporation in the United States, or other countries, or both:

AS/400®	AS/400e™	DB™
DB2®	Integrated Language Environment®	iSeries®
Iterations®	i5/OS™	Language Environment®
OS/400®	Power Systems™	Power Systems Software™
Power®	Rational®	Redbooks®
RPG/400®	System i®	WebSphere®
400®		

Adobe is either a registered trademark or a trademark of Adobe Systems Incorporated in the United States, and/or other countries.

Pentium is a trademark or registered trademark of Intel Corporation or its subsidiaries in the United States and other countries.

Windows is a trademark of Microsoft Corporation in the United States, other countries, or both.

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

Other product and service names might be trademarks of IBM or other companies.

November 2012 edition

Contents

Trademarks

The reader should recognize that the following terms, which appear in the content of this training document, are official trademarks of IBM or other companies:

IBM® is a registered trademark of International Business Machines Corporation.

The following are trademarks of International Business Machines Corporation in the United States, or other countries, or both:

AS/400®	AS/400e™	DB™
DB2®	Integrated Language Environment®	iSeries®
Iterations®	i5/OS™	Language Environment®
OS/400®	Power Systems™	Power Systems Software™
Power®	Rational®	Redbooks®
RPG/400®	System i®	WebSphere®
400®		

Adobe is either a registered trademark or a trademark of Adobe Systems Incorporated in the United States, and/or other countries.

Pentium is a trademark or registered trademark of Intel Corporation or its subsidiaries in the United States and other countries.

Windows is a trademark of Microsoft Corporation in the United States, other countries, or both.

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

Other product and service names might be trademarks of IBM or other companies.

Exercises description

Exercise instructions: The instructions describe what you are to accomplish. There are no definitive details on how to perform the tasks. You are given the opportunity to work through the exercise given what you learned in the unit presentation.

General instructions for all machine exercises

Each exercise begins with a statement of objectives, followed by detailed steps that describe the exercise.

Students should read through the complete exercise before starting the exercise and then read each step completely before performing that step.

When instructions refer to the name of a library, object, or source member, such as **AS06nnn**, replace the lower case **nnn** with the team number assigned to you by the instructor.

Use **AS06nnn** for your user profile name and **AS06** as your initial password. *Your password is set to expire at first signon*. When prompted, change it to something you can remember easily.

You work with your own library, **AS06nnn**.

Complete each exercise step and resolve any problems before you proceed to the next step.

Which editor to use: You can use the PC-based LPEX editor of RD for Power Systems or you can use SEU as your editor. LPEX is the recommended editor. It is straightforward and easy to use.

You can open the LPEX editor by **Start > All Programs > IBM Software Development Platform > IBM Rational Developer for Power Systems Software > IBM Rational Developer for Power Systems Software**.

Your instructor can help you with any questions you have regarding the use of LPEX or SEU.

Exercise 1. Coding and compiling RPG IV

What this exercise is about

In this exercise, you create a new RPGLE source member using the editor of your choice, either the editor in RSE/RD for Power Systems or SEU.

After having created the member, you compile it and, if necessary, correct any compilation errors.

You also have the opportunity to review a compilation listing and correct the cause of any diagnostic messages.

What you should be able to do

At the end of the exercise, you should be able to:

- Implement a simple example of the development process on the i
- Review a compiler listing to determine any compilation errors
- Correct any compilation errors

Introduction

Create a source member named DEVPGM01 in your AS06*nnn*/`QRPGLESRC` source file, where *nnn* is your team number as assigned by the instructor.

DEVPGM01 displays the sum of 2 + 2 in your user message queue.

Compile the RPG IV code and review your compilation listing to correct any problems.

Requirements

- *Student Notebook*
- Userid (**AS06*nnn***) and password **AS06**.

Exercise instructions

You can perform this exercise using either the editor in RSE or the SEU editor. Instructions are written assuming SEU, but if you are familiar with the LPEX editor, use it. Look at Appendix D for a refresher on Rational Developer for Power Systems and its RSE/LPEX user interface.

Part 1: Logging on and finding the source file

__ 1. Sign on to the i as **AS06*nnn***. Replace ***nnn*** with your student number. Your password at the first signon is **AS06**. It is set to expire. Change your password to something you will remember easily.

__ 2. Enter the command **WRKOBJPDM AS06*nnn***. (Replace ***nnn*** with your student number.)

__ 3. Review the list of objects in your library. Find the object QRPGLESRC.

Part 2: Create the source member DEVPGM01

__ 4. Enter option **12** in the column to the left of the QRPGLESRC file to work with it.

__ 5. Create a new member (**F6**). What is the source type that you should use? Use the prompt key if you need some help. _____

__ 6. Enter the code that follows into your source member. Notice that you can use SEU prompting to assist you in order to enter data correctly into the RPG IV specification templates.

__ 7. Here is the code that you enter into your QRPGLESRC file, member DEVPGM01. Note that the source templates are included for your reference only. Enter only the RPG IV code:

```
Do NOT enter the template line immediately below:
DName++++++++++ETDsFrom+++To/L+++IDc.Keywords++++++++Comments
DMessage          s             30    inz('The sum of 2 plus 2 is')
DSums             s              3  0 inz
Do NOT enter the template line immediately below:
 /..1....+....2....+....3....+....4....+....5....+....6....+....
 /free
  sum = 2 + 2;
  message = %trimr(Message) + ' ' + %char(sum);
  Dsply message '*REQUESTER';
  *InLr = *on;
 /end-free
```

__ 8. While you are entering your program, notice function keys **F4** (Prompt) and **F23** (Select Prompt). Try them out if you are not familiar with them.

__ 9. Notice the built-in functions (or BiFs) used in the program. **%trimr** removes rightmost blanks from a character string and **%char** converts other data types to character. We discuss BIFs throughout this class.

Part 3: Compiling your RPG IV source member

__ 10. When you have finished, exit from SEU. You see the **Exit** menu. If you have syntax errors in the code you have just entered, you are prompted to return to editing. If you see that this parameter is set to **Y**, you should return to SEU and correct your syntax problems.

__ 11. When you have closed the source member, you see a list of members in the `QRPGLESRC` file. If you do not see your member that you just created, you might have to refresh the list (**F5**).

__ 12. Compile your member. Option **14** can be used in the column to the left of the member name.

__ 13. When the compilation has completed, view your output using the **`WRKSPLF`** command.

Part 4: Correcting problems

__ 14. Review your compilation listing and determine whether the member was compiled successfully. Was the *PGM object created in your library?

__ 15. Look for diagnostic messages. Do you know what needs to be done to the program to correct any problems?

__ 16. Make any necessary corrections and recompile your program.

__ 17. Repeat this process until the *PGM member is successfully created.

Part 5: Executing the DEVPGM01 *PGM

__ 18. Find the *PGM object in your library.

__ 19. Call your program. You can use the **`CALL DEVPGM01`** command on the command line.

__ 20. If a message was not displayed on your screen, you should see the prompt `MW` at the bottom of your screen. Display your messages and you should see the message from your program.

End of exercise

Exercise 2. Sequencing RPG IV specifications and compiling

What this exercise is about

This exercise provides an opportunity to correctly sequence RPG IV code, to compile your sequenced source member, and to view the compiled listing.

What you should be able to do

At the end of the exercise, you should be able to:

- Find a specific RPG IV source member in your `QRPGLESRC` file
- Edit an RPG IV source member
- Correctly sequence RPG IV code so that it will compile
- Create an executable RPG IV program
- Manipulate the compiler options
- Find and view a compiled source listing

Introduction

You are given a source member VNRADR01 in AS06*nnn*/`QRPGLESRC` where *nnn* is your student number.

VNRADR01 is an inquiry program. It prompts the user for a vendor number. If the number is valid, it displays a vendor's address on the screen. All supporting objects needed for this exercise are provided.

The existing source file is not sequenced correctly. Put the specifications in the correct order and then compile the corrected source member.

When it is successfully compiled, call and test the program.

Exercise instructions

In the following instructions, we direct you on what to do without telling you explicitly how to do it. If you need any assistance, please ask your instructor.

Part 1: Log on

__ 1. Sign on to an i 5250 session.

Part 2: Edit the source member

If you are using the LPEX editor and filters, expand your `QRPGLESRC` filter. Sign on when or if prompted. Open the existing member in your `QRPGLESRC` file, **VNRADR01**, by double-clicking it.

__ 2. If you are using SEU, work with file `QRPGLESRC`. Find the source member **VNRADR01**. What is the source type? _____

__ 3. Edit the member **VNRAR01**.

Part 3: Sequence the RPG IV specifications

__ 4. Review the code in the member. The code is *incorrectly* sequenced and a compile would end in error. A copy of this source code is provided for you as follows.

__ 5. Correctly sequence the source code using SEU move commands.

__ 6. `M` moves a line of code. `B`efore or `A`fter designates the target location. `MM` is used for moving blocks of code. Type `MM` on the first and last block lines then place an `A` or `B` on the target line. The move is executed when the **Enter** key is pressed.

- RPG specifications are ordered in header (known as control) specification, file definition, data definition, and calculation specifications sequence.
- Like specifications are grouped together.
- When you are satisfied with the sequence, **exit** SEU and **save** the member.


```
 ****************************************************************
 *                       VNRADR01                               *
 *                  Vendor Address Inquiry                      *
 * This program prompts the user for a vendor number and displays *
 * the vendor's address information on the screen.              *
 *                                                              *
 * The user has options to exit the program and to print a vendor *
 * record.                                                      *
 *                                                              *
 * INDICATORS:                                                  *
 *   Exit      - the user requests to exit the program          *
 *   PrintIt   - the user requests to print vendor address      *
 *   NotFound  -  no vendor found to match the input vendor number *
 ****************************************************************
 ****************************************************************
D ToDaysDate      S               D
  // Named indicators used with display file
D WkStnInd        DS
D  Exit                    3      3N
D  NotFound               99     99N
D  PrintIt                10     10N
H DftActGrp(*yes) ExprOpts(*ResDecPos) DatFmt(*USA)
 ****************************************************************
 /Free

  TodaysDate = %date(*date); // Get date from system
  ExFmt     Prompt_Fmt; // Prompt for vendor number

  DoW Not Exit; // Do the following until user presses F3 (Exit)

      NotFound= *off; // Initialize the record found indicator
      Chain VndNbr Vendor_PF; // Find the vendor record

      If %Found(Vendor_PF);  // If we find a valid vendor record:
         ExFmt Dsply_Fmt;    // Disply the vendor record

         If PrintIt;         // If the user pressed F10,
            Write Vnadd_Fmt; // print the vendor record
         EndIf;

      Else;               // We did not find a record
         NotFound = *on;  // Set the record found indicator on
      EndIf;

            ExFmt Prompt_Fmt; // Prompt for a new vendor number
  EndDo;

  *inLr = *on;  // End the program
 /End-free
 ****************************************************************
  // Vendor Display Formats
FVndAdr01  CF   E             WorkStn InDDS(WkStnInd)
  // Vendor Data File
FVendor_PF IF   E           K Disk
  // Report Formats
FVnpAdr    O    E             Printer OflInd(PrtOver)
```

Part 4: Compile the source member

__ 7. Compile the member **VNRADR01**.

__ 8. If you are using 5250 emulation, you can use the `WRKSBMJOB` command to find the compilation job you started.

What command is being executed in the job?

_____

__ 9. When the compile has ended, continue with the next step.

Part 5: Find and view the compiler listing

__ 10. If you are not actively running a 5250 session, start one.

__ 11. Use work with spooled files (`WRKSPLF`) to find the compiler listing.

__ 12. Display the file named **VNRADR01**. The listing file name is always the same as the compiled member name.

Identify the field names copied in from the externally described files. They have specification types of I and O. These specifications are generated automatically for you by the compiler.

Input (I) specifications define the format of incoming data from files (physical files, logical files, display files for example).

Output (O) specifications define the format in which data will be written to a physical or logical file in addition to a display or printer file.

__ 13. Notice the **source specifications** section.

__ 14. Notice the **message summary** section.

__ 15. Locate the **cross reference** section. Which named indicators are used?

Hint

Look for the data type (**N**) fields.

Can you guess which field definitions are indicators?

_____

__ 16. Check the last message in the listing. Was the program object (***PGM**) placed in your library? (Check your listing) _____.

If the program was not created try to find the error and then compile again. If you need help, ask the instructor.

Part 6: Run your new program

__ 17. Use the `WRKOBJPDM` command to look at the objects in your AS06*nnn* library.

__ 18. Look for a new object, named **VNRADR01**. What is the object type?

_____

__ 19. To run the new program, enter `CALL VNRADR01` on the command line. You should see the vendor address inquiry screen. **Enter** a few valid vendor numbers (10001 through 10050) to test the application.

End of exercise

Optional Exercise

Step 1. Compiler options

__ 1. Prompt the **PDM** compile option before pressing **Enter**. Or, go back to LPEX and take the compile option and prompt it by pressing the **options** button.

__ 2. Specify ***SECLVL** and ***NOEXPDDS** for the compiler options parameter.

__ 3. Press the **Enter** key.

Step 2. Review the changes

__ 1. Display the compiled listing again.

__ 2. Are external field names listed? _____

__ 3. Look at the **message summary** section. The text describes the messages in more detail.

End of exercise

Exercise 3. Coding a report program

What this exercise is about

This exercise provides an opportunity to practice entering and compiling an RPG IV program.

What you should be able to do

At the end of the exercise, you should be able to:

- Code an RPG IV program using the SEU or LPEX editor
- Compile the RPG IV program
- Review the compiler listing and correct errors
- Run the program successfully

Introduction

Produce a listing of the records in the `ITEM_PF` database file.

Sample report output follows:

```
PAGE    1                COST OF ON HAND INVENTORY

ITEM#  DESCRIPTION                    COST      QTY OH     COST OH

20001  Telephone, one line          15.00          10
20002  Telephone, two line          89.00           5
20003  Speaker Telephone            85.00           6
20004  Telephone Extension Cord      1.10          25
20005  Dry Erase Marker Packs        2.25         428
20006  Executive Chairs            325.00          10
20007  Secretarial Chairs           55.00          13
20008  Desk Calendar Pads            5.00          56
20009  Diskette Mailers               .20         128
20010  Address Books                 6.00       1,680
20011  Desk lamp, brass             20.00           3
20012  Blue pens                    20.00           7
20013  Red pens                    100.00          14
20014  Black pens                  150.00          25
                                                                More...
```

Files used

All files are externally described:

- Physical file `ITEM_PF` (input) to be processed by key
- Printer file `ITPCOST`

Exercise instructions

Part 1: Create the PRTF ITPCOST

__ 1. Study the **DDS** source (also in your QDDSSRC file) for the printer file ITPCOST. Note the different format names:

```
*****************************************************************
*                          ITPCOST                              *
*                Cost of Inventory On Hand Report               *
*****************************************************************
* This printer file is used to print the cost of on hand        *
* inventory.  This report is double spaced.                     *
*                                                               *
* FORMATS:                                                      *
*   HEADING:  To be printed at the top of each page.            *
*   DETAIL:   To print the information for one item.            *
*   TOTAL:    To print the total value of all on hand items     *
*             at the end of the report.                         *
*                                                               *
*****************************************************************
A                                      REF(DICTIONARY)
A          R HEADING                   SKIPB(2) SPACEA(2)
A                                       1'PAGE'
A                                      +1PAGNBR EDTCDE(Z)
A                                      26'COST OF ON HAND INVENTORY'
A                                       1'ITEM#' SPACEB(2)
A                                      +2'DESCRIPTION'
A                                     +19'COST'
A                                      +6'QTY OH'
A                                      +5'COST OH'
A          R DETAIL                      SPACEA(2)
A            ITMNBR    R                1
A            ITMDESCR  R               +2
A            ITMCOST   R               +2EDTCDE(K)
A            ITMQTYOH  R               +2EDTCDE(1)
A            ITMCOSTOH R   +2          +2REFFLD(ITMCOST) EDTCDE(K)
A          R TOTAL                       SPACEB(1)
A                                   1'TOTAL COST OF INVENTORY ON HAND:'
A            TOTCOSTOH R   +4           53REFFLD(ITMCOST) EDTCDE(K)
```

__ 2. You use only the **HEADING** and **DETAIL** formats in this exercise.

__ 3. Create (**compile**) the printer file object from the DDS source provided.

Part 2: Write your RPG IV program ITRCOST

Use your lecture notebook to help you code this program. The sample program that we discussed in lecture helps to guide you through the process. The logic you write for coding one report applies to other report programs. What differs is how you manipulate the data. In this exercise, you do not modify the data in any way. You do that in a later exercise.

__ 4. Write an RPG IV program to list all the records in `ITEM_PF` sequentially using the **record** key. Use the sample report provided at the beginning of the exercise. Notice that we are going to print only some of the fields described in the PRTF. We add more fields in subsequent exercises and use the same PRTF again. Add a new member, **`ITRCOST`**, to your `QRPGLESRC` in your AS06*nnn* library.

__ 5. You noted in the previous step that the PRTF contains three separate formats. How many formats do you use in your program? _____

__ 6. Your program should reference and use only those formats that are required to produce a report like the sample report.

You might **`INCLUDE`** those formats to be used or **`IGNORE`** those formats to be dropped on the file specification.

__ 7. Enable page overflow in your program using the **`OflInd(PrtOver)`** keyword for your `ITPCOST` printer file. List the detail of each item record that is read in the report.

__ 8. Compile and test the program, checking the printed results against the sample provided.

Note

Notice that when you display spooled reports in your OUTQ, you do not see all the spacing and skipping that you specify in your printer file DDS. Spooled data is compressed to reduce the storage required.

End of exercise

Exercise 4. Adding overflow

What this exercise is about

In this exercise, you enhance the report program that you wrote in the previous exercise.

What you should be able to do

At the end of the exercise, you should be able to:

- Modify an RPG IV report program
- Compile your RPG IV program
- Review the compiler listing and correct errors
- Run your program successfully

Introduction

Enhance the program that produced the listing of the records in the `ITEM_PF` database file. You improve your listing program.

Sample report output follows:

```
PAGE    1                COST OF ON HAND INVENTORY

ITEM#  DESCRIPTION                    COST      QTY OH     COST OH

20001  Telephone, one line          15.00          10
20002  Telephone, two line          89.00           5
20003  Speaker Telephone            85.00           6
20004  Telephone Extension Cord      1.10          25
20005  Dry Erase Marker Packs        2.25         428
20006  Executive Chairs            325.00          10
20007  Secretarial Chairs           55.00          13
20008  Desk Calendar Pads            5.00          56
20009  Diskette Mailers               .20         128
20010  Address Books                 6.00       1,680
20011  Desk lamp, brass             20.00           3
20012  Blue pens                    20.00           7
20013  Red pens                    100.00          14
20014  Black pens                  150.00          25
20015  Number 2 pencils              2.50         150
20016  Number 3 pencils              4.50          25
20017  Two Drawer File Cabinets      5.50          15
20018  Manilla folders               4.00          50
20019  Hanging file folders          3.00         150
20020  Metal desk                  125.00           2
20021  Pink erasers                   .25         250
20022  Gum erasers                    .75         100
20023  Twelve inch rulers            1.25          15
20024  Eighteen inch rulers          1.00          65
20025  Staples                       2.50          14
20026  Three Ring Binders            4.00          10
20027  Three hole punch              8.00         149

PAGE    2                COST OF ON HAND INVENTORY

ITEM#  DESCRIPTION                    COST      QTY OH     COST OH

20028  Desk picture frame 5 X 7      3.50          47
20029  8 1/2 x 11 lined paper        1.00       1,500
20030  Headset                       1.50         523
20031  Folding chair                  .50         127
20032  Digital desk clock           19.99          44
20033  Banquet table, 6 ft.          2.00      10,128
20034  Executive Oak Desk          500.00           1
: : : :
```

Files used

All files are externally described:

- Physical file `ITEM_PF` (input) to be processed by key
- Printer file `ITPCOST`

Exercise instructions

Part 1: Modify your program ITRCOST

__ 1. Make a copy of your program, **ITRCOST**. In LPEX, you can edit your original program **ITRCOST**, and then use **Save as** to create your new member, ITRCOSTE.

If you are using PDM/SEU, use option **3** to copy the **ITRCOST** source member to a new member named ITRCOSTE.

Part 2: Enhance your ITRCOSTE program

__ 2. Enhance your program to print headings on the page overflow condition using the **HEADING** format of the ITPCOST PRTF.

__ 3. Test for overflow in your program. Overflow should be checked before you print a detail format.

__ 4. Compile and test the program, checking the printed results against the sample provided.

End of exercise

Exercise 5. Data definition

What this exercise is about

This exercise provides an opportunity to code RPG IV file specifications (F-specs) for externally defined files as well as data definition specifications (D-specs) to define work fields used in the program.

What you should be able to do

At the end of the exercise, you should be able to:

- Code F-specifications
- Code D-specifications

Introduction

Using either the LPEX or the SEU editor, modify an existing member, named **PORLIST** adding appropriate F-specs and D-specs given the following specifications.

Exercise instructions

Part 1: Examine the existing RPG IV source

__ 1. Locate and edit your copy of **PORLIST** in your QRPGLESRC file. A copy follows:

```
/Free
   // Headings on first page
   Write Heading;

   // Read first record of file
   Read Item_PF;

   DoW not %eof(Item_PF);

        // Page overflow?
        If PrtOver;
             Write Heading;
             PrtOver = *Off;
        EndIf;

        // Accumulate totals
        TotQtyAvl = ItmQtyOH + ItmQtyOO;
        // Set indicator for Low Available Quantity
        *in43 = (TotQtyAvl < 15);

        // Accumulate low quantity situations
        If *in43;
             Low = Low + 1;
        EndIf;

        // Calculate value (at cost) of available inventory
        AvlCost = ItmCost * TotQtyAvl;

        // Calulate value (at retail) of available inventory
        AvlPrice = ItmPrice * TotQtyAvl;

        // Accumulate the total number of records processed
        Count = Count + 1;

        // Print the detail format
        Write Detail;
        // Read second and subsequent records of file
        Read Item_PF;
   EndDo;

   // End of file processing

   // Move the program accumulators to the printer file fields
   LowCountP = Low;
   TotCountP = Count;
   // Print the record format for totals
   Write Footing;
   *InLr = *on;
/End-Free
```

__ 2. Notice that there are no files defined yet. Defining the files is part of the exercise.

__ 3. Review the following sample report:

```
6/24/03                 Item Master Listing                 Page:    1

Item No Description               Qty On Hand  Qty On Order  Tot Avail   Avail Cost  Avail List

20001   Telephone, one line                10             6         16       240.00    480.00
20002   Telephone, two line                 5            12         17     1,513.00  2,125.00
20003   Speaker Telephone                   6             8         14 Low  1,190.00  1,960.00
20004   Telephone Extension Cord           25            10         35        38.50    104.65
20005   Dry Erase Marker Packs             428           10        438       985.50  1,314.00
20006   Executive Chairs                   10             2         12 Low  3,900.00  7,200.00
20007   Secretarial Chairs                 13             5         18       990.00  1,890.00
20008   Desk Calendar Pads                 56            10         66       330.00    528.00
20009   Diskette Mailers                   128           200        328        65.60    278.80
  :       :::::::
  :       :::::::
20047   Yellow paper, 8 1/2 X 11           199           10         209       616.50    833.91
20048   3 hole white paper                 35            10         45        150.70    314.55
20049   3 hole lined paper                 10             4         14 Low     53.90     97.86
20050   Heavy duty stapler                  5             0          5 Low     45.70    106.25
                  Low Stock Count            7
                  Total Stock Count         50
                       *** End of Listing ***
```

__ 4. There are some fields used that you will define in your program to produce low stock count and total stock count.

Part 2: Add file definitions to PORLIST

__ 5. All files are externally described.

__ 6. Physical file, **ITEM_PF**, is to be processed sequentially by key as input only. It is a procedural file. If you want to view the DDS for **ITEM_PF**, a copy can be found in your QDDSSRC file. **ITEM_PF** references field reference file, **DICTIONARY**, which contains specific field information. A copy is also in your **QDDSSRC** file. All database file descriptions can also be found in the appendix.

__ 7. Printer file, **POPLIST** is an output file that is externally described. You have a copy in your **QDDSSRC** file. Assign an indicator, PrtOver, as you have done earlier, for page overflow.

Note

You could name the overflow indicator anything you wanted. Also, notice that the definition of the indicator is accomplished simply by referencing it on the printer file specification. It is not necessary to define the indicator explicitly on a data definition specification. However, many i installations make a practice of explicitly defining all indicators in order to make maintenance more straightforward.

Part 3: Review total field definitions in POPLIST

__ 8. Open your member, POPLIST. Identify the **FOOTING** format where the accumulators are defined; note the names of these accumulators. You need to identify where these fields are calculated in the program, in order to determine any other work variables which might need to be declared.

Hint

You could try compiling the PORLIST program as follows. The compiler listing will provide useful information in the cross reference section, detailing where every variable is defined and used.

Part 4: Compile and test your program PORLIST

__ 9. Compile the printer file used by PORLIST, POPLIST in your QDDSSRC source file.

__ 10. Compile your program PORLIST.

__ 11. Review your compiler listing. Notice the external files, ITEM_PF and POPLIST that have been included in your compilation.

__ 12. Correct any errors based on the compilation listing.

__ 13. When your program has compiled, run the command, CALL PORLIST, to run your program.

__ 14. Check your output queue and see whether the report was produced. Check the sample output above. If the report was produced, you have finished the exercise.

End of exercise

Exercise 6. Adding arithmetic function

What this exercise is about

This exercise provides an opportunity to enhance your ITRCOSTE program.

What you should be able to do

At the end of the exercise, you should be able to:

- Code arithmetic calculations to produce an extension and totalling of data
- Run the program

Introduction

In this exercise, you use the `ITPCOST` printer file and the ITRCOSTE RPG IV program that you worked with in an earlier exercise.

You must have completed the ITRCOSTE program in the earlier exercise to perform this exercise.

This exercise assumes that the student completed the earlier *Coding a Report Program* exercise successfully. If you were unable to complete that exercise, you should ask your instructor for assistance.

Introduction

Enhance the audit listing of the records in the `ITEM_PF` database file that you wrote earlier.

Sample report output follows:

```
PAGE    1                COST OF ON HAND INVENTORY

ITEM#  DESCRIPTION                    COST      QTY OH     COST OH

20001  Telephone, one line          15.00          10     150.00
20002  Telephone, two line          89.00           5     445.00
20003  Speaker Telephone            85.00           6     510.00
20004  Telephone Extension Cord      1.10          25      27.50
20005  Dry Erase Marker Packs        2.25         428     963.00
20006  Executive Chairs            325.00          10   3,250.00
20007  Secretarial Chairs           55.00          13     715.00
20008  Desk Calendar Pads            5.00          56     280.00
20009  Diskette Mailers               .20         128      25.60
20010  Address Books                 6.00       1,680  10,080.00
20011  Desk lamp, brass             20.00           3      60.00
20012  Blue pens                    20.00           7     140.00
20013  Red pens                    100.00          14   1,400.00
20014  Black pens                  150.00          25   3,750.00
20015  Number 2 pencils              2.50         150     375.00
20016  Number 3 pencils              4.50          25     112.50
20017  Two Drawer File Cabinets      5.50          15      82.50
20018  Manilla folders               4.00          50     200.00
20019  Hanging file folders          3.00         150     450.00
20020  Metal desk                  125.00           2     250.00
20021  Pink erasers                   .25         250      62.50
20022  Gum erasers                    .75         100      75.00
20023  Twelve inch rulers            1.25          15      18.75
20024  Eighteen inch rulers          1.00          65      65.00
20025  Staples                       2.50          14      35.00
20026  Three Ring Binders            4.00          10      40.00
20027  Three hole punch              8.00         149   1,192.00

PAGE    2                COST OF ON HAND INVENTORY

ITEM#  DESCRIPTION                    COST      QTY OH     COST OH

20028  Desk picture frame 5 X 7      3.50          47     164.50
20029  8 1/2 x 11 lined paper        1.00       1,500   1,500.00
20030  Headset                       1.50         523     784.50
20031  Folding chair                  .50         127      63.50
20032  Digital desk clock           19.99          44     879.56
20033  Banquet table, 6 ft.          2.00      10,128  20,256.00
20034  Executive Oak Desk          500.00           1     500.00
20035  Flourescent lamp             12.00          12     144.00
20036  India ink                     2.05          96     196.80
20037  Pencil sharpener, manual      5.55          13      72.15
20038  Mouse pad                      .37         208      76.96
20039  2 ring binders                2.05          12      24.60
20040  Ergonomic mouse               4.40          12      52.80
20041  Printer ribbon                2.95           6      17.70
20042  White out                      .98           1        .98
20043  Office tool kit              24.00          12     288.00
20044  Pink paper, 8 1/2 X 11        2.95         166     489.70
20045  Blue paper, 8 1/2 X 11        2.95          75     221.25
```

```
20046  Continuous 8,5 X 11 paper   20.00          24     480.00
20047  Yellow paper, 8 1/2 X 11     2.95         199     587.05
20048  3 hole white paper           3.35          35     117.25
20049  3 hole lined paper           3.85          10      38.50
20050  Heavy duty stapler           9.15           5      45.75

TOTAL COST OF INVENTORY ON HAND:                        51,755.90
```

Files used

All files are externally described:

- Physical file **ITEM_PF** (input) to be processed by key
- Printer file **ITPCOST** (no changes are required to the copy that you used in your previous exercise)

Exercise instructions

Part 1: Cost of inventory on hand report

__ 1. Study the **Cost of inventory on hand** report.

Part 2: Copy your source program ITRCOSTE

__ 2. Make a copy of your solution, **ITRCOSTE**, from the previous exercise and name it **ITRCOSTADD**.

__ 3. Modify your new program, **ITRCOSTADD**, to:

i. Produce the cost on hand field as shown in the sample report

ii. Produce the total line that you see in the sample output

Hint

It is always a good idea to add new functions individually. This can make debugging problems much easier than adding many new things at once. Code the first enhancement above. Then, **compile and test** your program. When you are satisfied with the results, code the **logic** to produce the **total line**. Remember to write the total line following the format of the `ITPCOST` printer file.

Part 3: Compile and test your program

__ 4. Repeat this process until your program produces the report correctly.

End of exercise

Exercise 7. Data manipulation

What this exercise is about

This exercise is an opportunity for you to use expressions and built-in functions in your RPG IV programs.

What you should be able to do

At the end of the exercise, you should be able to code:

- Logical expressions
- Character expressions
- Arithmetic expressions

Introduction

You are given report layout information and a description of the function of the program that you write. A PRTF (`VNPADR03`) is provided that you use to produce output from your RPG program to produce the report.

You will write a program, `VNRADR03`, to produce the report.

Purpose

Produce a listing of active vendors from the `VENDOR_PF` file.

General considerations

Sample report output is provided. Also, several demonstration programs are provided to show you the results of using several BIFs that you might elect to use in your program. The first steps of the exercise encourage you to run and browse these programs.

Files used

All files are externally described:

- `VENDOR_PF` (input) to be processed sequentially by key
- Printer file `VNPADR03` PRTF

Processing

Review the DDS source for the printer file and the PF in `QDDSSRC` in your student library. Make a note of the format names that you need to use in this program.

Exercise instructions

Part 1: Run the %SCAN and %SUBST demonstration programs

Experiment by running the sample programs, BFRSCAN and BFRSUBST. This program is provided to familiarize you with these built-in functions which you likely use as you code the program in this exercise.

__ 1. Call the **BFRSCAN** program and follow the instructions to see the results of using **%SCAN**. **%SCAN** searches a string using a search argument and returns a numeric value The program asks you for your first and last name. Try the program different ways. Enter your **first name** starting in the first position followed by a blank and then your **last name**. Next, try it with a blank in front of your first name. Can you see how %SCAN works? Now just fill all **20** characters with any **non blank** characters. What value does %SCAN return now?

__ 2. Call the **BFRSUBST** program. Enter any **string** of three words. You can try great is **RPGIV** to begin if you want. Rearrange and truncate the words in your input phrase by specifying **start** and **length** values for the three **%SUBST** expressions.

__ 3. To do this, **type** three words of your choice into the input phrase area so that no word is in the same relative position as you want in the output phrase. Next, using the **three substring functions**, pick up the three words in the sequence you want them to appear in the output phrase. With a little practice, you should be able to easily specify offsets and lengths to help the program rearrange the words in your phrase.

__ 4. If you have a question, ask your instructor.

Part 2: Review the sample report

__ 5. Examine the output that follows:

```
Page     1                              Active Vendors                                                  01/29/2002  13:45:56

                                                                                                        Vend  Vendor
 Vend                                                                                  VND      Zip  Area       Tel  Sales
  Num  Vendor Name                Vendor Street              Vendor City               State    Code  Code       No  Person

10001  John M.  Smith & Sons      2732 Fourth Ave.           New York                   NY     10001   201  662-1000  Bill
10002  Hugh Wilson & Son          4583 Landmark Towers       Summit                     CO     32110   303  454-1010  Jerry
10003  Paper and Pencil           280 Park Ave.              New York                   NY     10017   212  658-5346  Ross
10004  The Desk Shop              461 Congress Ave.          San Francisco              CA     91462   415  329-3435  Joyce
10006  Andy Glover Corp.          100 Plainfield Rd.         Montgomery                 AL     40612   205  600-7000  Carl
10007  The Chair Market           11 Rocking Horse Ct.       Hardford                   CT     02185   203  458-3636  John
10008  Office Joe                 5300 Bradfield Blvd.       Horsham                    NY     78231   501  742-6565  Howard
10009  Reams of Paper             3 Manilla Ct.              Oklahoma City              OK     72143   405  616-2525  Cindy
10010  Clifford Distribution      3 Mountain View Rd.        Las Vegas                  NV     63813   702  722-4585  Roy
 ....        ...                        ..                      .
 ....        ...                        ..                      .
 ....        ...                        ..                      .
 ....        ...                        ..                      .
10049  Susquehanna Supply         1122 Front St.             Chenango Valley            PA     74113   406  258-1257  Hector
10050  Genessee Office Products   1313 Koday Rd.             Binghamton                 NY     34506   816  555-1313  Antonio


       Number of active vendors:          46 *
```

__ 6. You should notice several things about this report. First, only the first name is shown for the vendor sales person contact at the vendor's office. Second, since we are concerned only about active vendors, we are showing a total count of active

vendors. Both active as well as inactive vendor records can be found in the **Vendor_PF** file.

Part 3: Review and compile PRTF VNPADR03

__ 7. Review the source file, **VNPADR03**:

```
     *****************************************************************
     *                          VNPADR03                             *
     *                    VENDOR ADDRESS REPORT                      *
     *****************************************************************
     * THIS PRINTER FILE FORMATS THE VENDOR ADDRESS DATA.            *
     * INDICATORS:  NONE                                             *
     *****************************************************************
     *                       CHANGE LOG                              *
     * DATE        PROGRAMMER      DESCRIPTION                       *
     * ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ *
     * 8/27/96     RJSLANEY        NEW FILE.                         *
     *****************************************************************
A                                     REF(VENDOR_PF)
A          R HEADING
A                                     SKIPB(001)
A                                     TEXT('VENDOR_LISTING')
A                                    1'PAGE'
A                                   +1PAGNBR EDTCDE(Z)
A                                   45'Active Vendors'
A                                  110DATE EDTCDE(Y)
A                                  122TIME
A
A                                  110'Vend' SPACEB(1)
A                                   +2'Vendor'
A
A                                    1'Vend' SPACEB(1)
A                                   87'VND'
A                                   +6'Zip  Area'
A                                   +6'Tel  Sales'
A
A                                    1'Num'  SPACEB(1)
A                                    8'Vendor Name'
A                                   35'Vendor Street'
A                                   62'Vendor City'
A                                   87'State'
A                                   +3'Code  Code'
A                                   +6'No  Person' SPACEA(2)
A
A          R DETAIL                   SPACEB(1)
A            VNDNBR    R             1EDTCDE(Z)
A            VNDNAME   R             8
A            VNDSTREET R            35
A            VNDCITY   R            62
A            VNDSTATE  R            87
A            VNDZIPCODER            +7
A            VNDAREACD R            +2
A            VNDTELNO  R            +1EDTWRD('0   -    ')
A            VNDSALES1     16       +1
A
A          R TOTAL                    SPACEB(1)
A                                     7'Number of active vendors:'
A            COUNT          9  0     35EDTCDE(1)
A                                   50'*'
```

__ 8. Compile your source member `VNPADR03` in your source file `QDDSSRC`. This creates your **PRTF** that is used by the RPG program to produce the report. In a subsequent exercise, code your own PRTF.

Part 4: Write the program

Here is an explanation of the steps to code your program, `VNRADR03`.

__ 9. Create a new source member, `VNRADR03`.

__ 10. Print the **headings** of the report for the first page.

__ 11. Read the first record of the `VENDOR_PF` file.

__ 12. You should process records until **End of File** is encountered. Code an **DoW** group as you have done before to handle this.

__ 13. For each record read:

- Check to see if the record is active (`VNDACTIVE = 'A'`). If the record is active:
  - Extract the first name from the sales person name field (**VndSales**) and place it in a shorter field of **16** characters named **VndSales1**. This field is shorter than the name field and all we want to print is the sales person's first name (the **VNDSALES** field contains first and last name separated by a blank). The shorter field will be placed in **VNDSALES1** by your program. Use BiFs where possible.

    For example, if the **VndSales** field contains the value, 'Jim Slonovski ' the field **VndSales1** should contain 'Jim '.
  - Print the **detail line format** of the PRTF.
  - Increment the record counter (see the definition of **COUNT** in the PRTF). You will print it at the bottom of the report as a total number of active records. You will **not** need to define this field in your program.
- You are responsible for printing headings for both the first page and whenever an overflow condition is encountered.
- When **EOF** for the **VENDOR_PF** is reached, you should print the **TOTAL** format of the PRTF and end the program.

Part 5: Test and execute your program

__ 14. Your program should produce output similar to the report that follows.

__ 15. To view the data to validate your report, you may use **SQL** if you know it, or enter this command:

```
RUNQRY *n Vendor_PF
```

You can start another 5250 session to view the data and your report at the same time.

__ 16. In this program, the field names used in the **PRTF** matched those in the `VENDOR_PF` file. Unfortunately, this is not always the case. In many applications, the **PRTF** fields are defined using different names than those in the associated data file. How would you code your program to ensure that the data from the PF appears in the PRTF?

_____

_____

End of exercise

Exercise 8. Printing from an RPG IV program

What this exercise is about

This exercise provides an opportunity to create a printer file with DDS and to use the file with an RPG IV program.

What you should be able to do

At the end of the exercise, you should be able to:

- Create a printer file with DDS incorporating:
  - Database fields
  - Programmer described fields
  - `EDTCDE, EDTWRD, SKIP, SPACE, PAGE, DATE, TIME` keywords
- Write RPG IV code to use externally described printer files and handle page overflow

Introduction

In this exercise, you again work with the vendor master file. Rather than use the `Vendor_PF`, we use the logical file, `VndNam_LF`, which uses the vendor name as its key.

The report list vendors, their YTD purchases and the outstanding balance due. Your program also calculates a total count of active vendors, a grand total of YTD purchases, and a grand total of the total balance due.

Purchasing has designed the report layout and has provided a copy of it for your use.

Exercise instructions

Part 1: Review the report layout and file

__ 1. Following is the report layout designed by purchasing:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9....+..
PAGE 6666                        Balance Due Vendor Report                              mm/dd/yy
    Vendor Name    and     Number  Vendor City           State   YTD Purchases  Current Bal. Due

OOOOOOOOOOOOOOOOOOOOOOOO  66666  OOOOOOOOOOOOOOOOOOOOO  OO  666,666,666.66-    6,666,666.66-
OOOOOOOOOOOOOOOOOOOOOOOO  66666  OOOOOOOOOOOOOOOOOOOOO  OO  666,666,666.66-    6,666,666.66-
OOOOOOOOOOOOOOOOOOOOOOOO  66666  OOOOOOOOOOOOOOOOOOOOO  OO  666,666,666.66-    6,666,666.66-
(VNDNAME)                (VNDNBR) (VNDCITY)           (VNDSTATE)  (VNDPRCHYTD)     (VNDBALANCE)

Number of Active Vendors        666,666,666
                                  (COUNT)
Total YTD Purchase Value  66,666,666,666.66-
                                (TOTPURCH)
Total Current Amount Due     666,666,666.66-
                                  (TOTBAL)
```

__ 2. Purchasing said to tell you that they allow you to exercise a little creativity in your report design as long as it is close to the above layout. They also have provided the field names for you to use.

Notice that the length of **TOTPURCH** is two digits longer than **VNDPRCHYTD**. The same is true for **TOTBAL** which is two digits longer than **VNDBALANCE**. You need to define these fields in your printer file.

__ 3. Also, here is the DDS for the logical file, `VNDNAM_LF`, that is based on the `VENDOR_PF` physical file:

```
A*********************************************************************
A*  Vendors by Name LF:  VNDNAM_LF
A*********************************************************************
A                                      ALTSEQ(QSYSTRNTBL)
A              R VENDOR_FMT            PFILE(VENDOR_PF)
A              K VNDNAME
```

__ 4. Compile your copy of the `VNDNAM_LF` logical file.

__ 5. Review the sample output provided:

```
PAGE    1                        Balance Due Vendor Report                        1/31/02

    Vendor Name and       Number  Vendor City       State        YTD Purchases    Current Bal. Due

AAA Pencils, Etc.         10046   Osbourne          FL           20,896.50        1,575.55
ACE Distributors          10019   Buckingham        ND           20,000.00        599.00
Allen, Allen and Allen    10036   Batavia           PA           100,000.00       10,000.00
Andy Glover Corp.         10006   Montgomery        AL           25,000.00        5,000.00
Best Furniture            10017   Chamblee          GA           25,000.00        .00
Brand Names Electronics   10031   Oak Terrace       NY           70,000.00        30,000.00
Clifford Distribution     10010   Las Vegas         NV           500,000.00       47,500.00
Collier Office Products   10040   Collier           CO           3,000.00         250.00
Electronic Paper          10033   High Falls        MI           650,000.00       125,000.00
Federal Paper             10025   Minneapolis       MN           600,000.00       300,000.00
Fetzner & Fetzner         10022   Syracuse          NH           50,000.00        750.00
G & K Office Supply       10042   Audubon           CT           1,000,000.00     275,000.00
Genessee Office Products  10050   Binghamton        NY           50,000.00        15,000.00
George Rudolph Inc.       10011   Raphael           TX           2,500.00         500.00
::::::::::::::::::
::::::::::::::::::

PAGE    3                        Balance Due Vendor Report                        1/31/02

    Vendor Name and       Number  Vendor City       State        YTD Purchases    Current Bal. Due

Susquehanna Supply        10049   Chenango Valley   PA           150,000.00       35,000.00
The Chair Market          10007   Hardford          CT           30,000.00        5,000.00
The Desk Shop             10004   San Francisco     CA           20,887.00        1,987.59
Thomas Corley             10035   Lockport          IL           23,000.00        5,000.00
Thornton Paper Supplies   10028   Vestal            NY           60,000.00        7,500.00
Tom Brayton Enterprises   10014   Courtland         PA           45,000.00        15,000.00
Winthrop Bliss Inc.       10023   Center Village    NY           400,000.00       5,500.00
World Wide Paper Inc.     10048   Glens Falls       IL           25,000,000.00    502,500.00

Number of Active Vendors                  46

Total YTD Purchase Value          38,748,188.50

Total Current Amount Due           2,914,762.10
```

Part 2: Code the printer file (PRTF) VNPADR04

When you code your DDS, you can use either LPEX or SEU. The report layout with the ruler should provide you the information you need to produce the DDS.

RDP includes an excellent design tool, the DDS print designer tool. If you already know how to use this tool, you can use it to develop your DDS. We do not teach you how to use this tool in this class. If you do not know how to use the designer, you should create the DDS using the LPEX editor or SEU.

__ 6. Use the printer layout provided to create the record formats to produce the heading, the detail information for each record selected by the RPG IV program, and the totals at the bottom of the report.

__ 7. Take care that if you move any fields around that you do not overlay any other fields. Look for error message **CPD7807** in your spooled output file when you attempt to create your printer file. The **PRTF** is created, but this message is issued as a warning.

__ 8. Notice that the total fields for `YTD purchases` and `YTD balance` owed can be defined by either referencing the fields upon which they are based or explicitly in your **PRTF** DDS.

One of the benefits of definition by reference is that if the fields in the field reference file (in our application, **DICTIONARY**) change in definition, the fields that reference them will also change in size when you re-create the **PRTF**. All you might have to change are spaces between fields. Redefinition is unnecessary.

__ 9. When you have coded your DDS (type **PRTF**), create (**compile**) your **PRTF**. You do not have to modify any of the defaults for overflow line, and so on.

Part 3: Code the RPG IV program, VNRADR04

As you write this program, reference previous exercises and the lecture notes to help you to code the logic of the program. The main things to consider are that you must select only active records (as you did in the previous exercise).

__ 10. The logic is very much the same as the program **VNRADR03**. The differences are in the data written (determined by your PRTF DDS) and the totals that are accumulated.

__ 11. When you have written your program (using LPEX or SEU), **compile** it and **test** it.

End of exercise

Exercise 9. Debugging an RPG IV program

What this exercise is about

This exercise provides an opportunity to use the Source View Debugger.

What you should be able to do

At the end of the exercise, you should be able to:

- Create programs that can be debugged using source code
- Start the debugger
- Set breakpoints
- Display data
- Change data

Introduction

If you have ever written a program with run-time errors and you dumped the program variables hoping to determine what caused your program to fail, you welcome the features of the i-based source view debugger.

Another programmer has been working with a program that has been in production. The programmer made some minor changes and now the program does not work the way we expect it to work. We have come to you for help with this very important program.

You are given member **VNRADR05** in source file AS06*nnn*/`QRPGLESRC`.

Compile **VNRADR05** to allow debugging with source and then correct the errors in the program.

Exercise instructions

Part 1: Run the working program

__ 1. Call **VNRADR05S**, the working version of the program. This program produces the active vendor report.

__ 2. Check the report produced. How many vendors are reported? ____

Part 2: Compile and run the program that contains bugs

__ 3. Compile member **VNRADR05** in source file AS06*nnn*/`QRPGLESRC` with debugging views `*ALL`. You can specify `*ALL` as a compilation option when in LPEX (compile) or when using PDM/SEU.

__ 4. After the program **VNRADR05** is compiled, call the program.

__ 5. Check the report produced. How many vendors are reported? ____

Part 3: Set up the debug session

__ 6. Begin debugging the program **VNRADR05** using the `STRDBG` command.

__ 7. Do you see your source code?

__ 8. Note the function key descriptions on the debug screen.

__ 9. Place an unconditional breakpoint on the **DoWhile** line of code.

__ 10. Spend a few minutes scrolling through the program.

__ 11. Notice the F-spec for the printer file. The file is *program described* and that means that the output specifications are coded in the program. Scroll down to the output specifications and you see the code that describes printer output. In this class, we have taught you the techniques to use to write efficient and easily maintained RPG IV code.

However, in your career, you are certain to encounter some code written using program described output.

Part 4: End your debug session setup and test your program

__ 12. Use **F12** to end the setup of your debug session.

__ 13. Call your program, **VNRADR05**.

__ 14. At what breakpoint does the program stop? Why did it stop at this breakpoint?

_____

_____

__ 15. Notice the message in the lower left corner of your display. You often see debug feedback in this area of the display.

__ 16. Press **F10** to step to the next executable statement. Where does the program step to and stop?

_____

When you press **F10 (Step)**, the next statement of the module object shown in the display module source display is run, and the program object is stopped again. In the case of code that is conditioned, the condition is tested and the next executable statement is executed as determined by the result of testing the condition.

__ 17. Why did it step to this point? _____

__ 18. Use the **debug** function key to display the variables *VndActive* and *VndSales*.

__ 19. Use the **EVAL** command on the debug command line to display the value of the field name VndNbr. Does it contain a valid value? _____

The value of a variable can be checked and can also be changed using the EVAL command.

__ 20. Now you know that the program read at least one record. Should the program not have processed it in the DoW loop (assuming it is correctly coded)?

_____

__ 21. Having used the **VENDOR_PF** file, with the solution at the beginning of this exercise, you know that there are records in the file. Why did we not enter the loop?

_____

_____

__ 22. You probably have discovered the problem with the loop and know what to do to correct it. You can correct the code in the next step.

__ 23. End the program.

Part 5: Fix the program

__ 24. Correct the program bug you have found, save it, and recompile it again with ***ALL** debug option. If you had problems saving your changed source code, what happened?

__ 25. You can test your program again. Did you encounter another error?____ What is the error?

_____

__ 26. Cancel the program assuming that it has ended with an error.

Part 6: Continue to debug

__ 27. STRDBG again and this time, set a breakpoint again at the line of code:

```
Count = Count + 1;
```

__ 28. Use **F6** to place a second breakpoint on the `READ` operation inside the loop.

__ 29. End setup of your debug session with **F12**.

__ 30. Call your program. This time, you should enter the loop and the program should stop at the line of code that increments `COUNT`.

__ 31. Display the value of `COUNT`.

__ 32. Continue the debug session using **F12**. Your program should stop at the read statement. Use **`Eval`** or **F11** to check whether the value of the field, **VndNbr,** has changed and contains a valid value. It should as we know this file contains 46 records. You can remove the breakpoint on the `READ`, once you know that new records are being read.

__ 33. Continue the debug session using **F12**. Display the value of `Count` as you stop at the breakpoint on the statement that increments `COUNT`. Observe the value of `COUNT` as it increases to **9**.

__ 34. When `COUNT` displays a value of **9**, press **F12** one final time.

__ 35. You receive a run-time error message.

__ 36. Why does this error occur? _____

Part 7: Correct the source program

__ 37. Use **F3** to end the program.

__ 38. Edit the source member. Did you determine the error in the previous step? Correct the error.

__ 39. Exit the editor to save the changes and compile and test the program.

__ 40. Do you see a message "`You are in debug mode`"?

Why does this happen?

_____

You can use the **function** key in SEU to request a system command line and end the debugger. If you are using LPEX, just go to your **5250** session and end the debugger on a command line.

__ 41. Exit the **editor** and **save** your member.

Part 8: Compile and test the program

__ 42. Compile your source member, **VNRADR05**.

__ 43. Check to make sure the program compiled successfully.

__ 44. Call **VNRADR05**. It should run with no errors.

__ 45. Look at the spooled file and verify there are **46** records in the report.

End of exercise

Exercise 10.Coding subroutines

What this exercise is about

In this exercise, you analyze the transactions that are used to update and maintain a database file.

What you should be able to do

At the end of the exercise, you should be able to:

- Code subroutines using BegSR and EndSR
- Code ExSR to execute a subroutine

Introduction

This exercise is the first of two exercises that work with the same application.

Management has asked that you modify purchase order data with updates that have been entered into a transaction file. You are given the transaction data file with several transactions already entered into it. You are also given a PRTF that produces a simple transaction log.

The transactions update the data by using a logical view of the data.

- Review the transaction file, `POTRANS_PF`.
- Review the PRTF that defines the audit report layout, `POPMNT02`.
- Code the RPG IV program that analyzes the transactions and lists them on the report. Use the same PRTF that is used in the next exercise.

Exercise instructions

Part 1: Analyze the output report

__ 1. Review the following report that is the desired output that your maintenance program in the next exercise should produce. For now, you are not updating the file. You simply analyze the transactions.

Your final report in the next exercise produces results similar to this:

```
...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9....+...10....
     16:08:04         Open Purchase Order File Maintenance Log    1/31/03

Trans
 Code  P.O. #       Item #   Qty OO    Unit Cost Qty Recd. Date Recd. Status Message

  A    300002      20023          5      100.00       5    2000/05/17    C   Record Added
       300005      20002          6       89.00       0                      Before Transaction Processed
  C    300005      20002         13         .00       0                      Record Changed
  D    300007      20028          0         .00       0                      Record Deleted
       300004      20013          3      100.00       3    1996/06/29    C   Before Transaction Processed
  C    300004      20013          7         .00       0                      Record Changed
  A    300002      20028          5      100.00       5    2000/05/17    C   Record Added
  D    300001      20070          0         .00       0                      Record not found
  C    300004      20071         25         .00       0                      Record not found
       300007      98877          0         .00       0                      Invalid Transaction
  A    300002      20023          5      100.00       5    2000/05/17    C   Record already exists - not added
  A    300002      20028          5      100.00       5    2000/05/17    C   Record already exists - not added
  d    300001      20007          0         .00       0                      Invalid Transaction
  c    300002      20015          8       85.00       0    2000/05/22        Invalid Transaction
  D    300001      20007          0         .00       0                      Record Deleted
       300002      20015         10        2.50       0                      Before Transaction Processed
  C    300002      20015          8       85.00       0    2000/05/22        Record Changed
  A    300002      20015          8       85.00       0    2000/05/22        Record already exists - not added
  A    300009      20025         35       25.00       0                      Record Added
  A    300007      20035         25       25.00       0                      Record Added

         Additions                      4
         Deletions                      2
         Changes                        3
         Invalid Trans                  8

         Total Transactions            17
```

__ 2. For now, your report should be similar to the one below, using only the **HEADING** and **TOTAL** formats. (Note that the totals differ between the two examples. Can you determine why this might be? The reason becomes clear during the next exercise.)

```
10:17:16         Open Purchase Order File Maintenance Log    7/03/03
Trans
 Code  P.O. #       Item #   Qty OO    Unit Cost Qty Recd. Date Recd. Status M
         Additions                          7
         Deletions                          3
         Changes                            4
         Invalid Trans                      3
         Total Transactions                17
```

__ 3. The PRTF, which is provided to you, is named **POPMNT02**. Review the **DDS** below and review the sample report as you do so. In this part of the exercise, you only reference the **HEADING** and **TOTAL** print formats. What variables are used to accumulate the total values?

_____

_____

_____

_____

_____

_____

```
*                           POPMNT02                            *
*****************************************************************
 * This display file prompts for line item maintenance.          *
 *                                                               *
 * INDICATORS:                                                   *
 *   10 - Record not found                                       *
 *   11 - Record already exists                                  *
 *   12 - Record added                                           *
 *   13 - Record deleted                                         *
 *   14 - Record changed                                         *
 *   15 - Invalid Transaction
 *****************************************************************
A                                     REF(POTRANS_PF POLINE_PF)
A          R HEADING                  SKIPB(2)
A                                     5TIME
A                                    22'Open Purchase Order File Maintenan-
A                                      ce Log'
A                                    65DATE EDTCDE(Y)
A                                    80'Page'
A                                    +1PAGNBR EDTCDE(J)
A                                      SPACEA(2)
A**
A                                     1'Trans'
A                                     2'Code'
                                        SPACEB(1)
A                                     8'P.O. #'
A                                    21'Item #'
A                                    30'Qty OO'
A                                    40'Unit Cost'
A                                    50'Qty Recd.'
A                                    60'Date Recd.'
A                                    71'Status'
A                                    78'Message'
A                                      SPACEA(2)
A
A          R TRNDETAIL                 SPACEA(1)
A            TRNCODE    R             3REFFLD(POTRAN_FMT/TRNCODE *LIBL/POT-
A                                      RANS_PF)
A            POTPONBR   R             8REFFLD(POTRAN_FMT/POTPONBR *LIBL/PO-
A                                      TRANS_PF)
A            POTITMNBR R              20REFFLD(POTRAN_FMT/POTITMNBR *LIBL/P-
A                                      OTRANS_PF)
A            POTQTYOO   R             30EDTCDE(J) REFFLD(POTRAN_FMT/POTQTYO-
```

```
A                                        O *LIBL/POTRANS_PF)
A            POTITMCOSTR                42REFFLD(POTRAN_FMT/POTITMCOST *LIBL/-
A                                        POTRANS_PF) EDTCDE(J)
A            POTQTYREC R                50EDTCDE(J) REFFLD(POTRAN_FMT/POTQTYR-
A                                        EC *LIBL/POTRANS_PF)
A            POTDATREC R                60REFFLD(POTRAN_FMT/POTDATREC *LIBL/P-
A                                        OTRANS_PF) EDTWRD('    /  /  ')
A            POTSTATUS R                74REFFLD(POTRAN_FMT/POTSTATUS *LIBL/P-
A                                        OTRANS_PF)
A  10                                   78'Record not found'
A  11                                   78'Record already exists - not added'
A  12                                   78'Record Added'
A  13                                   78'Record Deleted'
A  14                                   78'Record Changed'
A  15                                   78'Invalid Transaction'
A
A          R POORIG                      SPACEA(2)
A            PONBR     R                 8REFFLD(POLINE_FMT/PONBR *LIBL/POLIN-
A                                        E_PF)
A            ITMNBR    R                20REFFLD(POLINE_FMT/ITMNBR *LIBL/POLI-
A                                        NE_PF)
A            POLQTYOO  R                30EDTCDE(J) REFFLD(POLINE_FMT/POLQTYO-
A                                        O *LIBL/POLINE_PF)
A            POLITMCOSTR                42REFFLD(POLINE_FMT/POLITMCOST *LIBL/-
A                                        POLINE_PF) EDTCDE(J)
A            POLQTYREC R                50EDTCDE(J) REFFLD(POLINE_FMT/POLQTYR-
A                                        EC *LIBL/POLINE_PF)
A            POLDATREC R                60REFFLD(POLINE_FMT/POLDATREC *LIBL/P-
A                                        OLINE_PF) EDTWRD('    /  /  ')
A            POLSTATUS R                74REFFLD(POLINE_FMT/POLSTATUS *LIBL/P-
A                                        OLINE_PF)
A                                       78'Before Transaction Processed'
A
A          R TOTAL                       SPACEB(3)
A                                       10'Additions'
A            ADDCNT         5  0        40EDTCDE(Z) SPACEA(2)
A                                       10'Deletions'
A            DELCNT         5  0        40EDTCDE(Z) SPACEA(2)
A                                       10'Changes'
A            CHGCNT         5  0        40EDTCDE(Z) SPACEA(2)
A                                       10'Invalid Trans'
A            ERRCNT         5  0        40EDTCDE(Z) SPACEA(3)
A                                       10'Total Transactions'
A            TOTCNT         5  0        40EDTCDE(Z)
```

__ 4. Compile the **POPMNT02** printer file in your library.

Part 2: Analyze the purchase order transaction file

__ 5. Review the **DDS** of the following transaction file. The transactions are already entered and are provided to you in your library.

```
A*****************************************************************
A*  PO line Transaction PF:  POTRANS_PF
A*****************************************************************
A                                      REF(DICTIONARY)
A          R POTRAN_FMT                TEXT('PO Transaction Record')
A            TRNCODE        1A         TEXT('Transaction Code')
A                                      COLHDG('Trans' 'Code')
A            POTPONBR  R               REFFLD(PONBR)
A            POTITMNBR R               REFFLD(ITMNBR)
A            POTQTYOO  R               REFFLD(POLQTYOO)
A            POTITMCOSTR               REFFLD(POLITMCOST)
A            POTDATREC R               REFFLD(POLDATREC)
A            POTQTYREC R               REFFLD(POLQTYREC)
A            POTSTATUS R               REFFLD(POLSTATUS)
```

__ 6. Each transaction record in the file is identified by a transaction type, (the **TRNCODE** field):

- **A** for record Addition
- **C** for record Update (Change)
- **D** for record Deletion

You also notice from the report that some records in the file have an incorrect transaction code, (or none at all).

Part 3: Code the program

Write the program **PORMNTSUM** to print a summary of the information contained within the **POTRANS_PF**.

Use the existing printer file **POPMNT02**.

Read through the POTRANS_PF sequentially, identifying the transaction type. For each of the recognized transaction codes, increment the appropriate record count. Invalid codes also need to be counted.

Here are some points to consider:

__ 7. Ensure that heading information is printed at the top of the report.

__ 8. Use a **select** group, (or, if you prefer, an If/Elseif/Else group) to test for each transaction code.

__ 9. Modularize your code by using separate subroutines to process each transaction (that is, increment the corresponding record count in a subroutine).

__ 10. When all transaction records have been read from **POTRANS_PF**, print the summary information. Remember to check for page overflow.

__ 11. Code and compile your program.

Part 4: Run the PORMNTSUM program

__ 12. Run your program to test it. Make sure that it produces the desired report.

Part 5: Modify your PORMNTSUM program

Enhance your program to print the transaction detail as well as the summary. You should use the **TRNDETAIL** print format. Your report should be similar to this one:

```
    13:11:12         Open Purchase Order File Maintenance Log    7/29/03
Trans
 Code  P.O. #       Item #   Qty OO    Unit Cost Qty Recd. Date Recd. Status Message
  A    300002      20023          5      100.00       5    2000/05/17    C
  C    300005      20002         13         .00       0
  D    300007      20028          0         .00       0
  C    300004      20013          7         .00       0
  A    300002      20028          5      100.00       5    2000/05/17    C
  D    300001      20070          0         .00       0
  C    300004      20071         25         .00       0
       300007      98877          0         .00       0                       Invalid Transaction
  A    300002      20023          5      100.00       5    2000/05/17    C
  A    300002      20028          5      100.00       5    2000/05/17    C
  d    300001      20007          0         .00       0                       Invalid Transaction
  c    300002      20015          8       85.00       0    2000/05/22        Invalid Transaction
  D    300001      20007          0         .00       0
  C    300002      20015          8       85.00       0    2000/05/22
  A    300002      20015          8       85.00       0    2000/05/22
  A    300009      20025         35       25.00       0
  A    300007      20035         25       25.00       0
         Additions                          7
         Deletions                          3
         Changes                            4
         Invalid Trans                      3
         Total Transactions                17
```

__ 13. Code a new subroutine to print the **TRNDETAIL** format, checking first for page overflow.

__ 14. Execute your new subroutine from each transaction processing routine.

__ 15. Invalid transactions should also be flagged with an appropriate message. Review the printer file **DDS** to identify the message and corresponding conditioning indicator. You might want to rename the indicator using the techniques employed in previous exercises.

__ 16. Compile and test your program. Compare your results with the sample report. Your report should look very similar to the one at the top, except that the messages only show several *invalid transaction* items as you are not updating the physical file.

End of exercise

Exercise 11.Maintaining database files

What this exercise is about

In this exercise, you update a database file based on transactions read from another file. These transactions are the ones that you listed in the previous exercise.

What you should be able to do

At the end of the exercise, you should be able to:

- Use various opcodes to access database records
- Write a file maintenance program
- Use `CPYF` to populate a DB file with data

Introduction

Management has asked that you modify purchase order data with updates that already have been entered into a transaction file. You are given the transaction data file (**POTRANS_PF**) with several transactions already entered into it. You are given a PRTF (**POPMNT02**) that produces a simple transaction log.

The transactions update the data by using a logical view of the data. This LF is also provided.

You are also given a program that does most of the job. You add the logic that processes one type of transaction. Since you now know how to code subroutines, write your code in a subroutine. Your responsibility is to:

- Review the purchase orders open line items file, **POOPNLI_LF**.
- Review the transaction file, **POTRANS_PF**.
- Review the **PRTF** that defines the audit report layout, **POPMNT02**.
- Modify the **RPG IV** program to process the transactions, update the purchase orders line items file and produce the audit trail.

Exercise instructions

Part 1: Analyze the output report

__ 1. Review the following report that is the desired output that your program should produce.

```
...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9....+...10....
     16:08:04         Open Purchase Order File Maintenance Log    1/31/02

Trans
 Code  P.O. #       Item #   Qty OO    Unit Cost Qty Recd. Date Recd. Status Message

  A    300002      20023          5      100.00       5    2000/05/17    C   Record Added
       300005      20002          6       89.00       0                      Before Transaction Processed
  C    300005      20002         13         .00       0                      Record Changed
  D    300007      20028          0         .00       0                      Record Deleted
       300004      20013          3      100.00       3    1996/06/29    C   Before Transaction Processed
  C    300004      20013          7         .00       0                      Record Changed
  A    300002      20028          5      100.00       5    2000/05/17    C   Record Added
  D    300001      20070          0         .00       0                      Record not found
  C    300004      20071         25         .00       0                      Record not found
       300007      98877          0         .00       0                      Invalid Transaction
  A    300002      20023          5      100.00       5    2000/05/17    C   Record already exists - not added
  A    300002      20028          5      100.00       5    2000/05/17    C   Record already exists - not added
  d    300001      20007          0         .00       0                      Invalid Transaction
  c    300002      20015          8       85.00       0    2000/05/22        Invalid Transaction
  D    300001      20007          0         .00       0                      Record Deleted
       300002      20015         10        2.50       0                      Before Transaction Processed
  C    300002      20015          8       85.00       0    2000/05/22        Record Changed
  A    300002      20015          8       85.00       0    2000/05/22        Record already exists - not added
  A    300009      20025         35       25.00       0                      Record Added
  A    300007      20035         25       25.00       0                      Record Added

         Additions                      4
         Deletions                      2
         Changes                        3
         Invalid Trans                  8

         Total Transactions            17
```

__ 2. Currently, the add transactions are treated as invalid transactions as you can determine from the current report:

```
16:05:54         Open Purchase Order File Maintenance Log    1/31/02       Page     1

Trans
 Code  P.O. #       Item #   Qty OO    Unit Cost Qty Recd. Date Recd. Status Message

  A    300002      20023          5      100.00       5    2000/05/17    C   Invalid Transaction
       300005      20002          6       89.00       0                        Before Transaction Processed
  C    300005      20002         13         .00       0                        Record Changed
  D    300007      20028          0         .00       0                        Record Deleted
       300004      20013          3      100.00       3    1996/06/29    C   Before Transaction Processed
  C    300004      20013          7         .00       0                        Record Changed
  A    300002      20028          5      100.00       5    2000/05/17    C   Invalid Transaction
  D    300001      20070          0         .00       0                        Record not found
  C    300004      20071         25         .00       0                        Record not found
       300007      98877          0         .00       0                        Invalid Transaction
  A    300002      20023          5      100.00       5    2000/05/17    C   Invalid Transaction
  A    300002      20028          5      100.00       5    2000/05/17    C   Invalid Transaction
  d    300001      20007          0         .00       0                        Invalid Transaction
  c    300002      20015          8       85.00       0    2000/05/22        Invalid Transaction
  D    300001      20007          0         .00       0                        Record Deleted
       300002      20015         10        2.50       0                        Before Transaction Processed
  C    300002      20015          8       85.00       0    2000/05/22        Record Changed
  A    300002      20015          8       85.00       0    2000/05/22        Invalid Transaction
  A    300009      20025         35       25.00       0                        Invalid Transaction
  A    300007      20035         25       25.00       0                        Invalid Transaction

         Additions
         Deletions                    2
         Changes                      3
         Invalid Trans               12

         Total Transactions          17
```

__ 3. The PRTF, which is provided to you, is named `POPMNT02`. Review the **DDS** and review the sample report as you do so.

__ 4. The **POPMNT02** printer file should have been compiled in the previous exercise and should exist in your library.

```
     *                          POPMNT02                             *

A                                      REF(POTRANS_PF POLINE_PF)
A          R HEADING                   SKIPB(2)
A                                     5TIME
A                                    22'Open Purchase Order File Maintenan-
A                                      ce Log'
A                                    65DATE EDTCDE(Y)
A                                    80'Page'
A                                    +1PAGNBR EDTCDE(J)
A                                      SPACEA(2)
A**
A                                     1'Trans'
A                                     2'Code'
                                        SPACEB(1)
A                                     8'P.O. #'
A                                    21'Item #'
A                                    30'Qty OO'
A                                    40'Unit Cost'
A                                    50'Qty Recd.'
A                                    60'Date Recd.'
A                                    71'Status'
A                                    78'Message'
A                                      SPACEA(2)
A
A          R TRNDETAIL                 SPACEA(1)
A            TRNCODE   R               3REFFLD(POTRAN_FMT/TRNCODE *LIBL/POT-
A                                      RANS_PF)
A            POTPONBR  R               8REFFLD(POTRAN_FMT/POTPONBR *LIBL/PO-
A                                      TRANS_PF)
A            POTITMNBR R              20REFFLD(POTRAN_FMT/POTITMNBR *LIBL/P-
A                                      OTRANS_PF)
A            POTQTYOO  R              30EDTCDE(J) REFFLD(POTRAN_FMT/POTQTYO-
A                                      O *LIBL/POTRANS_PF)
A            POTITMCOSTR              42REFFLD(POTRAN_FMT/POTITMCOST *LIBL/-
A                                      POTRANS_PF) EDTCDE(J)
A            POTQTYREC R              50EDTCDE(J) REFFLD(POTRAN_FMT/POTQTYR-
A                                      EC *LIBL/POTRANS_PF)
A            POTDATREC R              60REFFLD(POTRAN_FMT/POTDATREC *LIBL/P-
A                                      OTRANS_PF) EDTWRD('    /  /  ')
A            POTSTATUS R              74REFFLD(POTRAN_FMT/POTSTATUS *LIBL/P-
A                                      OTRANS_PF)
A  10                                 78'Record not found'
A  11                                 78'Record already exists - not added'
A  12                                 78'Record Added'
A  13                                 78'Record Deleted'
A  14                                 78'Record Changed'
A  15                                 78'Invalid Transaction'
A
A          R POORIG                    SPACEA(2)
A            PONBR     R               8REFFLD(POLINE_FMT/PONBR *LIBL/POLIN-
A                                      E_PF)
```

```
A            ITMNBR    R            20REFFLD(POLINE_FMT/ITMNBR *LIBL/POLI-
A                                       NE_PF)
A            POLQTYOO  R            30EDTCDE(J) REFFLD(POLINE_FMT/POLQTYO-
A                                       O *LIBL/POLINE_PF)
A            POLITMCOSTR            42REFFLD(POLINE_FMT/POLITMCOST *LIBL/-
A                                       POLINE_PF) EDTCDE(J)
A            POLQTYREC R            50EDTCDE(J) REFFLD(POLINE_FMT/POLQTYR-
A                                       EC *LIBL/POLINE_PF)
A            POLDATREC R            60REFFLD(POLINE_FMT/POLDATREC *LIBL/P-
A                                       OLINE_PF) EDTWRD('    /  /  ')
A            POLSTATUS R            74REFFLD(POLINE_FMT/POLSTATUS *LIBL/P-
A                                       OLINE_PF)
A                                   78'Before Transaction Processed'
A
A          R TOTAL                    SPACEB(3)
A                                   10'Additions'
A            ADDCNT         5  0     40EDTCDE(Z) SPACEA(2)
A                                   10'Deletions'
A            DELCNT         5  0     40EDTCDE(Z) SPACEA(2)
A                                   10'Changes'
A            CHGCNT         5  0     40EDTCDE(Z) SPACEA(2)
A                                   10'Invalid Trans'
A            ERRCNT         5  0     40EDTCDE(Z) SPACEA(3)
A                                   10'Total Transactions'
A            TOTCNT         5  0     40EDTCDE(Z)
```

Part 2: Analyze the purchase orders open line item file

```
A*********************************************************************
A*  PO Open Line Item LF:  POOPNLI_LF
A*********************************************************************
A
A          R POLINE_FMT                TEXT('PO Line Item Record')
A                                      PFILE(POLINE_PF)
A          K PONBR
A          K ITMNBR
```

__ 5. Browse the data in the **POLINE_PF** file. While you are processing the **POOPNLI_LF** logical file, you need to understand the physical, **POLINE_PF**, on which it is based.

An SQL SELECT statement is provided to you in your **QRPGLESRC** file. It is named LAB11SQL1. This statement displays the **POLINE_PF** data used in the program.

To run the SQL statement, you need to copy it. To do this, use the **COPY** facility of the PC 5250 emulator (see the tool bar) to copy the text in your member. Next, on a command line, enter **STRSQL** and press **Enter**. You see a command screen with the heading Enter SQL Statements. Use the **page down** key if the screen shows only a partial page of empty command lines. Paste your **SELECT** statement to the right of the arrow (**===>**) and press **Enter**. You see the results of your query. Once you understand the results, press **Enter**, and then use **F3** to exit SQL. When prompted to exit, press **Enter**.

If you have any problems, ask your instructor for assistance.

Part 3: Analyze the purchase order transaction file

__ 6. Review the **DDS** of the following transaction file. The transactions are already entered and are provided to you in your library.

__ 7. Browse the data in the **POTRANS_PF** file.

Another SQL SELECT statement is provided to you in your **QRPGLESRC** file. It is named **LAB11SQL2**.

This statement displays the **POTRANS_PF** data used in the program.

To run the SQL statement, you need to copy it as well. To do this, use the **COPY** facility of the PC 5250 emulator (see the tool bar) to copy the text in your member. Next, on a command line, enter **STRSQL** and press **Enter**. You see a command screen with the heading Enter SQL Statements. Use the **page down** key if the screen shows only a partial page of empty command lines. Paste your **SELECT** statement to the right of the arrow (===>) and press **Enter**. You see the results of your query. When you understand the results, press **Enter** and then use **F3** to exit SQL. When prompted to exit, press **Enter**.

If you have any problems, ask your instructor for assistance.

```
A*****************************************************************
A*  PO line Transaction PF:  POTRANS_PF
A*****************************************************************
A                                       REF(DICTIONARY)
A          R POTRAN_FMT                 TEXT('PO Transaction Record')
A            TRNCODE        1A         TEXT('Transaction Code')
A                                       COLHDG('Trans' 'Code')
A            POTPONBR  R                REFFLD(PONBR)
A            POTITMNBR R                REFFLD(ITMNBR)
A            POTQTYOO  R                REFFLD(POLQTYOO)
A            POTITMCOSTR                REFFLD(POLITMCOST)
A            POTDATREC R                REFFLD(POLDATREC)
A            POTQTYREC R                REFFLD(POLQTYREC)
A            POTSTATUS R                REFFLD(POLSTATUS)
```

Part 4: Program description

The program named PORMNT02 reads the **POTRANS_PF** file sequentially. Based upon the transaction type, the current program changes an existing purchase order record or deletes an existing record. All other transactions are considered to be invalid. You are responsible for enhancing the existing program to handle an add transaction and for ensuring that the transaction is processed correctly. See the sample report to understand the transactions that you process.

Here is a more detailed description of the existing program:

__ 8. The PORMNT02 program handles page overflow, as your other programs do.

__ 9. For each transaction, PORMNT02 prints the **POTRANS_PF** transaction whether the transaction is valid or invalid.

__ 10. For a delete transaction, PORMNT02 makes sure that the record to be deleted exists and deletes it if found. The program prints the appropriate message and adds to the delete counter for each valid delete transaction. If a record to be deleted is not found, the programs print an error message and adds one to the error counter.

__ 11. For a change transaction, PORMNT02 checks first that the record exists. If it does, it prints the **POOPNLI_LF** record before the **POTRANS_PF** update is applied; then, the program updates the fields changed by the **POTRANS_PF** record, prints the change message and adds one to the change counter.

If the change is invalid, PORMNT02 prints an error message and add to the error counter.

__ 12. When all transactions are processed, the program calculates the total number of transactions processed, and prints not only the total of all transactions, but also the sum of each type of transaction as shown on the report.

Part 5: Modify the PORMNT02 program

Your job is to add the logic to handle the add transaction, indicated by a transaction code of 'A'.

There are two places in the existing program that need your attention:

__ 13. In the SELECT group, check for an add transaction and execute the ADD subroutine, which should be named AddSR.

__ 14. Code the subroutine to process a transaction with the code 'A':

__ a. Make sure that the record does not already exist. Use an I/O opcode that returns a found or not found condition

__ b. If the add transaction is valid:

i. Print the appropriate message.

ii. Add one to the add counter.

iii. Build fields of the new record using the data from the **POTRANS_PF**.

iv. Initialize any fields that are not referenced in the **POTRANS_PF**.

v. Write the new record to the file.

__ c. Handle an invalid addition by:

i. Printing an error message as shown on the report,

ii. Adding one to the error count field.

__ 15. Review the code for the change and delete transactions. The process for much of your add logic is similar.

__ 16. Review the current program and understand the logic. Notice the use of named indicators.

Notice the modular coding style employing many subroutines:

```
 //                          PORMNT02                                  *
 //                  PO Maintenance - Open Line Items                  *
 //********************************************************************
 //                                                                    *
 // This program allows changes to the Open Line Items:                *
 //        Additions                                                   *
 //        Deletions                                                   *
 //        Changes                                                     *
 //                                                                    *
 // INDICATORS:                                                        *
 //   10 - Record not found                                            *
 //   11 - Record exists (duplicate add)                               *
 //   12 - Record added                                                *
 //   13 - Record deleted                                              *
 //   14 - Record changed                                              *
 //   15 - Invalid Transaction                                         *
 //********************************************************************
 // PO Transaction File
FPOTRANS_PFIF   E             DISK
 // Open Line Items File
FPOOPNLI_LFUF A E           K DISK
 // Maintenance Printer File
FPOPMNT02  O    E             PRINTER OFLIND(OverFlow) INDDS(Indicators)

D Indicators      DS
D  RecNotFound           10     10N
D  RecExists             11     11N
D  RecAdded              12     12N
D  RecDeleted            13     13N
D  RecChanged            14     14N
D  InvalidTrans          15     15N


 // Build the Line Item Key
C     LItmKey       KLIST
C                   KFLD                    POTPONBR
C                   KFLD                    POTITMNBR

 // Write headings on first page of report
 /FREE
  Write Heading;

  // Read first transaction record
  Read POTrans_PF;

  DoW not %eof(POTrans_PF);

    // Reset  Indicators
    RecNotFound = *off;
    RecExists = *off;
```

```
  RecAdded = *off;
  RecDeleted = *off;
  RecChanged = *off;
  InvalidTrans = *off;

  // Process transactions
  Select;

  // Add the code to  check for and handle an Add transacation here

  When Trncode = 'D';
    Exsr DelSr;

  When Trncode = 'C';
    Exsr ChgSr;

  Other;
    Exsr InvTrSr;

  EndSl;

  // Read next transaction record
  Read POTrans_PF;

EndDo;                                                    //END OF DO

// EOJ Processing
Exsr PrtTotals;
*inlr = *on;

// Additions - Insert your subroutine to handle additions here









// Deletions
BEGSR DelSR;

  // Delete the record
  // If no record found (Error), set RecNotFound
  // If record exists (OK), delete and set RecDeleted
  DELETE LItmKey POOPNLI_LF;
  If %found(POOPNLI_LF);
    RecDeleted = *ON;
```

```
      PONbr = PotPONbr;
      Itmnbr =  POTITMNBR;
      Delcnt = Delcnt + 1;

    Else;
      RecnotFound = *on;
      Errcnt = Errcnt + 1;
    EndIf;

    Exsr PrtDetail;
ENDSR;

// Changes
BEGSR ChgSr;
    // Check for record
    CHAIN LItmKey POOPNLI_LF;

    If %found(POOPNLI_LF);
      // Print Original fields
      Exsr PrtPOOrig;
      // Update quantity
      ponbr = potponbr;
      ITMNBR = potitmnbr;
      POLQTYOO = POTQTYOO;
      POLITMCOST = POTITMCOST;
      POLDATREC = POTDATREC;
      POLQTYREC = POTQTYREC;
      POLSTATUS = POTSTATUS;
      // Change the record
      UPDATE POLINE_FMT;
      // If record exists, set RecChanged
      RecChanged = *ON;
      Chgcnt = Chgcnt + 1;
    Else;
      RecnotFound = *on;  // Set if no record found
      Errcnt = Errcnt + 1;
    EndIf;

    // Print transaction
    Exsr PrtDetail;

ENDSR;

// Invalid Transaction
BegSr InvTrSr;
    InvalidTrans = *on;
    Errcnt = Errcnt + 1;
    Exsr PrtDetail;
    InvalidTrans = *off;
EndSr;

// Print Totals Subroutine
Begsr PrtTotals;
    TotCnt = ChgCnt + DelCnt + ErrCnt + Addcnt;
    ExSr CheckOV;
    Write Total;
```

```
EndSr;

//  Print original record fields, transaction and new fields
BegSr PrtPOOrig;
  ExSr CheckOV;
  Write POOrig;
EndSr;

//  Print transaction
BegSr PrtDetail;
  ExSr CheckOV;
  Write TrnDetail;
EndSr;

// Check for Overflow
BegSr CheckOV;
  If OverFlow;
    Write Heading;
    OverFlow = *off;
  EndIf;
EndSr;

/END-FREE
```

Part 6: Compile and test your program

__ 17. Because your program updates the **POOPNLI_LF** while you are testing, you might corrupt the data. To refresh the data in the file, you should use the **CPYF** command. **Copy** the data from the master copy that is in the AS06XXX library.

__ 18. When you are satisfied that the **POOPNLI_LF** file has been updated correctly and your report is correct, you are finished.

__ 19. Otherwise, use your debugging skills as needed.

Part 7: Modify your program

__ 20. Notice that the key is constructed using **KLIST**/**KFLD**. If you are using a V5R2 system or higher for this exercise, remove the **KLIST**/**KFLD** code.

__ 21. Construct the key to the **POOPNLI_LF** file using a parameter string.

__ 22. Compile and test your program. You need to refresh your **POLINE_PF** file before and after the test.

End of exercise

Exercise 12.Coding an inquiry program

What this exercise is about

This exercise provides an opportunity to create and use a two-format display file in an interactive program.

What you should be able to do

At the end of the exercise, you should be able to:

- Create a display file using DDS
- Write an RPG program that uses a display file for a simple inquiry application

Introduction

Company documents frequently lack complete vendor information. A program to display a vendor's entire record would be helpful.

You are given screen design information and a program description.

You create a display file and program to display a vendor's record on the screen.

Exercise instructions

Part 1: Understanding the requirements

__ 1. When the program is initially called, the user is prompted for a vendor number. There is an option to exit. See the following prompt screen format:

```
...+....1....+....2....+....3....+....4....+....5....+....6....+....7..
-AS06999                    Vendor Inquiry                        DIRTCITY
-                                                            6/01/11 08:15:58
- Vendor number. . . . : _____
-
+
-
-
-
-
+
-
-
-
-
+
-
-
-
-
+
-
- Please press enter to continue
-  F3 = Exit
-Invalid vendor number - press reset and re-enter
```

__ 2. For a valid vendor number, display the results as below:

```
...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+...
-AS06999                    Vendor Inquiry                          DIRTCITY
-                                                          2/01/02  08:15:58
-
-
+
-
- Vendor number . .:   10035
- Name . . . . . . :   Thomas Corley
- Address  . . . . :   285 Avenue B
+                      Lockport               IL  11014
- Telephone. . . . :   ( 212 )  348-5618
- Sales Person . . :   James Milburn
- Purchases YTD  . :        23,000.00
- Balance Owed . . :         5,000.00
+
-
-
-
-
+
-
-
- F3 = Exit  F12 = Return to previous Display
-
```

__ 3. If a valid vendor number is entered, the entire vendor record is displayed. See the previous figure. Note the use of function keys. **F3** ends the program and **F12** returns to the prompt display.

Part 2: Create the display file VNDINQ

__ 4. If you know how to use SDA or Screen Designer to create display files, please use one or the other. Otherwise, use the LPEX editor or SEU to code the DDS for each of the two formats above.

__ 5. Name the **DDS** source member, **VNDINQ** (type **DSPF**), in file AS06*nnn*/QDDSSRC.

__ 6. Create two formats:

__ a. The prompt format

__ b. The detail format

__ 7. Name your formats **PROMPT_FMT** and **DSPLY_FMT**.

__ 8. Include the **user ID**, **system name**, **date**, and **time** in your formats.

__ 9. Use keyword `CA03(03)` to enable the function key for both formats and to support the text **F3 = Exit**.

__ 10. Include **Purchases YTD** and **Balance Owed**, with appropriate editing. If the Balance Owed is greater than 5000.00, display it using reverse image and the color red. This condition should be based on indicator 60 being set by the program.

Note

Currency symbol

Students should use their local currency symbol. No currency symbol is shown in the sample solution to avoid CCSID issues in countries where the currency symbol is not a dollar sign.

__ 11. Use keyword `ERRMSG` ('Invalid vendor number - press **Reset** and re-enter' **nn**) to display an error message on line **24** if the vendor number entered is invalid.

__ 12. The **nn** is your program's error indicator if the vendor number is invalid. Remember to condition the keyword with indicator nn.

__ 13. Consult the Appendix in this *Exercise Guide* for the `Vendor_PF` file and **DICTIONARY** file to review field names and their characteristics.

Part 3: Understanding the program logic

__ 14. In this program, you access the `VENDOR_PF` file using its key, **VNDNBR**, obtaining the value of the key from the prompt format of your display file.

__ 15. Use the program description described previously as a guide for the logic you write to interact with your display file. Note the use of indicator 60 to test the balance. If you reset the indicator in your display file, there is no need to reset it in your program.

Part 4: Create the program VNRINQ

__ 16. Create a new member, **VNRINQ**, in file AS06*nnn*/`QRPGLESRC`.

__ 17. Use your sample inquiry program in your notebook to help guide you as you code this inquiry program.

__ 18. Refer to your display file for indicator usage.

__ 19. Compile the program.

Part 5: Test your program VNRINQ

__ 20. Call **VNRINQ**.

__ 21. Enter a **valid vendor** number (10001-10050). You should see the vendor's information.

__ 22. Enter an **invalid vendor** number. You should see the error message you coded in the `ERRMSG` keyword.

__ 23. Enter a **valid vendor** number (10001-10050) again. Verify the error message is not displayed.

__ 24. Press **F3** to end the program. Your program should end.

End of exercise

Appendix A. Physical and logical files DDS

Field Reference PF: DICTIONARY

```
A*********************************************************************
A** Field Reference PF:  DICTIONARY
A*********************************************************************
A          R REFFMT                      TEXT('Field Reference File')
A*
A** Fields Used in Vendor Mastor File, VENDOR_PF
A*
A            VNDNBR          5  0       TEXT('Vendor Number')
A                                       COLHDG('Vend' 'Num')
A            VNDNAME        25          TEXT('Vendor Name')
A                                       COLHDG('Vendor' 'Name')
A            VNDSTREET      25          TEXT('Vendor Street')
A                                       COLHDG('Vendor Street')
A            VNDCITY        23          TEXT('Vendor City')
A                                       COLHDG('Vendor City')
A            VNDSTATE        2          TEXT('Vendor State')
A                                       COLHDG('Vnd' 'ST')
A            VNDADDR3       25          TEXT('Address Line 3')
A                                       COLHDG('Address Line 3')
A            VNDZIPCODE      5  0       TEXT('Zip Code')
A                                       COLHDG('Zip' 'Code')
A            VNDAREACD       3  0       TEXT('Vendor Area Code')
A                                       COLHDG('Vend' 'Area' 'Code')
A            VNDTELNO        7  0       TEXT('Vendor Telephone Number')
A                                       COLHDG('Vendor' 'Tel' 'No' )
A            VNDDISCPCT      3  3       TEXT('Discount % For Prompt Pymt')
A                                       COLHDG('Disc' 'Per' 'Cent')
A            VNDDUEDAYS      2  0       TEXT('Days Until Payment is Due')
A                                       COLHDG('Terms' 'Days')
A            VNDCLASS        2  0       TEXT('Vendor Class')
A                                       COLHDG('Vnd' 'Cls')
A            VNDACTIVE       1          TEXT('A=Active D=Delete S=Suspend')
A                                       COLHDG('Act' 'Rec' 'CD')
A            VNDSALES       25          TEXT('Vendor Salesperson')
A                                       COLHDG('Vendor' 'Sales' 'Person')
A            VNDDISCMTD      7  2       TEXT('Discount Taken This Month')
A                                       COLHDG('Vend' 'Disc' 'MTD')
A            VNDDISCYTD      9  2       TEXT('Discount Taken This Year')
A                                       COLHDG('Vend' 'Disc' 'YTD')
```

```
A            VNDPRCHMTD     9  2       TEXT('Purchases This Month')
A                                      COLHDG('Vend' 'Purch' 'MTD')
A            VNDPRCHYTD    11  2       TEXT('Purchases This Year')
A                                      COLHDG('Vend' 'Purch' 'YTD')
A            VNDBALANCE     9  2       TEXT('Vendor Balance Owed')
A                                      COLHDG('Vend' 'Balance' 'Owed')
A            VNDSERVRTG     1          TEXT('Vendor Service Rating')
A                                      COLHDG('Vnd' 'Srv' 'Rtg')
A            VNDDLVRTG      1          TEXT('Vendor Delivery Rating')
A                                      COLHDG('Vnd' 'Del' 'Rtg')
A            VNDCOMMENT    25          TEXT('Comments About This Vendor')
A                                      COLHDG('Comments')
A*
A* Fields Used In Item Master File, ITEM_PF
A*
A            ITMNBR         5  0       TEXT('Item Number')
A                                      COLHDG('Item' 'Num')
A            ITMDESCR      25          TEXT('Item Description')
A                                      COLHDG('Item' 'Description')
A            ITMQTYOH       7  0       TEXT('Quantity on Hand')
A                                      COLHDG('Qty' 'on' 'Hand')
A            ITMQTYOO       7  0       TEXT('Quantity on Order')
A                                      COLHDG('Qty' 'on' 'Order')
A            ITMCOST        5  2       TEXT('Item Unit Cost')
A                                      COLHDG('Item' 'Unit' 'Cost')
A            ITMPRICE       5  2       TEXT('Item Unit Price')
A                                      COLHDG('Item' 'Unit' 'Price')
A            ITMVNDCAT#     7          TEXT('Vendor Catalog Number')
A                                      COLHDG('Vendor' 'Catalog' 'Number')
A*
A** Fields Used For Purchase Order Summary File, POSUM_PF
A*
A            PONBR          6  0       TEXT('Purchase Order Number')
A                                      COLHDG('Purch' 'Order' 'Number')
A            POTOTAMT       7  2       TEXT('Purchase Order Amount')
A                                      EDTCDE(3)
A                                      COLHDG('Purch' 'Order' 'Amount')
A            PODATE         8  0       TEXT('PO Date: YYYYMMDD')
A                                      COLHDG('PO' 'Date' 'YYYYMMD')
A            POSTATUS       1          TEXT('O=On Order C=Complete +
A                                      D=Delete')
A                                      COLHDG('PO' 'Sts')
A                                      VALUES(' ' 'O' 'C' 'D')
A*
```



```
A** Fields Used in Purchase Order Line Item File, POLINE_PF
A*
A            POLQTYOO       5  0       TEXT('PO Item Quantity On Order')
A                                      COLHDG('Qty' 'Ord')
A            POLITMCOST     5  2       TEXT('Item Unit Cost')
A                                      COLHDG('Item' 'Unit' 'Cost')
A            POLDATREC      8  0       TEXT('Date Received')
A                                      COLHDG('Date' 'Rec' 'YYYYMMDD')
A            POLQTYREC      5  0       TEXT('Item Quantity Received')
A                                      COLHDG('Qty' 'Rec')
A            POLSTATUS      1          TEXT('Blank=On Order, C=Complete +
A                                      D=Delete I=Incomplete')
A                                      COLHDG('PO' 'Ln' 'Sts')
A                                      VALUES(' ' 'C' 'D' 'I')
A*
A** Fields Used in Accounts Payable Open Invoice File, APINV_PF
A*
A            APINVNBR       8          TEXT('Vendor Invoice Number')
A                                      COLHDG('Vendor' 'Invoice' 'Number')
A            APDATE         8  0       TEXT('Date Order Complete')
A                                      COLHDG('Date' 'Compl' 'YYYYMMDD')
A            APDISCOUNT     5  2       TEXT('Vendor Invoice Discount +
A                                      Available')
A                                      EDTCDE(3)
A                                      COLHDG('Inv' 'Disc' 'Avail')
A            APNETPAID      7  2       TEXT('Net Amount Paid')
A                                      EDTCDE(3)
A                                      COLHDG('Net' 'Amount' 'Paid')
A            APSTATUS       1          TEXT('Blank=No Action  D=Delete +
A                                      T=To Pay P=Paid')
A                                      COLHDG('AP' 'Sts')
A                                      VALUES(' ' 'D' 'T' 'P')
A            APDATEPAID     8  0       TEXT('Date Paid')
A                                      COLHDG('Date' 'Paid' 'YYYYMMDD')
A            APCHECK#       6  0       TEXT('Check Number')
A                                      COLHDG('Check' 'Number')
A            APDUEDATE      8  0       TEXT('Vendor Invoice Due Date +
A                                      YYYYMMDD')
A                                      COLHDG('Due' 'Date' 'YYYYMMDD')
```

Accounts Payable Invoice PF: APINV_PF

```
A*****************************************************************
A*  Accounts Payable Invoice PF:  APINV_PF
A*****************************************************************
A                                     REF(DICTIONARY)
A                                     UNIQUE
A          R APINV_FMT                TEXT('Open Payables Record')
A            PONBR     R
A            VNDNBR    R
A            APINVNBR  R
A            APDATE    R
A            POTOTAMT  R
A            APDISCOUNTR
A            APNETPAID R
A            APSTATUS  R
A            APDATEPAIDR
A            APCHECK#  R
A            APDUEDATE R
A          K PONBR
```

Item Master PF: ITEM_PF

```
A*****************************************************************
A*  Item Master PF:  ITEM_PF
A*****************************************************************
A                                     REF(DICTIONARY)
A                                     UNIQUE
A          R ITEM_FMT                 TEXT('Item Master Record')
A            ITMNBR    R
A            ITMDESCR  R
A            ITMQTYOH  R
A            ITMQTYOO  R
A            ITMCOST   R
A            ITMPRICE  R
A            VNDNBR    R
A            ITMVNDCAT#R
A          K ITMNBR
```

Join LF for delinquency notices: PODLNQ_LF

```
A*%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
A*  Join LF for delinquency notices:  PODLNQ_LF
A*%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
A          R PODLNQ_FMT                JFILE(POSUM_PF POLINE_PF VENDOR_PF)
A          J                           JOIN(1 2)
A                                      JFLD(PONBR PONBR)
A                                      JDUPSEQ(ITMNBR)
A          J                           JOIN(1 3)
A                                      JFLD(VNDNBR VNDNBR)
A*  Fields from  POSUM_PF:
A            PONBR                     JREF(1)
A            VNDNBR                    JREF(1)
A            PODATE
A*  Fields from POLINE_PF
A            ITMNBR
A            POLQTYOO
A            POLITMCOST
A            POLQTYREC
A* Fields from VENDOR_PF:
A            VNDNAME
A            VNDAREACD
A            VNDTELNO
A            VNDSALES
A*
A          K VNDNBR
A          K PONBR
```

PO line item LF: POITEM_LF

```
A*%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
A*  PO line item LF:  POITEM_LF
A*%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
A
A          R POLINE_FMT                TEXT('PO Line Item Record')
A                                      PFILE(POLINE_PF)
A          K ITMNBR
A          O POLSTATUS                 CMP(EQ 'D')
```

PO line item PF: POLINE_PF

```
A*********************************************************************
A*  PO line item PF:  POLINE_PF
A*********************************************************************
A                                      REF(DICTIONARY)
A                                      UNIQUE
A          R POLINE_FMT                TEXT('PO Line Item Record')
A            PONBR     R
A            ITMNBR    R
A            POLQTYOO  R
A            POLITMCOSTR
A            POLDATREC R
A            POLQTYREC R
A            POLSTATUS R
A          K PONBR
A          K ITMNBR
```

PO Open Line Item LF: POOPNLI_LF

```
A*********************************************************************
A*  PO Open Line Item LF:  POOPNLI_LF
A*********************************************************************
A
A          R POLINE_FMT                TEXT('PO Line Item Record')
A                                      PFILE(POLINE_PF)
A          K PONBR
A          K ITMNBR
```

PO Summary PF: POSUM_PF

```
A*********************************************************************
A*   PO Summary PF: POSUM_PF
A*********************************************************************
A                                      REF(DICTIONARY)
A                                      UNIQUE
A          R POSUM_FMT                 TEXT('PO Summary Record')
A            PONBR     R
A            VNDNBR    R
A            POTOTAMT  R
A            PODATE    R
A            POSTATUS  R
A          K PONBR
```

Vendor master PF: VENDOR_PF

```
A*****************************************************************
A*  Vendor master PF:  VENDOR_PF
A*****************************************************************
A                                      REF(DICTIONARY)
A                                      UNIQUE
A          R VENDOR_FMT                TEXT('Vendor Master File Record')
A            VNDNBR    R
A            VNDNAME   R
A            VNDSTREET R
A            VNDCITY   R
A            VNDSTATE  R
A            VNDZIPCODER
A            VNDAREACD R
A            VNDTELNO  R
A            VNDDISCPCTR
A            VNDDUEDAYSR
A            VNDCLASS  R
A            VNDACTIVE R
A            VNDSALES  R
A            VNDDISCMTDR
A            VNDDISCYTDR
A            VNDPRCHMTDR
A            VNDPRCHYTDR
A            VNDBALANCER
A            VNDSERVRTGR
A            VNDDLVRTG R
A            VNDCOMMENTR
A          K VNDNBR
```

Vendors by Name LF: VNDNAM_LF

```
A*****************************************************************
A*  Vendors by Name LF:  VNDNAM_LF
A*****************************************************************
A                                      ALTSEQ(QSYSTRNTBL)
A          R VENDOR_FMT                PFILE(VENDOR_PF)
A          K VNDNAME
```

Appendix B. Exercise solutions

Exercise 1: Coding and compiling RPG IV

```
DMessage          s             30    inz('The sum of 2 plus 2 is')
DSum              s              3  0 inz

 /free
  sum = 2 + 2;
  message = %trimr(Message) + ' ' + %char(sum);
  Dsply message '*REQUESTER';
  *InLr = *on;
 /end-free
```

The field **Sum** was undefined because the D-spec was entered incorrectly and defined a field named '**Sums**'.

Exercise 2: Sequencing RPG IV specifications and compiling

The **CRTBNDRPG** command is used to compile the program.

Exit, NotFound, and PrintIt are the indicators used in the program.

```
 *                       VNRADR01                                       *
 *                  Vendor Address Inquiry                              *
 ************************************************************************
 * This program prompts the user for a vendor number and displays      *
 * the vendor's address information on the screen.                     *
 *                                                                      *
 * The user has options to exit the program and to print a vendor      *
 * record.                                                             *
 *                                                                      *
 * INDICATORS:                                                         *
 *   Exit      - the user requests to exit the program                 *
 *   PrintIt   - the user requests to print vendor address             *
 *   NotFound  -  no vendor found to match the input vendor number     *
 ************************************************************************
H DftActGrp(*yes) ExprOpts(*ResDecPos) DatFmt(*USA)
// Vendor Display Formats
FVndAdr01  CF   E             WorkStn InDDS(WkStnInd)
  // Vendor Data File
FVendor_PF IF   E           K Disk
  // Report Formats
FVnpAdr    O    E             Printer OflInd(PrtOver)
 ********************************************************************
D ToDaysDate      S               D
  // Named indicators used with display file
D WkStnInd        DS
D  Exit                    3      3N
D  NotFound               99     99N
D  PrintIt                10     10N
 ********************************************************************
 /Free


 TodaysDate = %date(*date); // Get date from system
 ExFmt     Prompt_Fmt; // Prompt for vendor number

 DoW Not Exit; // Do the following until user presses F3 (Exit)

     NotFound= *off; // Initialize the record found indicator
     Chain VndNbr Vendor_PF; // Find the vendor record

     If %Found(Vendor_PF);  // If we find a valid vendor record:
        ExFmt Dsply_Fmt;    // Disply the vendor record

        If PrintIt;         // If the user pressed F10,
           Write Vnadd_Fmt; // print the vendor record
        EndIf;

     Else;               // We did not find a record
NotFound = *on;  // Set the record found indicator on
      EndIf;
```

```
            ExFmt Prompt_Fmt; // Prompt for a new vendor number
  EndDo;

  *inLr = *on;  // End the program
 /End-free
```

Exercise 3: Coding a report program

```
FItem_PF   IF   E           K Disk
FItpcost   O    E             Printer Include(Detail)
F                                     Include(Heading)
F                                     OflInd(PrtOver)
**
 /Free
  Write Heading;  // Produce the heading
  // Read first record to get started
  Read      Item_Pf;

  DoW not %eof(Item_PF); // We have at least one record so
                         // enter loop to process remaining
                         // records.

           Write  Detail;
  // Read subsequent records
           Read   Item_Pf;
  EndDo;

  *InLr=*on;
 /End-free
```

Exercise 4: Adding overflow

```
 // File Declarations
FItem_PF   IF   E           K Disk
FItpCost   O    E             Printer Include(Heading)
F                                     Include(Detail)
F                                     OflInd(PrtOver)


  // Headings on first page
 /Free
  Write     Heading;

  // Read first record to get started
  Read Item_Pf;

  Dow not %eof(Item_PF); // We have at least one record so
                         // enter loop to process remaining
                        // records.




                              // Headings on page overflow
         If PrtOver;          // check for overflow
            Write Heading;
            PrtOver = *Off;
         Endif;

  Write  Detail; // Print detail record
  Read Item_Pf;  // read subsequent records
  EndDo;
  *InLR = *On;
  /End-free
```

Exercise 5: Data definition

```
FItem_PF   IF   E           K Disk
FPOPList   O    E             Printer OflInd(PrtOver)

D Low              S               3  0
D Count            S              +2    Like(low)

 /Free
    // Headings on first page
    Write Heading;

    // Read first record of file
    Read Item_PF;

    DoW not %eof(Item_PF);

         // Page overflow?
         If PrtOver;
              Write Heading;
              PrtOver = *Off;
         EndIf;

         // Accumulate totals
         TotQtyAvl = ItmQtyOH + ItmQtyOO;
         // Set indicator for Low Available Quantity
         *in43 = (TotQtyAvl < 15);

         // Accumulate low quantity situations
         If *in43;
              Low = Low + 1;
         EndIf;

         // Calculate value (at cost) of available inventory
         AvlCost = ItmCost * TotQtyAvl;

         // Calulate value (at retail) of available inventory
         AvlPrice = ItmPrice * TotQtyAvl;

         // Accumulate the total number of records processed
         Count = Count + 1;

         // Print the detail format
         Write Detail;
```

```
            // Read second and subsequent records of file
            Read Item_PF;
        EndDo;

        // End of file processing

        // Move the program accumulators to the printer file fields
        LowCountP = Low;
        TotCountP = Count;
        // Print the record format for totals
        Write Footing;
        *InLr = *on;
    /End-Free
```

Exercise 6: Adding arithmetic function

```
 // File Declarations
FItem_PF   IF   E           K Disk
FItpCost   O    E             Printer OflInd(PrtOver)


  // Headings on first page
 /Free
  Write     Heading;

  // Read first record to get started
  Read Item_Pf;

  Dow not %eof(Item_PF); // We have at least one record so
                         // enter loop to process remaining
                        // records.
  // Calculate cost on hand
         ItmCostOH = ItmCost * ItmQtyOH;
  // Calculate total cost on hand
         TotCostOH = TotCostOH + ItmCostOH;

                                // Headings on page overflow
         If PrtOver;            // Check for overflow
            Write Heading;
            PrtOver = *Off;
         Endif;

  Write  Detail; // Print detail record format
  Read Item_Pf;  // Read subsequent records
  EndDo;
  Write  Total;  // Print total record format
  *InLR = *On;
  /End-free
```

Exercise 7: Data manipulation

```
//                       VNRADR03S
// Active Vendor Report
//*************************************************************************
// Print listing of ACTIVE vendors from Vendor_PF file.
//
// INDICATORS:
//   PrtOverFlow - Printer overflow
//   LR - Close files, end program
//
//*************************************************************************
FVendor_PF IF   E           K DISK                          Input database file
FVnpAdr03  O    E             Printer OflInd(PrtOverFlow)   Output printer file

 /FREE
  Write Heading;                                          //Print page 1 heading

  // Read first record
  Read Vendor_Fmt;                                         //Read input record

  Dow not %eof (Vendor_PF);              //While NOT EOF

    // Include selection for only active records

    If VndActive = 'A';                                    //For active rec only

      // Put salesperson first name in smaller field on report
      VndSales1 = %Subst(VndSales:1:
                  %scan(' ':VndSales));

      // Check for overflow

      If  PrtOverFlow;
        Write Heading;
        PrtOverFlow = *off;
      EndIf;

      Write Detail;                                       //Print detail record

      // Increment counter of active records processed thus far

      Count = Count + 1;

      // End conditional logic for active records only
    EndIf;

     // Read second and remaining records
    Read Vendor_Fmt;                                        //Read input record
  EndDo;

  // Total processing
    // Check for overflow
    If PrtOverFlow;
```

```
        Write Heading;
      EndIf;
      //
      Write Total;                                          //Print record count
      *inLr = *on;

/END-FREE
```

Exercise 8: Printing from an RPG program

```
A*%%TS  DD  20020131  073734  QUSER       REL-V5.1    iSeries WDT
A*%%PR 1066132I
A*********************************************************************
A*                          VNPADR04S
A*                    VENDOR Balance Due Report                      *
A*********************************************************************
A* THIS PRINTER FILE FORMATS THE VENDOR ADDRESS DATA.                 *
A* INDICATORS:  NONE                                                 *
A*********************************************************************
A                                     REF(VENDOR_PF)
A          R HEADING                  TEXT('VENDOR_LISTING')
A                                     SKIPB(1)
A                                    1'PAGE'
A                                   +1PAGNBR
A                                     EDTCDE(Z)
A                                   35'Balance Due Vendor Report'
A                                   90DATE
A                                     EDTCDE(Y)
A                                    1'    Vendor Name'
A                                     SPACEB(2)
A                                   17'   and'
A                                   28'Number'
A                                   36'Vendor City'
A                                   58'State'
A                                   66'YTD Purchases'
A                                   81'Current Bal. Due'
A                                     SPACEA(1)
A          R DETAIL
A                                     SPACEB(1)
A            VNDNAME    R        O    1
A            VNDNBR     R        O   28EDTCDE(Z)
A            VNDCITY    R        O   +2
A            VNDSTATE   R        O   +2
A            VNDPRCHYTDR         O   64EDTCDE(J)
A            VNDBALANCER         O   +4SPACEA(2)
A                                     EDTCDE(J)
A          R TOTAL
A                                     SPACEB(1)
A                                    1'Number of Active Vendors'
A            COUNT          9  0O   +8SPACEA(2)
A                                     EDTCDE(1)
A                                    1'Total YTD Purchase Value'
A            TOTPURCH   R   +2   O   27SPACEA(2)
A                                     EDTCDE(J)
A                                     REFFLD(VNDPRCHYTD)
A                                    1'Total Current Amount Due'
A            TOTBAL     R   +2   O   30SPACEA(14)
A                                     EDTCDE(J)
A                                     REFFLD(VNDBALANCE)

      //*********************************************************************
      //                       VNRADR03S
```

```
// Active Vendor Report
//*************************************************************************
// Print listing of ACTIVE vendors from Vendor_PF file.
//
// INDICATORS:
//   PrtOverFlow - Printer overflow
//   LR - Close files, end program
//
//*************************************************************************
FVndNam_LF IF   E           K DISK                          Input database file
FVnpAdr04S O    E             Printer OflInd(PrtOverFlow)   Output printer file

/FREE
 Write Heading;                                          //Print page 1 heading

 // Read first record
 Read Vendor_Fmt;                                          //Read input record

 Dow not %eof (VndNam_LF);              //While NOT EOF

   // Include selection for only active records

   If VndActive = 'A';                                   //For active rec only

     // Accumulate totals
     TotPurch = TotPurch + VndPrchYTD;
     TotBal   = TotBal + VndBalance;

     // Check for overflow

     If  PrtOverFlow;
       Write Heading;
       PrtOverflow = *off;
     EndIf;

     Write Detail;                                        //Print detail record

     // Increment counter of active records processed thus far

     Count = Count + 1;

     // End conditional logic for active records only
   EndIf;

    // Read second and remaining records
   Read Vendor_Fmt;                                        //Read input record
 EndDo;

 // Total processing
   // Check for overflow
   If PrtOverFlow;
     Write Heading;
   EndIf;
   //
   Write Total;                                          //Print record count
   *inLr = *on;
```

```
 //
/END-FREE
```

Exercise 9: Debugging an RPG IV program

```
 //                        VNRADR05S
 // Active Vendor Report
 //*************************************************************************
 // Print listing of ACTIVE vendors from Vendor_PF file.
 //
 // INDICATORS:
 //   LR - Last record
 //*************************************************************************
HDATFMT(*USA)
 //*************************************************************************
FVendor_PF IF   E           K DISK
 // Printer file is program described using system supplied Qprint PRTF
FQprint    O    F  132        Printer OflInd(*InOF)

 //*************************************************************************
DToDaysDate       S               D
DTime             S              6  0
DCount            S              3  0
 // Add alpha field to contain all of salesperson name that fits report
DVndSales16       S             16
 //*************************************************************************
 //
 /FREE
  // Obtain job date and current time for report heading
  Eval TodaysDate = %date(*date);
  Except ExcVndHdr;                                        //Print page 1 heading
  // Using Except opcode to print printer formats defined in O-Specifications
  Read Vendor_Fmt;
  //
  DoW not %Eof(Vendor_PF);                                   //Process all records
   //                                                              until End of File
    // Include selection for only active records
    If VndActive = 'A';
      // Move lefmost characters of salesperson into a field that fits report
      VndSales16 = %Subst(VndSales:1:16);

      Except ExcVndDtl;                                    //Print detail record
      // Keep a count of the records processed
      Count=Count+1;

    EndIf;
    // Read second and subsequent records
    Read Vendor_Fmt;
  EndDo;

  Except ExcVndTot;                                        //Print record count
  *InLr = *on;
  //*************************************************************************
 /END-FREE
  // Printer output is defined in program written output specifications. You
  // will encounter code like this when maintaining programs that were coded
  // using RPG/400 (RPG III). You should use the RPG IV reference manuals for
```

```
  // further information.
OQPrint    E            ExcVndHdr      3 06
O         OR    OF
 //                      Page Heading
O                                          'Page'
O                      Page             +   2
O                                         55 'Active Vendors'
O                      TodaysDate          122
O                      Time                132 '  :  :  '

O          E           ExcVndHdr      1
O         OR    OF
 //                      Column Heading Line # 1
O                                        114 'Vend'
O                                        122 'Vendor'

O          E           ExcVndHdr      1
O         OR    OF
 //                      Column Heading Line # 2
O                                          5 'Vend'
O                                         89 'VND'
O                                         98 'Zip'
O                                        104 'Area'
O                                        114 'Tel'
O                                        121 'Sales'

O          E           ExcVndHdr      2
O         OR    OF
 //                      Column Heading Line # 3
O                                          5 'Num'
O                                         32 'Vendor Name              '
O                                         59 'Vendor Street            '
O                                         84 'Vendor City           '
O                                         91 'State'
O                                         98 'Code'
O                                        104 'Code'
O                                        114 'No'
O                                        122 'Person'

 //*************************************************************************
O          EF          ExcVndDtl      2
 //                      Vendor Detail Information
O                      VndNbr         Z     5
O                      VndName              32
O                      VndStreet            59
O                      VndCity              84
O                      VndState             88
O                      VndZipcode           98
O                      VndAreaCd           104
O                      VndTelNo            114 '0   -    '
O                      VndSales16          132

O          EF          ExcVndTot   2
O                                          32 'Number of active vendors:'
O                      Count          1    45
O                                          47 '*'
```

Exercise 10: Coding subroutines

Part A:

```
FPOPMNT02  O    E             Printer OfLind(OverFlow)

FPOTrans_PFIF   E             Disk
/Free
  Read POTrans_PF;
  Write Heading;
  Dow Not %Eof(POTrans_PF);
    Select;
    When TrnCode = 'A';
      Exsr AddSubr;       // Process ADD transaction
    When TrnCode = 'C';
      Exsr ChgSubr;       // Process CHANGE transaction
    When TrnCode = 'D';
      Exsr DltSubr;       // Process DELETE transaction

    Other;
      Exsr ErrSubr;       // Process INVALID transaction
    EndSL;
    TotCnt = TotCnt + 1;
    Read POTrans_PF;
  Enddo;
  If OverfLow;
    Write Heading;
    OverFlow = *Off;
  Endif;
  Write Total;
  *InLR = *On;

   //-----------------Subroutines---------------------------------
  BegSR AddSubr;
    AddCnt = AddCnt + 1;

  EndSR;
  BegSR ChgSubr;
    ChgCnt = ChgCnt + 1;
  EndSR;
  BegSR DltSubr;
    DelCnt = DelCnt + 1;
  EndSR;
  BegSR ErrSubr;
    ErrCnt = ErrCnt + 1;
  EndSR;
 /End-Free
```

Part B:

```
FPOPMNT02  O    E             Printer OfLind(OverFlow)
F                                     IndDS(Indicators)
FPOTrans_PFIF   E             Disk
D Indicators      DS
D   Invalid                       N   Overlay(Indicators:15)
 /Free
  Read POTrans_PF;
  Write Heading;
  Dow Not %Eof(POTrans_PF);
    Select;
    When TrnCode = 'A';
      Exsr AddSubr;       // Process ADD transaction
    When TrnCode = 'C';
      Exsr ChgSubr;       // Process CHANGE transaction
    When TrnCode = 'D';
      Exsr DltSubr;       // Process DELETE transaction
    Other;
      Exsr ErrSubr;       // Process INVALID transaction
    EndSL;
    TotCnt = TotCnt + 1;
    Read POTrans_PF;
  Enddo;
  If OverfLow;
    Write Heading;
    OverFlow = *Off;
  Endif;
  Write Total;
  *InLR = *On;

   //-----------------Subroutines---------------------------------
  BegSR AddSubr;
    AddCnt = AddCnt + 1;
    Exsr PrtDetail;
  EndSR;
  BegSR ChgSubr;
    ChgCnt = ChgCnt + 1;
    Exsr PrtDetail;
  EndSR;
  BegSR DltSubr;
    DelCnt = DelCnt + 1;
    Exsr PrtDetail;
  EndSR;
  BegSR ErrSubr;
    ErrCnt = ErrCnt + 1;
    Invalid = *On;
    Exsr PrtDetail;
    Invalid = *Off;
  EndSR;
  BegSR PrtDetail;
```

```
    If OverFlow;
      Write Heading;
      OverFlow = *Off;
    Endif;
    Write TrnDetail;
  EndSR;
 /End-Free
```

Exercise 11: Maintaining database files

```
 //*********************************************************************
 //                          PORMNT02                                  *
 //                  PO Maintenance - Open Line Items                  *
 //*********************************************************************
 //                                                                    *
 // This program allows changes to the Open Line Items:               *
 //        Additions                                                   *
 //        Deletions                                                   *
 //        Changes                                                     *
 //                                                                    *
 // INDICATORS:                                                        *
 //   10 - Record not found                                            *
 //   11 - Record exists (duplicate add)                               *
 //   12 - Record added                                                *
 //   13 - Record deleted                                              *
 //   14 - Record changed                                              *
 //   15 - Invalid Transaction                                         *
 //*********************************************************************
 // PO Transaction File
FPOTRANS_PFIF   E             DISK
 // Open Line Items File
FPOOPNLI_LFUF A E           K DISK
 // Maintenance Printer File
FPOPMNT02  O    E             PRINTER OFLIND(OverFlow) INDDS(Indicators)

D Indicators      DS
D  RecNotFound            10     10N
D  RecExists              11     11N
D  RecAdded               12     12N
D  RecDeleted             13     13N
D  RecChanged             14     14N
D  InvalidTrans           15     15N


 // Build the Line Item Key
C     LItmKey       KLIST
C                   KFLD                    POTPONBR
C                   KFLD                    POTITMNBR

 // Write headings on first page of report
 /FREE
  Write Heading;

  // Read first transaction record
  Read POTrans_PF;

  DoW not %eof(POTrans_PF);

    // Reset  Indicators
    RecNotFound = *off;
    RecExists = *off;
    RecAdded = *off;
```

```
    RecDeleted = *off;
    RecChanged = *off;
    InvalidTrans = *off;

    // Process transactions
    Select;

    When Trncode = 'A';
      Exsr AddSr;

    When Trncode = 'D';
      Exsr DelSr;

    When Trncode = 'C';
      Exsr ChgSr;

    Other;
      Exsr InvTrSr;

    EndSl;

    // Read next transaction record
    Read POTrans_PF;

  EndDo;                                                    //END OF DO

  // EOJ Processing
  Exsr PrtTotals;
  *inlr = *on;

  // Additions
  BEGSR ADDSR;
    SetLL LItmKey POOPNLI_LF; // Does record already exist?

    If not %equal(POOPNLI_LF);
      ponbr = potponbr;
      ITMNBR = potitmnbr;
      POLQTYOO = POTQTYOO;
      POLITMCOST = POTITMCOST;
      POLDATREC = POTDATREC;
      POLQTYREC = POTQTYREC;
      POLSTATUS = POTSTATUS;
      // If no record found (OK), add and set RecAdded
      WRITE POLINE_FMT;
      RecAdded = *ON;
      Addcnt = Addcnt + 1;

      // If record does not exist, set RecExists
    Else;
      RecExists = *on;
      ErrCnt = ErrCnt + 1;

    EndIf;                                                  //END OF IF

    // Print transaction
    Exsr PrtDetail;
```

```
ENDSR;

// Deletions
BEGSR DelSR;

  // Delete the record
  // If no record found (Error), set RecNotFound
  // If record exists (OK), delete and set RecDeleted
  DELETE LItmKey POOPNLI_LF;
  If %found(POOPNLI_LF);
    RecDeleted = *ON;
    Delcnt = Delcnt + 1;

  Else;
    RecnotFound = *on;
    Errcnt = Errcnt + 1;
  EndIf;

  Exsr PrtDetail;
ENDSR;

// Changes
BEGSR ChgSr;
  // Check for record
  CHAIN LItmKey POOPNLI_LF;

  If %found(POOPNLI_LF);
    // Print Original fields
    Exsr PrtPOOrig;
    // Update quantity
    ponbr = potponbr;
    ITMNBR = potitmnbr;
    POLQTYOO = POTQTYOO;
    POLITMCOST = POTITMCOST;
    POLDATREC = POTDATREC;
    POLQTYREC = POTQTYREC;
    POLSTATUS = POTSTATUS;
    // Change the record
    UPDATE POLINE_FMT;
    // If record exists, set RecChanged
    RecChanged = *ON;
    Chgcnt = Chgcnt + 1;
  Else;
    RecnotFound = *on;  // Set if no record found
    Errcnt = Errcnt + 1;
  EndIf;

  // Print transaction
  Exsr PrtDetail;

ENDSR;

// Invalid Transaction
BegSr InvTrSr;
  InvalidTrans = *on;
```

```
      Errcnt = Errcnt + 1;
      Exsr PrtDetail;
      InvalidTrans = *off;
    EndSr;

    // Print Totals Subroutine
    Begsr PrtTotals;
      TotCnt = ChgCnt + DelCnt + ErrCnt + Addcnt;
      ExSr CheckOV;
      Write Total;
    EndSr;

    //  Print original record fields, transaction and new fields
    BegSr PrtPOOrig;
      ExSr CheckOV;
      Write POOrig;
    EndSr;

    //  Print transaction
    BegSr PrtDetail;
      ExSr CheckOV;
      Write TrnDetail;
    EndSr;

    // Check for Overflow
    BegSr CheckOV;
      If OverFlow;
        Write Heading;
        OverFlow = *off;
      EndIf;
    EndSr;

   /END-FREE
```

To change **KLIST**/**KFLD** to parameters:

1. Delete or comment out the **KLIST**/**KFLD** lines of code at the beginning of calculations.
2. Modify each opcode that uses the **LitmKey** to use a parameter string of (`PotPONbr: PotItmNbr`). For example:

   `SetLL (PotPONbr : POItmNbr) POOPNLI_LF`

Exercise 12: Coding an inquiry program

```
A*%%***************************************************************
A*                            VNDINQS                               *
A*              Inquiry by Vendor Number Display File               *
A*%%***************************************************************
A* THIS DISPLAY FILE CONTAINS THESE FORMATS:                        *
A*                                                                  *
A*     PROMPT_FMT - Prompts for Vendor Number                       *
A*     DSPLY_FMT  - Displays a vendor record                        *
A*                                                                  *
A* INDICATORS:                                                      *
A*   03 - User requests to exit the program                         *
A*   12 - Cancel (Return to prompt screen)                          *
A*   60 - Balance due > 5000.00                                     *
A*   96 - Invalid vendor number                                     *
A*%%***************************************************************
A                                      REF(*LIBL/VENDOR_PF)
A                                      INDARA
A                                      CA03(03 'End Program')
A          R PROMPT_FMT
A                                    1  2USER
A                                    1 30'Vendor Inquiry'
A                                       COLOR(WHT)
A                                    1 71SYSNAME
A                                    2 61DATE
A                                       EDTCDE(Y)
A                                    2 71TIME
A                                    3  3'Vendor number. . . . :'
A            VNDNBR_INQR     D  I    3 28COLOR(WHT)
A                                       REFFLD(VNDNBR DICTIONARY)
A  96                                   ERRMSG('Invalid vendor number - pre-
A                                       ss reset and re-enter' 96)
A                                   22  3'Please press enter to continue'
A                                   23  4'F3 = Exit'
A                                       COLOR(BLU)
A          R DSPLY_FMT
A                                       CA12(12 'Return to previous display-
A                                       Display')
A                                    1  2USER
A                                    1 30'Vendor Inquiry'
A                                       COLOR(WHT)
A                                    1 71SYSNAME
A                                    2 61DATE
```

```
A                                       EDTCDE(Y)
A                                    2 71TIME
A                                    7  3'Vendor number . .:'
A            VNDNBR    R        O  7 24
A                                    8  3'Name . . . . . :'
A                                    9  3'Address  . . . . :'
A            VNDNAME   R        O  8 24
A            VNDSTREET R        O  9 24
A            VNDCITY   R        O 10 24
A            VNDSTATE  R        O 10 49
A            VNDZIPCODER        O 10 53
A                                   11  3'Telephone. . . . :'
A            VNDAREACD R        O 11 26
A                                   11 24'('
A                                   11 30')'
A            VNDTELNO  R        O 11 33
A                                       EDTWRD('0  -    ')
A                                   12  3'Sales Person . . :'
A            VNDSALES  R        O 12 24
A                                   13  3'Purchases YTD  . :'
A            VNDPRCHYTDR            13 24EDTCDE(J)
A                                   14  3'Balance Owed . . :'
A            VNDBALANCER            14 26EDTCDE(J)
A  60                                DSPATR(HI)
A  60                                COLOR(RED)
A                                   23  4'F3 = Exit  F12 = Return to previou-
A                                        s Display'
A                                        COLOR(BLU
```

```
     // Vendor master File
     FVendor_PF IF   E           K Disk
      // Display File
     FVndinqs   CF   E             Workstn IndDS(WkIndicators)
      // Indicator Data Structure
     D WkIndicators    DS
     D Exit                    3      3N
     D Cancel                 12     12N
     D HighBalance            60     60N
     D NotFound               96     96N
      /FREE

       Exfmt Prompt_fmt; // Display Prompt_Fmt

       Dow NOT Exit;  // Continue process until user presses F3
```

```
    Chain Vndnbr_inq Vendor_PF;     // Read record; valid key?

    If %found(Vendor_PF);

        // Record found; display the Dsply_Fmt
      DoW NOT Cancel;
        // Check whether balance owed is greater than 5000.00
        HighBalance = VndBalance > 5000.00;
        // Display details
        Exfmt Dsply_fmt;

        IF Exit;   // F3 pressed

          *InLR = *ON;
          Return;  // exit program

        Endif;

      EndDo;

    Else;
      NotFound = *on;
    endif;

    // No Item record found or F12 - display prompt
    Cancel = *OFF; // Reset indicator
    Exfmt Prompt_fmt; // Redisplay Prompt format

  enddo;
  *InLR = *ON;
  //
 /END-FREE
```

Appendix C. Sample legacy programs

Included in this appendix are programs written before free-format calculations were available. The purpose of providing these programs is to enable you to familiarize yourself with code that you will encounter on the job.

Many of these programs are similar to the programs that you coded in the exercises.

Coding and Compiling RPG IV

```
DMessage          S              30    Inz('The sum of 2 plus 2 is')
DSum              S               3  0 INZ
C                   Eval      Sum = 2 + 2
C                   Eval      Message = %trimr(Message) + ' ' +
C                                       %char(sum)
C     message       dsply     '*REQUESTER'
C                   Eval      *inLr = *on
C                   return
```

Sequencing RPG IV Specifications and Compiling

```
     ****************************************************************
     *                          VNDADR01                            *
     *                    Vendor Address Inquiry                    *
     ****************************************************************
     * This display file contains two record formats:               *
     *     PROMPT_FMT - prompts the user for a vendor number        *
     *     DSPLY_FMT  - displays vendor address                     *
     *                                                              *
     * INDICATORS:                                                  *
     *   03 - user requests to exit the program                     *
     *   10 - user requests to print vendor address                 *
     *   99 - invalid vendor number                                 *
     ****************************************************************
     *                      CHANGE LOG                              *
     * DATE        PROGRAMMER      DESCRIPTION                      *
     * ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ *
     *01/22/96     R.P.  Giv        New File                        *
     ****************************************************************
A                                       DSPSIZ(24 80 *DS3)
A                                       REF(*LIBL/VENDOR_PF)
A          R PROMPT_FMT
A                                       CA03(03)
A                                     1  2USER
A                                     1 71SYSNAME

A                                     1 30'Vendor Address Inquiry'
A                                       DSPATR(HI)
A                                       COLOR(WHT)

A                                     2 61DATE
A                                       EDTCDE(Y)
A                                     2 71TIME

A                                     8 24'Enter Vendor Number .  .'
A            VNDNBR    R        B  8 50CHECK(RZ)
A  99                                 9 30'Invalid Vendor Number'
A                                       COLOR(WHT)

A                                    12 29'Press Enter to Continue'
A                                       COLOR(BLU)

A                                    23  4'F3 = Exit'
A                                       COLOR(BLU)

A          R DSPLY_FMT
A                                       CA10(10 'Print Vendor Record')
A                                     1  2USER
A                                     1 30'Vendor Address Inquiry'
A                                       DSPATR(HI)
A                                       COLOR(WHT)
A                                     1 71SYSNAME
```

```
A                                    2 61DATE
A                                      EDTCDE(Y)
A                                    2 71TIME

A                                    8 26'Vendor Number .  .  :'
A            VNDNBR    R         O  8 48COLOR(WHT)

A            VNDNAME   R         O 10 38COLOR(GRN)
A                                   10 19'Name .  .  .  .  .  :'

A            VNDSTREET R         O 11 38COLOR(GRN)
A                                   11 19'Street .  .  .  .  :'

A                                   12 19'City/State/Zip :'
A            VNDCITY   R         O 12 38COLOR(GRN)
A            VNDSTATE  R         O 12 65COLOR(GRN)
A            VNDZIPCODER         O 12 70EDTCDE(Z)
A                                      COLOR(GRN)
A                                   14 29'Press Enter to Continue'
A                                      COLOR(BLU)

A                                   23  2'F10 = Print Vendor Address'
A                                      COLOR(BLU)


 ****************************************************************
 *                          VNPADR                              *
 *                    Vendor Address Report                     *
 ****************************************************************
 * This printer file formats the vendor address data.           *
 * INDICATORS:  NONE                                            *
 ****************************************************************
 *                     CHANGE LOG                               *
 * DATE        PROGRAMMER      DESCRIPTION                      *
 * ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ *
 * 1/22/96     SSSMITH         New file.                        *
 ****************************************************************
A                                      REF(VENDOR_PF)
A          R VNADD_FMT
A                                      SKIPB(001)
A                                      TEXT('VENDOR ADDRESS FORMAT')
A                                    30'VENDOR ADDRESS'
A                                      SPACEA(1)
A                                    80DATE EDTCDE(Y)
A                                    90TIME SPACEA(1)
A
A                                     2'VENDOR #:'
A            VNDNBR    R            14
A            VNDNAME   R            30SPACEA(1)

A            VNDSTREET R            30SPACEA(1)

A            VNDCITY   R            30
A            VNDSTATE  R            55
A            VNDZIPCODER            60
A
```

```
 ****************************************************************************
 *                        VNRADR01S                                         *
 *                   Vendor Address Inquiry                                 *
 ****************************************************************************
 * This program prompts the user for a vendor number and then displays*
 * the vendor's address information on the screen.                          *
 *                                                                          *
 * The user has options to exit the program and to print a report.          *
*                                                                           *
 * INDICATORS:                                                              *
 *   03 - EXIT - the user requests to exit the program                      *
 *   10 - PRINT- the user requests to print vendor address                  *
 *   99 - ERROR - no vendor found to match the input vendor number          *
 ****************************************************************************
 *                                                                          *
 * DATE        PROGRAMMER      DESCRIPTION                                  *
 * ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ *
 * 5/30/00     R.P. Giv        New Program                                  *
 *                                                                          *
 ****************************************************************************
H Indent('| ')
 ****************************************************************************
 *                   Vendor Display Formats
FVndAdr01  CF   E             WorkStn
 *                   Vendor Data File
FVendor_PF IF   E           K Disk
 *                   Report Formats
FVnpAdr    O    E             Printer
 ****************************************************************************
DToDaysDate       s               D
 ****************************************************************************
C                   Move      Udate         TodaysDate
C                   ExFmt     Prompt_Fmt

C                   DoW       Not *in03

C                   Eval      *In99 = *OFF
C     VndNbr        Chain     Vendor_PF
 **
C                   If        %Found(Vendor_PF)
C                   ExFmt     Dsply_Fmt
 **
C                   If        *in10
C                   Write     Vnadd_Fmt
C                   EndIf
C
C                   Else
C                   Eval      *In99 = *ON
C                   EndIf
 **
C                   ExFmt     Prompt_Fmt
C                   EndDo
 **
C                   Eval      *inLr = *on
```

Coding a Report Program

```
 ****************************************************************************
 *                            ITRCOSTS                                      *
 *                  LIST OF INVENTORY ITEMS                                 *
 *                                                                          *
 * This program lists all the items in the ITEM_PF file.                    *
 *                                                                          *
 * INPUT FILE:  ITEM_PF, externally described                               *
 *              Format:   ITEM_FMT                                          *
 *                                                                          *
 * PRINTER FILE:  ITPCOST, externally described.                            *
 *                Formats:  HEADING, DETAIL, TOTAL                          *
 *                                                                          *
 * INDICATORS:                                                              *
 *   73 - Printer overflow                                                  *
 *   LR - Close files, end program                                          *
 ****************************************************************************
FItem_PF   IF   E           K Disk
FItpcost   O    E             Printer Oflind(*In73)
F                                   Include(Detail)
 **
C                   Read      Item_Pf                                LR
C                   If        not *InLR
C                   Write     Detail
C                   Endif
```

Adding Overflow

```
 ***************************************************************************
 *                           ITRCOSTOFS                                    *
 *                  Cost of Inventory On Hand Report                       *
 ***************************************************************************
 *                                                                         *
 * This program lists all the items in the ITEM_PF file. It prints         *
 * headings on the first page and whenever overflow is encountered.  *
 *                                                                         *
 * INPUT FILE:  ITEM_PF, externally described                              *
 *              Format:   ITEM_FMT                                         *
 *                                                                         *
 * PRINTER FILE:  ITPCOST, externally described.                           *
 *                Formats:  HEADING, DETAIL, TOTAL                         *
 *                                                                         *
 * INDICATORS:                                                             *
 *   30 - first page processing complete                                   *
 *   73 - Printer overflow                                                 *
 *   LR - Close files, end program                                         *
 ***************************************************************************
FItem_Pf   IF   E             Disk
FItpcost   O    E             Printer Oflind(*IN73)
F                                     Ignore(Total)
 ** Write Headings on first page
C                   If        Not *in30
C                   Write     Heading
 ** Headings printed; bypass them next cycle
C                   Eval      *In30 = *on
C                   EndIf
 **
C                   Read      Item_PF                                LR
C                   If        not *InLR
 ** Check for overflow
C                   If        *in73
C                   Write     Heading
C                   Eval      *in73 = *off
C                   EndIf
 ** Write detail line on report
C                   Write     Detail
C                   Endif
```

Alternate Solution for Adding Overflow

```
 ** File Declarations
FItem_PF   IF   E           K Disk
FItpCost   O    E             Printer OfLind(*In73)
F                                     Include(Detail)
F                                     Include(Heading)

D FirstPage       S               N   Inz(*On)

 ** Headings on first page
C                   If        FirstPage
C                   Write     Heading
C                   Eval      FirstPage = *Off
C                   Endif

 ** Read next record
C                   Read      Item_PF

C                   If        %Eof(Item_PF)
C                   Eval      *InLR = *ON

 ** Print if not yet EOF
C                   Else

 ** Headings on page overflow
C                   If        *In73
C                   Write     Heading
C                   Eval      *In73 = *Off
C                   Endif

C                   Write     Detail
C                   Endif
```

Data Definition

```
A                                      REF(ITEM_PF)
 ** Page heading
A          R HEADING                    SKIPB(6)  SPACEA(1)
A                                      5DATE  EDTCDE(Y)
A                                     30'Item Master Listing'
A                                     65'Page:'
A                                     +1PAGNBR  EDTCDE(Z)
A                                      5'Item No'  SPACEB(2)
A                                     13'Description'
A                                     39'Qty On Hand'
A                                     52'Qty On Order'
A                                     66'Tot Avail'
A                                     78'Avail Cost'
A                                     90'Avail List'
 ** Item detail
A          R DETAIL                     SPACEB(1)
A            ITMNBR    R              5
A            ITMDESCR  R             13
A            ITMQTYOH  R             41EDTCDE(J)
A            ITMQTYOO  R             55EDTCDE(J)
A            TOTQTYAVL      8  0     65EDTCDE(J)
A  43                                 76'Low'
A            AVLCOST   R   +2        80REFFLD(ITMCOST) EDTCDE(J)
A            AVLPRICE  R   +2        90REFFLD(ITMPRICE) EDTCDE(J)
 ** Report ending
A          R FOOTING                    SPACEB(2)
A                                     25'Low Stock Count'
A            LOWCOUNTP      5  0     46EDTCDE(J) SPACEA(1)
A                                     25'Total Stock Count'
A            TOTCOUNTP      8  0     42EDTCDE(J) SPACEA(1)
A
A                                     30'*** End of Listing ***'
```

```
FItem_PF   IF   E           K Disk
FPOPList   O    E             Printer Oflind(*In88)

D Low             S              3  0
D Count           S             +2    Like(Low)

C                   If        NOT *In99
 ** Headings on first page
C                   Write     Heading
C                   Eval      *In99 = *ON
C                   Endif

C                   Read      Item_PF                                LR

C                   If        *In88
 ** Page overflow
C                   Write     Heading
```

```
C                   Eval      *In88 = *Off
C                   Endif


C                   If        *InLR
 ** End of file reached
C                   Eval      LowCountP = Low
C                   Eval      TotCountP = Count
C                   Write     Footing
 ** Not EOF so process record
C                   Else

C                   Eval      TotQtyAvl = ItmQtyOH + ItmQtyOO
C                   Eval      *In43 = (TotQtyAvl < 15)
 ** Accumulate low quantity situations
C                   If        *in43
C                   Eval      Low = Low + 1
C                   EndIf
 **
C                   Eval      AvlCost = ItmCost * TotQtyAvl
C                   Eval      AvlPrice = ItmPrice * TotQtyAvl
C                   Eval      Count = Count + 1
C
C                   Write     Detail

C                   Endif
```

Adding Arithmetic Function

```
 ****************************************************************************
 *                            ITRCOSTADS                                    *
 *                  Cost of Inventory On Hand Report                        *
 ****************************************************************************
 *                                                                          *
 * This program lists all the items in the ITEM_PF file. It prints          *
 * headings on the first page and whenever overflow is encountered.  *
 *                                                                          *
 * INPUT FILE:  ITEM_PF, externally described                               *
 *              Format:   ITEM_FMT                                          *
 *                                                                          *
 * PRINTER FILE:  ITPCOST, externally described.                            *
 *                Formats:  HEADING, DETAIL, TOTAL                          *
 *                                                                          *
 * INDICATORS:                                                              *
 *   30 - first page processing complete                                    *
 *   73 - Printer overflow                                                  *
 *   LR - Close files, end program                                          *
 ****************************************************************************
 *                      CHANGE LOG                                          *
 * DATE        PROGRAMMER      DESCRIPTION                                  *
 * ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ *
 * 3/25/96     N. D. Cator     New Program                                  *
 *                                                                          *
 ****************************************************************************
FItem_Pf   IF   E             Disk
FItpcost   O    E             Printer Oflind(*IN73)
 ** Write Headings on first page
C                   If        Not *in30
C                   Write     Heading
 ** Headings printed; bypass them next cycle
C                   Eval      *In30 = *on
C                   EndIf
 **
C                   Read      Item_PF                                LR
C                   If        not *InLR
 ** Process each record as long as there are records (*inLR = EOF)
C                   Eval      ItmCostOH = ItmCost * ItmQtyOH
C                   Eval      TotCostOH = TotCostOH + ItmCostOH
 ** Check for overflow
C                   If        *in73
C                   Write     Heading
C                   Eval      *in73 = *off
C                   EndIf
 ** Write detail line on report
C                   Write     Detail
C                   EndIf
 ** End of file processing
C                   If        *inLR
 ** Check for overflow
C                   IF        *in73
C                   WRITE     HEADING
```

```
C                   Eval      *in73 = *off
C                   EndIf
 ** Print totals
C                   Write     TOTAL
C                   EndIf
```

Data Manipulation

```
 ****************************************************************** ***
 *                       VNRADR03S
 * Active Vendor Report
 ******************************************************************
 * Print listing of ACTIVE vendors from Vendor_PF file.
 *
 * INDICATORS:
 *   30 - first page processing complete                            *
 *   88 - Printer overflow                                          *
 *   LR - Close files, end program                                  *
 *
 ******************************************************************
 *
 * DATE        PROGRAMMER      DESCRIPTION
 * ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 * 5/17/2000   R. P. Gee       New Program
 *
 ******************************************************************
FVendor_PF IF   E           K DISK
FVnpAdr03  O    E             Printer OflInd(*In88)
 ******************************************************************
DVndSales1        S              16
 ******************************************************************
C                   If        Not *in30
C                   Write     Heading
 ** Headings printed; bypass them next cycle
C                   Eval      *In30 = *on
C                   EndIf
 **
C                   Read      Vendor_Fmt                            LR
C                   If        not *InLR
 ** Include selection for only active records
C                   If        VndActive = 'A'
 ** Move VENDOR_PF fields to VNPADR03 PRTF fields
C                   Eval      VENDNAME   = VndName
C                   Eval      VENDSTREET = VndStreet
C                   Eval      VENDCITY   = VndCity
C                   Eval      VENDSTATE  = VndState
C                   Eval      VENDZIPCD  = VndZipCode
C                   Eval      VENDAREACD = VndAreaCd
C                   Eval      VENDPHONE  = VndTelNo
 ** Put salesperson first name in smaller field on report
C                   Eval      VndSales1 = %Subst(VndSales:1:
C                             %scan(' ':VndSales))
 ** Check for overflow
C                   If        *in88
C                   Write     Heading
C                   Eval      *in88 = *off
C                   EndIf
 **
C                   Write     Detail
 ** Increment counter of active records processed thus far
```

```
C                   Eval      Count = Count + 1
 ** End conditional logic for active records only
C                   EndIF
 ** EndIF for processing of records in file
C                   EndIf
 ** Total processing
C                   If        *inLR
 ** Check for overflow
C                   If        *in88
C                   Write     Heading
C                   Eval      *in88 = *off
C                   EndIf
 **
C                   Write     Total
C                   EndIf
```

Printing from an RPG IV Program

This is a different program than the one written in this class.

```
A                                      REF(DICTIONARY)

 ** Heading Format
A          R HEAD_FMT                   SKIPB(2) SPACEA(1)
A                                      2DATE(*YY) EDTCDE(Y)
A                                     30'Purchase Orders'
A                                     50TIME
A                                     60'Page:'
A                                     +1PAGNBR EDTCDE(Z)

A                                     32'Line Items' SPACEB(1)

A                                     10'Item' SPACEB(2)
A                                     22'PO'
A                                     32'Quantity'
A                                     45'Item Cost'
A                                     60'Line Cost'

 ** Detail format
A          R LNITM_FMT                  SPACEB(1)
A            ITMNBR     R             10
A            PONBR      R             20
A            POLQTYOO   R             30EDTCDE(J)
A            POLITMCOSTR              47EDTCDE(J)
A            POLINECOSTR   +3         58REFFLD(POLITMCOST)
A                                      EDTCDE(J)

 ** Subtotal Format
A          R SUBTOT_FMT                 SPACEB(2) SPACEA(1)
A                                      2'Sub-Totals'

A                                      4'No. of Line Items:'
A                                      SPACEB(1)
A            SUBCOUNT       5S 0      22EDTCDE(Z)
A            SUBQTY     R   +2        27REFFLD(POLQTYOO)
A                                      EDTCDE(J)
A            SUBCOST    R   +2        59REFFLD(POLITMCOST)
A                                      EDTCDE(J)
A  40                                 +2'**'

 ** Totals Format
A          R TOTAL_FMT                  SPACEB(2)
A                                      2'Total of Line Items:'

A                                      4'Count:' SPACEB(1)
A            TOTCOUNT   R   +2         20REFFLD(SUBCOUNT *SRC)
A                                      EDTCDE(Z)
A            TOTCOST    R   +4         56REFFLD(POLITMCOST)
A                                      EDTCDE(J)
```

```
 ** File declarations
FPOItem_LF IF   E           K Disk
FPopLnItmS O    E             Printer Oflind(*In73)

 ** Work variable declarations
D FirstPage       S                N   Inz(*On)
D LineItem        S                    Like(ItmNbr)

C                   Read      POItem_LF
C                   If        Not %Eof(POItem_LF)

 ** First-time processing
C                   If        FirstPage
C                   Write     Head_Fmt
C                   Eval      FirstPage = *Off
C                   Eval      LineItem  = ItmNbr
C                   Endif

 ** Check for page overflow
C                   If        *In73
C                   Write     Head_Fmt
C                   Eval      *In73 = *Off
C                   Endif

 ** Check for Item Number change and print subtotal
C                   If        LineItem <> ItmNbr

C                   Eval      *In40 = SubQty > 6

C                   Write     SubTot_Fmt
C                   Eval      LineItem  = ItmNbr
C                   Eval      SubCount = 0
C                   Eval      SubCost  = 0
C                   Eval      SubQty   = 0
C                   Endif

 ** Accumulate subtotal values before printing details
C                   Eval      PoLineCost = PolQtyOO * PolItmCost
C                   Eval      SubCost  = SubCost + PoLineCost
C                   Eval      TotCost  = TotCost + PoLineCost
C                   Eval      SubQty   = SubQty  + PolQtyOO
C                   Eval      SubCount = SubCount + 1
C                   Eval      TotCount = TotCount + 1

C                   Write     LnItm_Fmt

C                   Else

 ** Print the final subtotal record and the total format
C                   Eval      *In40 = SubQty > 6

C                   Write     SubTot_Fmt

C                   Write     Total_Fmt
```

```
C                   Eval      *InLR = *On
C                   Endif
```

Debugging an RPG IV Program

```
 ****************************************************************************
 *                          VNRADR05S
 ****************************************************************************
 * Active Vendor Report
 *
 * Print listing of ACTIVE vendors from Vendor_PF file.
 *
 * INDICATORS:
 *   LR - Last record
 ****************************************************************************
 *
 * DATE        PROGRAMMER      DESCRIPTION
 * ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 * 5/30/00     D.  J.  Enger     New Program
 *
 ****************************************************************************
H DatFmt(*USA)
 ****************************************************************************
FVendor_PF IF   E           K DISK

FQPRINT    O    F  132        Printer OflInd(*InOF)

 ****************************************************************************
DToDaysDate       S               D
DTime             S              6  0
DCount            S              9  0
 * Add alpha field to contain all of salesperson name that fits report
DVndSales16       S             16
 ****************************************************************************
 * Obtain job date and current time for report heading
C                   Move      *DATE         ToDaysDate
C                   Time                    Time
 ****************************************************************************
 *
C                   Except    ExcVndHdr                                Print page 1 heading
 *
C                   Read      Vendor_Fmt                         LR
C                   DoW       Not *InLR                                Read all records
 *
 * Include selection for only active records
C                   If        VndActive = 'A'
 * Move lefmost characters of salesperson into a field that fits report
C                   Eval      VndSales16=%Subst(VndSales:1:16)

C                   Except    ExcVndDtl                                Print detail record
 * Keep a count of the records processed
C                   Eval      Count=Count+1

C                   End
C                   Read      Vendor_Fmt                         LR
C                   End

C                   Except    ExcVndTot                                Print record count
C                   Return
 ****************************************************************************
OQPrint    E            ExcVndHdr      3 06
O         OR    OF
```

```
 *                    Page Heading
O                                          'Page'
O                    Page             +   2
O                                       55 'Active Vendors'
O                    TodaysDate         122
O                    Time               132 '  :  :  '

O          E         ExcVndHdr      1
O         OR    OF
 *                   Column Heading Line # 1
O                                      114 'Vend'
O                                      122 'Vendor'

O          E         ExcVndHdr      1
O         OR    OF
 *                   Column Heading Line # 2
O                                        5 'Vend'
O                                       89 'VND'
O                                       98 'Zip'
O                                      104 'Area'
O                                      114 'Tel'
O                                      121 'Sales'

O          E         ExcVndHdr      2
O         OR    OF
 *                   Column Heading Line # 3
O                                        5 'Num'
O                                       32 'Vendor Name              '
O                                       59 'Vendor Street            '
O                                       84 'Vendor City            '
O                                       91 'State'
O                                       98 'Code'
O                                      104 'Code'
O                                      114 'No'
O                                      122 'Person'

 ***************************************************************************
O          EF        ExcVndDtl      2
 *                   Vendor Detail Information
O                    VndNbr         Z     5
O                    VndName              32
O                    VndStreet            59
O                    VndCity              84
O                    VndState             88
O                    VndZipcode           98
O                    VndAreaCd           104
O                    VndTelNo            114 '0   -    '
O                    VndSales16          132

O          EF        ExcVndTot   2
O                                       32 'Number of active vendors:'
O                    Count          1    45
O                                       47 '*'
```

Structured Programming

This program uses CALLs; in this class, you used subroutines and reworked a different program.

```
 ** File declarations
FPodLnq_LF IF   E           K Disk

D Delinquent      S             15P 5
D POCount         S              2P 0
D BkOrdTotal      S              7P 2
D StopCode        S              1A   Inz('N')

D PrvPONbr        S                   Like(PONbr)
D PrvVndNbr       S                   Like(VndNbr)
D PrvVndName      S                   Like(VndName)
D PrvVndSales     S                   Like(VndSales)
D PrvVndAreaCD    S                   Like(VndAreaCD)
D PrvVndTelNo     S                   Like(VndTelNo)

C     *Entry        Plist
C                   Parm                    Delinquent

C     PorFaxParm    Plist
C                   Parm                    Delinquent
C                   Parm                    BkOrdtotal
C                   Parm                    PrvVndName
C                   Parm                    PrvVndSales
C                   Parm                    PrvVndAreaCd
C                   Parm                    PrvVndTelNo
C                   Parm                    POCount
C                   Parm                    StopCode

C                   Read      PodLnq_LF
C                   Exsr      SaveDetail
C                   Dow       Not %Eof(PodLnq_LF)

C                   If        PoDate < Delinquent

C                   If        PrvVndNbr = VndNbr

 ** New Purchase Order?  If so, add to count
C                   If        PrvPONbr  <> PONbr
C                   Eval      POCount = POCount + 1
C                   Eval      PrvPONbr = PONbr
C                   Endif

C                   Else
 ** New Vendor - print/fax existing details before proceeding
C                   If        POCount > 0
C                   Call      'PORFAX'      PorFaxParm
C                   Endif
C                   Exsr      SaveDetail
C                   Endif
```

```
 ** Accumulate totals regardless
C                   Eval      BkOrdTotal = BkOrdTotal +
C                               ((PolQtyOO - PolQtyRec) * PolItmCost)
C                   Endif

C                   Read      PodLnq_LF
C                   Enddo

C                   Eval      StopCode = 'Y'
C                   Call      'PORFAX'      PorFaxParm

C                   Eval      *InLR = *ON
C                   Return

C     SaveDetail    Begsr
C                   Eval      PrvVndNbr    = VndNbr
C                   Eval      PrvVndName   = VndName
C                   Eval      PrvVndSales  = VndSales
C                   Eval      PrvVndAreaCD = VndAreaCD
C                   Eval      PrvVndTelNo  = VndTelNo
C                   Eval      POCount = 0
C                   Eval      BkOrdTotal = 0
C                   Endsr
```

Maintaining Database Files

```
 ****************************************************************************
 *                            PORMNT02S                                     *
 *                      PO Maintenance - Open Line Items                    *
 ****************************************************************************
 *                                                                           *
 * This program allows changes to the Open Line Items:                       *
 *                                                                           *
 * INDICATORS:                                                               *
 *   10 - Record not found                                                   *
 *   11 - Record exists (duplicate add)                                      *
 *   12 - Record added                                                       *
 *   13 - Record deleted                                                     *
 *   14 - Record changed                                                     *
 *   15 - Invalid Transaction                                                *
 *   88 - Printer Overflow                                                   *
 ****************************************************************************
 ** PO Transaction File
FPOTRANS_PFIF   E             DISK
 ** Open Line Items File
FPOOPNLI_LFUF A E           K DISK
 ** Maintenance Printer File
FPOPMNT02  O    E             PRINTER OFLIND(*in88)
 **
 ** Build the Line Item Key
C     LItmKey       KLIST
C                   KFLD                    POTPONBR
C                   KFLD                    POTITMNBR
 ** Write headings on first page of report
C                   Write     Heading
 ** Read first transaction record
C                   Read      POTrans_PF
C                   DoW       not %eof(POTrans_PF)
 ** Reset Indicators
C                   Eval      *in10 = *off
C                   Eval      *in11 = *off
C                   Eval      *in12 = *off
C                   Eval      *in13 = *off
C                   Eval      *in14 = *off
C                   Eval      *in15 = *off
 ** Process transactions
C                   Select
 *
C                   When      Trncode = 'A'
C                   Exsr      AddSr
 *
C                   When      Trncode = 'D'
C                   Exsr      DelSr
 *
C                   When      Trncode = 'C'
C                   Exsr      ChgSr
C                   Other
C                   Exsr      InvTrSr
```

```
C                   EndSl
 ** Read next transaction record
C                   Read      POTrans_PF

C                   EndDo
 ** EOJ Processing
C                   Exsr      PrtTotals
C                   Eval      *inlr = *on

 ** Additions
C     ADDSR         BEGSR
 ** Record already exists?
C     LItmKey       Chain     POOPNLI_LF
C                   If        not %found(POOPNLI_LF)
C                   Eval      ponbr = potponbr
C                   Eval      ITMNBR = potitmnbr
C                   EVAL      POLQTYOO = POTQTYOO
C                   Eval      POLITMCOST = POTITMCOST
C                   EVAL      POLDATREC = POTDATREC
C                   EVAL      POLQTYREC = POTQTYREC
C                   Eval      POLSTATUS = POTSTATUS
 ** If no record found (OK), add and seton 12
C                   WRITE     POLINE_FMT
C                   EVAL      *IN12 = *ON
C                   Eval      Addcnt = Addcnt + 1
 ** If record does exist (error), seton 11
C                   Else
C                   Eval      *in11 = *on
C                   Eval      ErrCnt = ErrCnt + 1
C                   EndIf
 ** Print transaction
C                   Exsr      PrtDetail
C                   ENDSR
 ** Deletions
C     DelSR         BEGSR
 ** Delete the record
 ** If no record found (Error), seton 10
 ** If record exists (OK), delete and seton 13
C     LItmKey       DELETE    POOPNLI_LF
C                   If        %found(POOPNLI_LF)
C                   Eval      *IN13 = *ON
C                   Eval      Delcnt = Delcnt + 1
C                   Else
C                   Eval      *in10 = *on
C                   Eval      Errcnt = Errcnt + 1
C                   EndIf
C
C                   Exsr      PrtDetail
C                   ENDSR
 ** Changes
C     ChgSr         BEGSR
 ** If no record, seton *IN10
C     LItmKey       CHAIN     POOPNLI_LF
C                   If        %found(POOPNLI_LF)
 * Print Original fields
C                   Exsr      PrtPOOrig
```

```
 * Update quantity
C                   Eval      ponbr = potponbr
C                   Eval      ITMNBR = potitmnbr
C                   EVAL      POLQTYOO = POTQTYOO
C                   Eval      POLITMCOST = POTITMCOST
C                   EVAL      POLDATREC = POTDATREC
C                   EVAL      POLQTYREC = POTQTYREC
C                   Eval      POLSTATUS = POTSTATUS
 ** Change the record
C                   UPDATE    POLINE_FMT
 ** If record exists, seton 14
C                   EVAL      *IN14 = *ON
C                   Eval      Chgcnt = Chgcnt + 1
C                   Else
C                   Eval      *in10 = *on
C                   Eval      Errcnt = Errcnt + 1
C                   EndIf
 ** Print transaction
C                   Exsr      PrtDetail
C                   ENDSR
C
 ** Invalid Transaction
C     InvTrSr       BegSr
C                   Eval      *in15 = *on
C                   Eval      Errcnt = Errcnt + 1
C                   Exsr      PrtDetail
C                   Eval      *in15 = *off
C                   EndSr
 ** Print Totals Subroutine
C     PrtTotals     Begsr
C                   Eval      TotCnt = ChgCnt + DelCnt + ErrCnt + Addcn
C                   ExSr      CheckOV
C                   Write     Total
C                   EndSr
 **  Print original record fields, transaction and new fields
C     PrtPOOrig     BegSr
C                   ExSr      CheckOV
C                   Write     POOrig
C                   EndSr
 **
C     PrtDetail     BegSr
C                   ExSr      CheckOV
C                   Write     TrnDetail
C                   EndSr
 ** Check for Overflow
C     CheckOV       BegSr
C                   If        *in88
C                   Write     Heading
C                   Eval      *in88 = *off
C                   EndIf
C                   EndSr
```

Coding an Inquiry Program

```
A*********************************************************************** ****
A*                           VNDINQS                                    *
A*              Inquiry by Vendor Number Display File                   *
A*********************************************************************** ****
A* THIS DISPLAY FILE CONTAINS THESE FORMATS:                            *
A*                                                                      *
A*     PROMPT_FMT - Prompts for Vendor Number                           *
A*     DSPLY_FMT  - Displays a vendor record                            *
A*                                                                      *
A* INDICATORS:                                                          *
A*   01 - First time indicator                                          *
A*   03 - User requests to exit the program                             *
A*   96 - Invalid vendor number                                         *
A*********************************************************************
A*                       CHANGE LOG                                     *
A* DATE        PROGRAMMER      DESCRIPTION                              *
A* ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ *
A* 8/01/96     N. D. Cator     New Display File                         *
A*********************************************************************
A                                      REF(*LIBL/VENDOR_PF)
A                                      CA03(03 'End Program')
A          R PROMPT_FMT
A                                    1  2USER
A                                    1 30'Vendor Inquiry'
A                                      COLOR(WHT)
A                                    1 71SYSNAME
A                                    2 61DATE
A                                      EDTCDE(Y)
A                                    2 71TIME
A                                    3  3'Vendor number. . . . :'
A            VNDNBR_INQR     D  I  3 28COLOR(WHT)
A                                      REFFLD(VNDNBR DICTIONARY)
A  96                                  ERRMSG('Invalid vendor number - pre-
A                                      ss reset and re-enter' 96)
A                                   22  3'Please press enter to continue'
A                                   23  4'F3 = Exit'
A                                      COLOR(BLU)
A          R DSPLY_FMT
A                                      CA12(12 'Return to previous display-
A                                      Display')
A                                    1  2USER
A                                    1 30'Vendor Inquiry'
A                                      COLOR(WHT)
A                                    1 71SYSNAME
A                                    2 61DATE
A                                      EDTCDE(Y)
A                                    2 71TIME
A                                    7  3'Vendor number . .:'
A            VNDNBR    R        O  7 24
A                                    8  3'Name . . . . . . :'
A                                    9  3'Address  . . . . :'
A            VNDNAME   R        O  8 24
```

```
A            VNDSTREET R        O  9 24
A            VNDCITY   R        O 10 24
A            VNDSTATE  R        O 10 49
A            VNDZIPCODER        O 10 53
A                                 11  3'Telephone. . . . :'
A            VNDAREACD R        O 11 26
A                                 11 24'('
A                                 11 30')'
A            VNDTELNO  R        O 11 33
A                                      EDTWRD('0  -    ')
A                                 12  3'Sales Person . . :'
A            VNDSALES  R        O 12 24
A                                 13  3'Purchases YTD  . :'
A            VNDPRCHYTDR          13 24EDTCDE(J)
A                                 14  3'Balance Owed . . :'
A            VNDBALANCER          14 26EDTCDE(J)
A  60                                  DSPATR(HI)
A  60                                  COLOR(RED)
A                                 23  4'F3 = Exit  F12 = Return to previou-
A                                      s Display'
A                                      COLOR(BLU)
```

```
 ** RPG program......:  VNRINQS
 ** Vendor master File
FVendor_PF IF   E           K Disk
 ** Display File
FVndinqs   CF   E             Workstn
 **
C                   Dow       NOT *In03
C     Vndnbr_inq    Chain     Vendor_PF

C                   If        %found(Vendor_PF)
C                   DoW       NOT *In12
 * Check whether balance owed is greater than 5000.00
C                   Eval      *in60 = VndBalance > 5000.00
 ** Display details
C                   Exfmt     Dsply_fmt

C                   IF        *In03
C                   Eval      *InLR = *ON
C                   Return
C                   Endif
c                   EndDo
c                   Else
C                   Eval      *In96 = *on
c                   endif
 ** No Item record found or F12 - display prompt
C                   Eval      *In12 = *OFF
C                   Exfmt     Prompt_fmt
c                   enddo
C                   Eval      *InLR = *ON
C                   Return
C                   IF        *In03
C                   Eval      *InLR = *ON
```

```
C                   Return
 ****************************************************************
C     *Inzsr        Begsr
C                   Exfmt     Prompt_fmt
C                   Endsr
```

Appendix D. Rational Developer for Power Systems

Sample startup and use example for Exercise 1

Part 1: Start Remote Systems Explorer (RSE)

RSE allows you to connect to an i server and define a collection of filters (lists of items that you define). You can then access and operate on whatever items are of interest you.

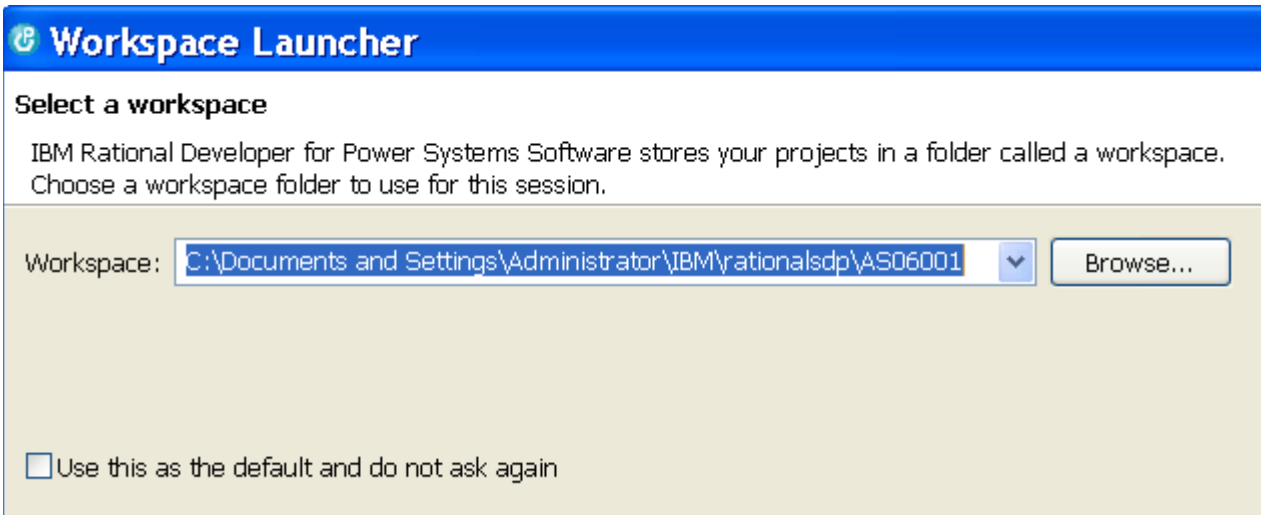
First you must start the product. Follow these steps to start the product:

__ 1. Click **Start** on the task bar of your Desktop.

__ 2. Select **Programs** or **All Programs**, depending on your system.

__ 3. Select **IBM Software Delivery Platform > IBM Rational Developer for Power Systems Software > IBM Rational Developer for Power Systems Software**.

__ 4. A dialog will appear. Here you specify the directory of the workspace where your projects and other resources, such as folders, subfolders, and files that you are developing in the workbench, will reside.

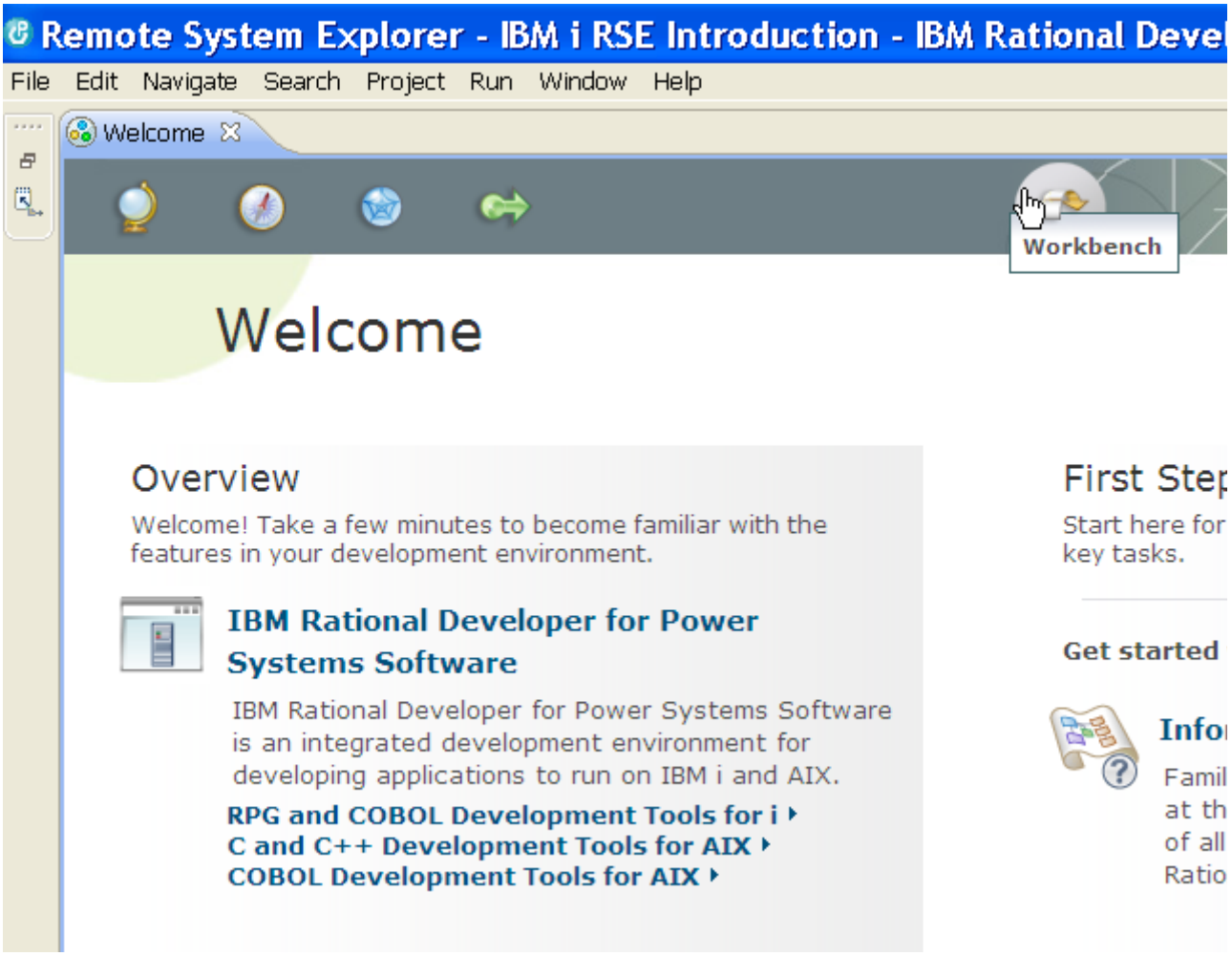


__ 5. Change the field in this dialog and use a unique directory name, for example, AS06*nnn* (where *nnn* is your team number, such as AS06001).

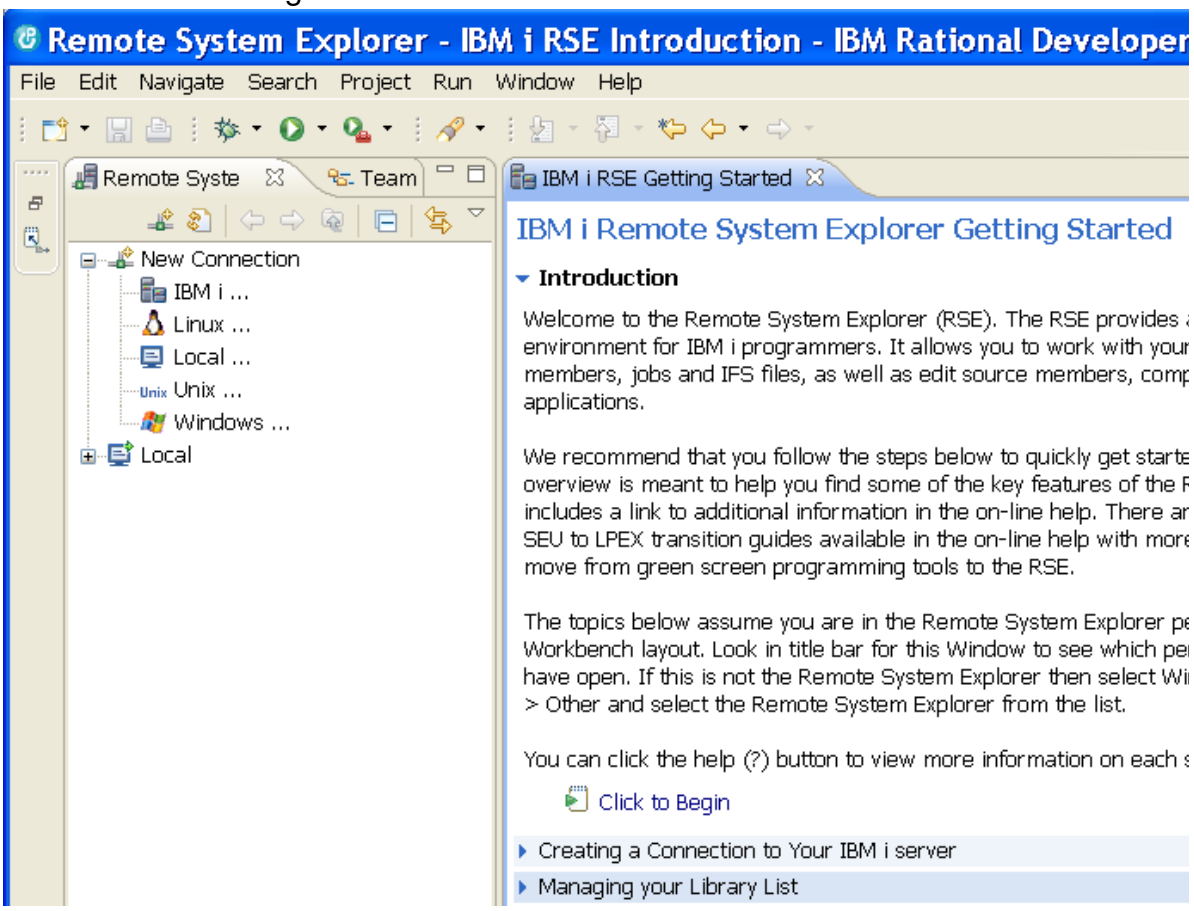
Make sure the **Use this as the default and do not ask again** checkbox is not selected.

__ 6. Click **OK** to open the Workbench.

__ 7. Click the **icon** in the top, middle of the Welcome page to go to the Workbench.



__ 8. Click the **maximize** button to maximize or the **restore** button to return the Workbench to its original size.

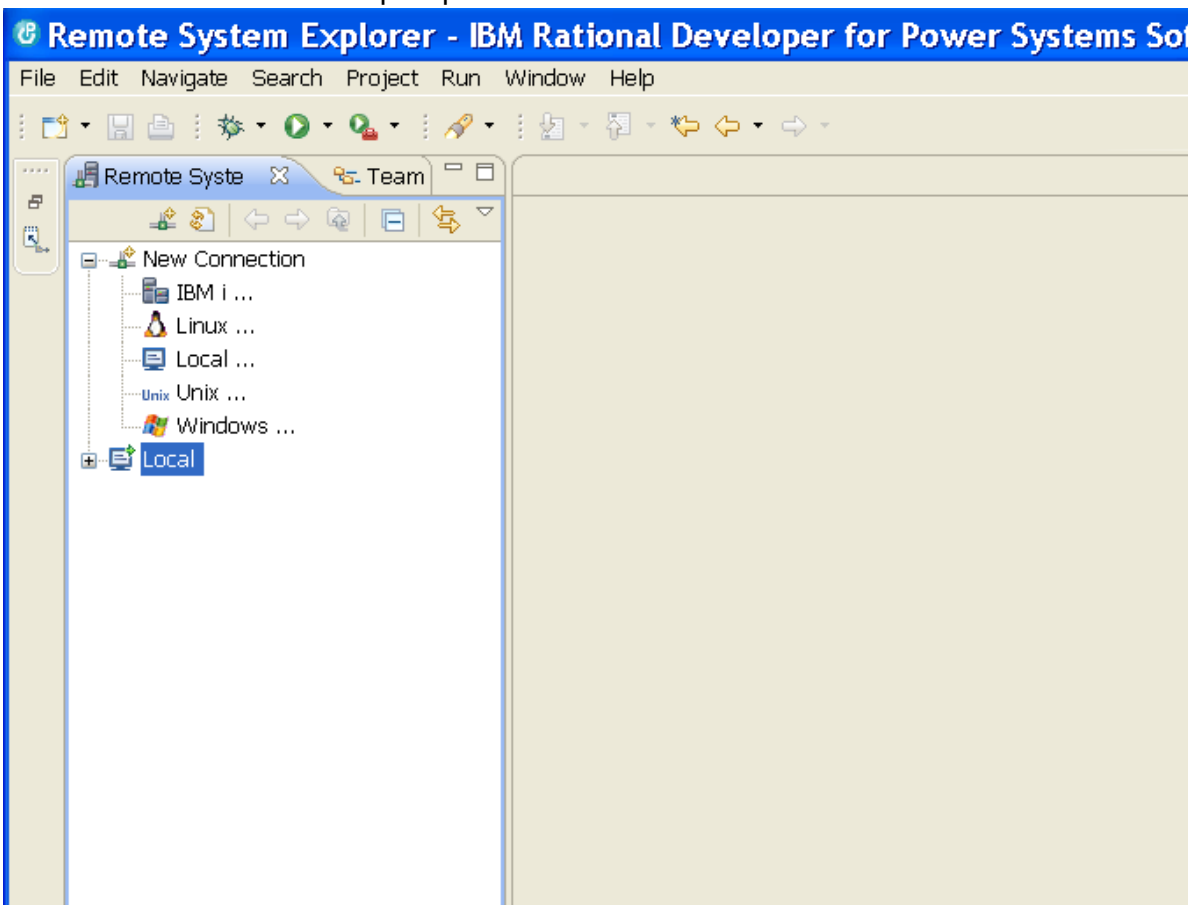


__ 9. Click the **X** in the **IBM i RSE Getting Started** tab. This display is always available by choosing the **Help** > **IBM i RSE Getting Started** menu option.

A *workbench* is a desktop development environment. The workbench aims to achieve seamless tool integration and controlled openness by providing a common paradigm for the creation, management, and navigation of workbench resources. Each workbench window contains one or more views and an editor.

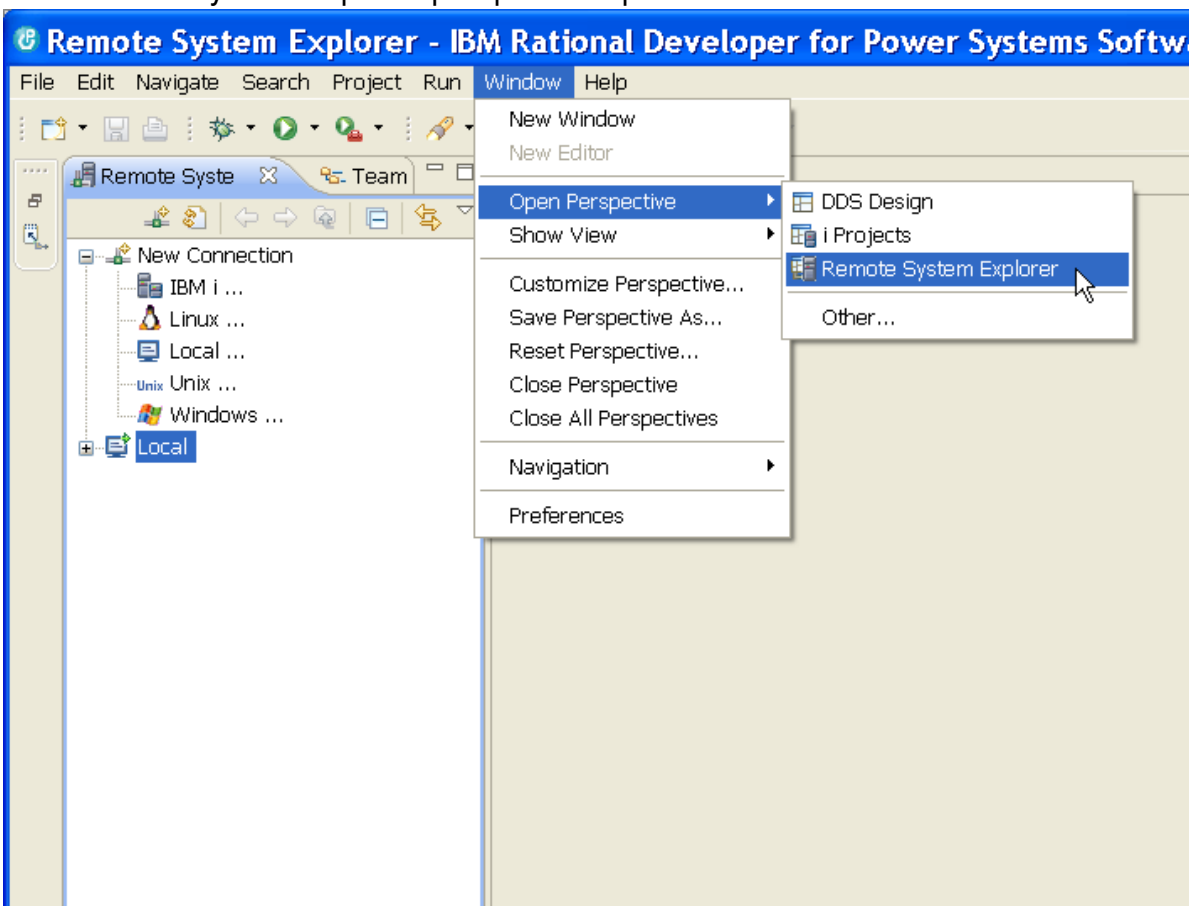
Part 2: Opening the Remote System Explorer perspective

__ 10. Check for the name of the perspective.



__ 11. If you see a different perspective (not the Remote System Explorer) in the Workbench, or no perspective, click **Window > Open Perspective > Remote System Explorer** from the Workbench menu.

The Remote System Explorer perspective opens.

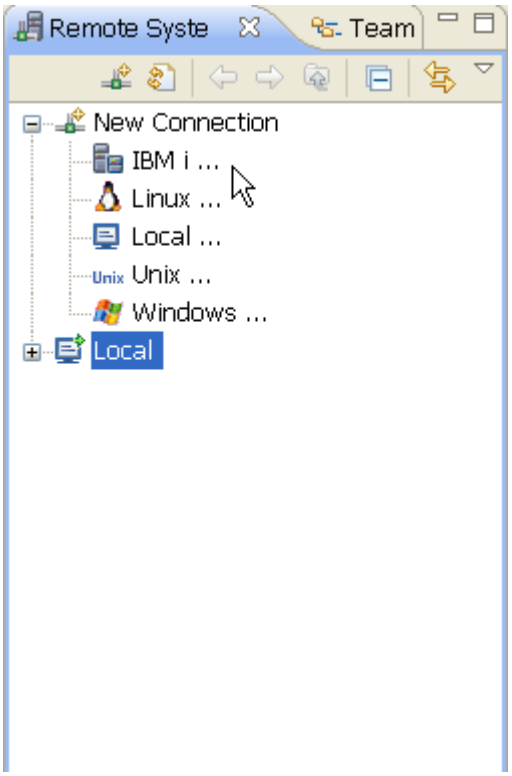


You work in the Remote System Explorer perspective in the Workbench. This perspective allows an IBM i programmer to display the connections that she has already configured, create a new connection, connect to and disconnect from the connections that she has defined, and work with IBM i files, commands, jobs, and integrated file system files.

When you first open the Remote System Explorer, you are not connected to any System except your local hard drive on your workstation. To connect to a remote IBM i system, you need to define a connection. When you define a connection, you specify the name or IP address of the remote system and you give your connection a unique name that acts as a label in your workspace so that you can easily connect and disconnect. When you connect to the IBM i system, the workbench prompts you for your user ID and password on that host.

The first time you connect to an IBM i system, you need to specify a profile. All connections, filters, and filter pools belong to profiles.

__ 12. In the **Remote Systems** view, New Connection is automatically expanded to show the various remote systems types you can connect to through the **Remote System Explorer**.

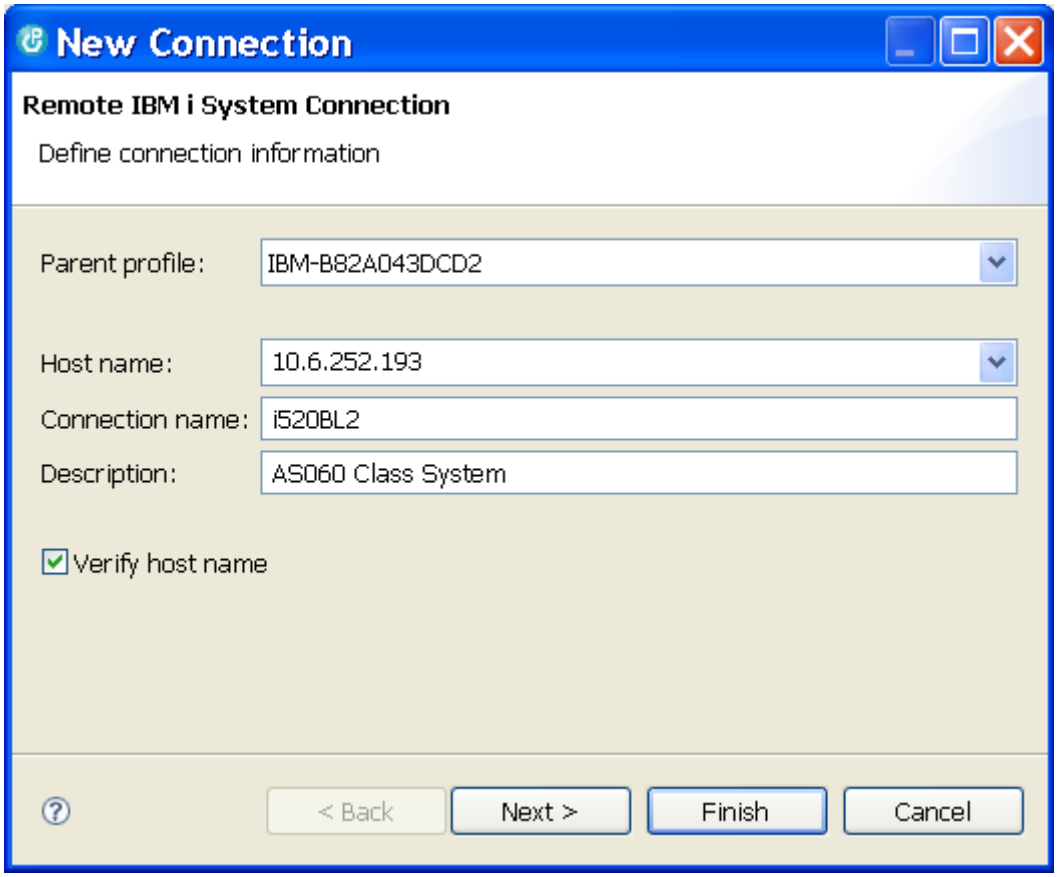


__ 13. Double-click the **IBM i** icon to configure a connection to an i system.

__ 14. The profile defaults to the name of the workstation. The **Remote IBM i System Connection** page opens.

On this page, you specify the information for your connection. The cursor on this page is positioned in the **Host name** field.

In the Host name field, type **<*i_server*>** (check with your instructor if you do not remember the system name or IP address to enter).



__ 15. Leave the **Parent profile** default value. You do not need to change it.

__ 16. Leave the **Verify host** name checkbox selected.

__ 17. Click **Finish** to define your system.

You have configured a connection.

__ 18. In the **Remote Systems** view, your new connection is expanded to reveal your subsystems. The **Objects** subsystem is the subsystem you will use most often. It is very similar to PDM in that it allows you to access objects in the QSYS file system and to perform actions on those objects.

__ 19. Once expended, the first three entries under the **Objects** subsystem are named after the PDM options, because they have similar capabilities:

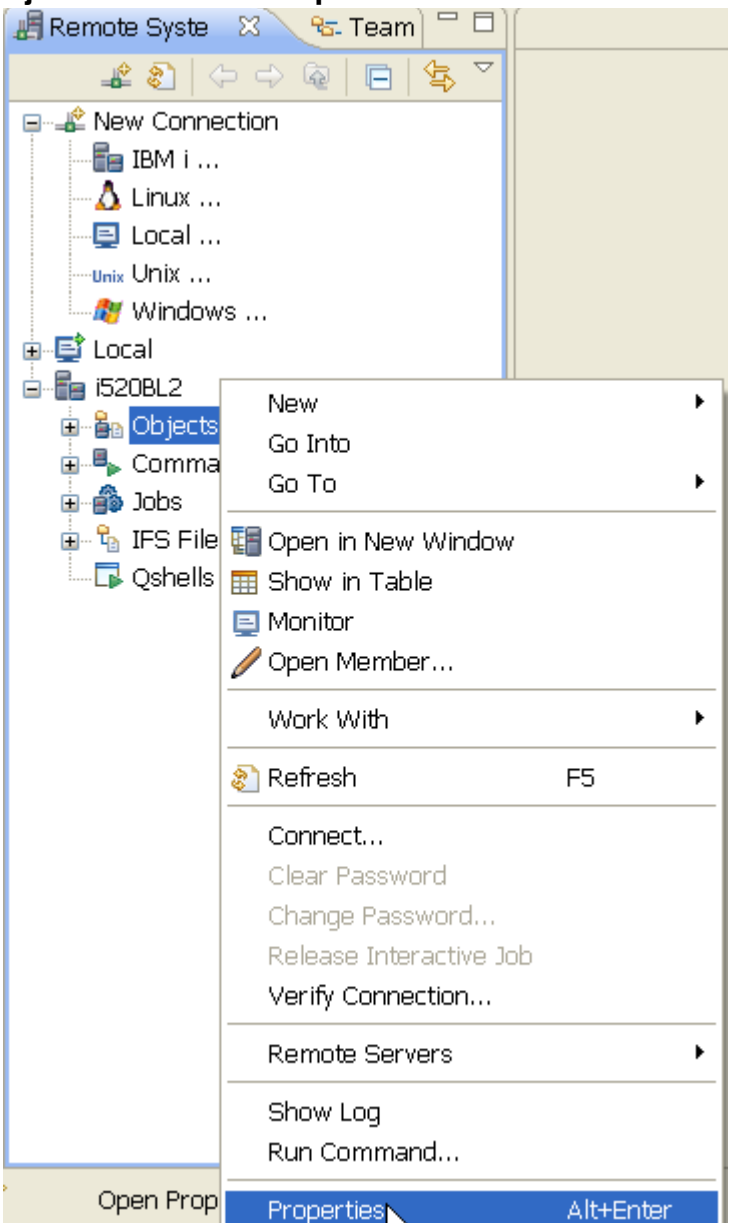
- Work with libraries is similar to WRKLIBPDM.
- Work with objects is similar to WRKOBJPDM.
- Work with members is similar to WRKMBRPDM.
- In addition, there are entries for working with library lists and user libraries:

- Library list is similar to WRKLIBPDM in PDM. You can start with the predefined library list filter that, when expanded, lists all libraries in your library list.
- User libraries allow you to work with all user libraries you can access on that i server.

You also have more entries to work with under the connection itself, and you can see from these entries that Remote System Explorer goes well beyond PDM. It allows you to explore i jobs and commands and the integrated file system.

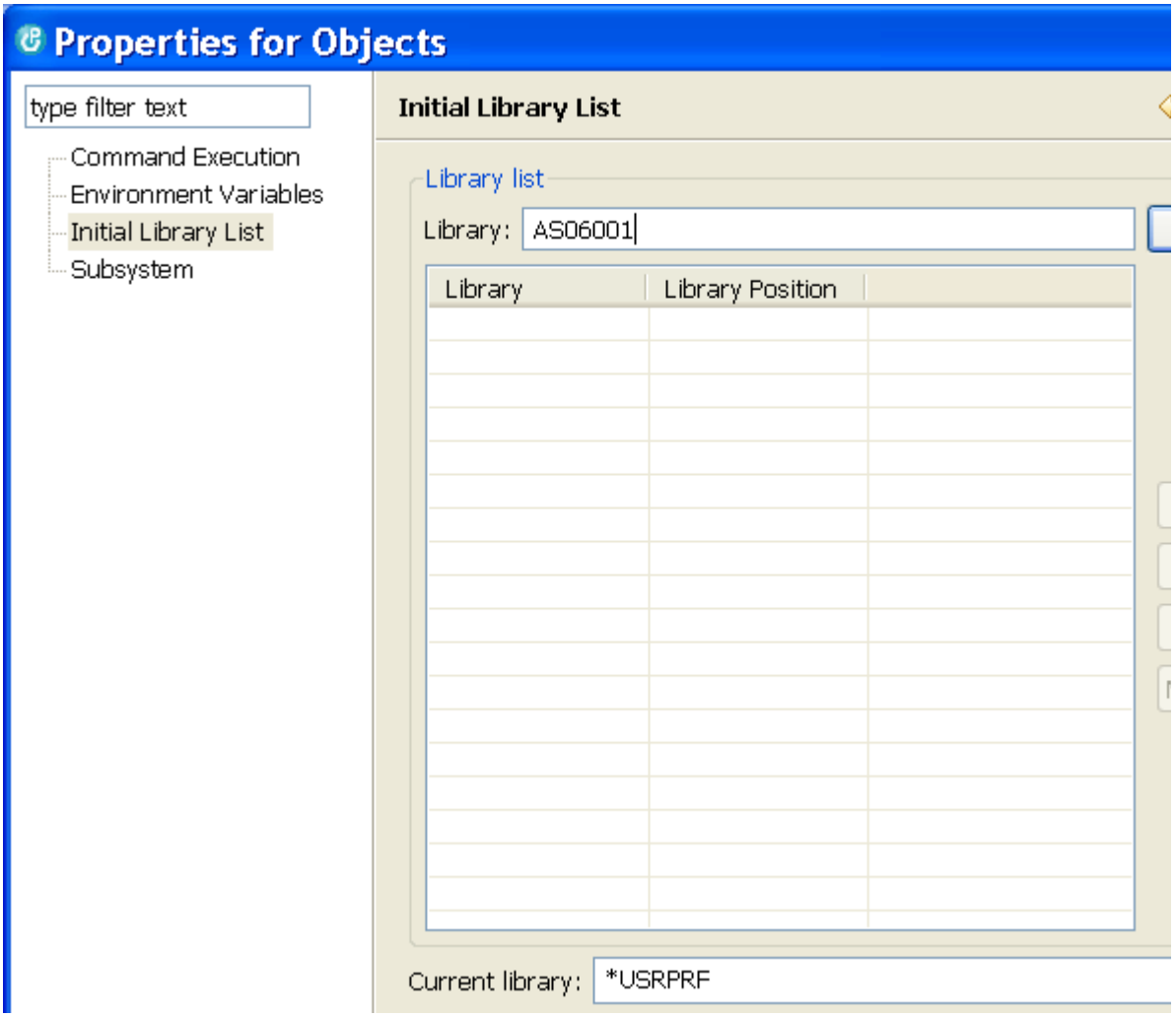
__ 20. Work with a library in your library list, and add the library that you will be using in this exercise:

Right-click **Objects** and select **Properties** on the menu.



__ 21. Select **Initial Library List** on the left pane.

__ 22. Type the name of **<your library>** in the Library field, and click **Add**.



__ 23. Click **OK**.

This will add the library to your library list every time you use this connection.

__ 24. Expand the Library list folder.

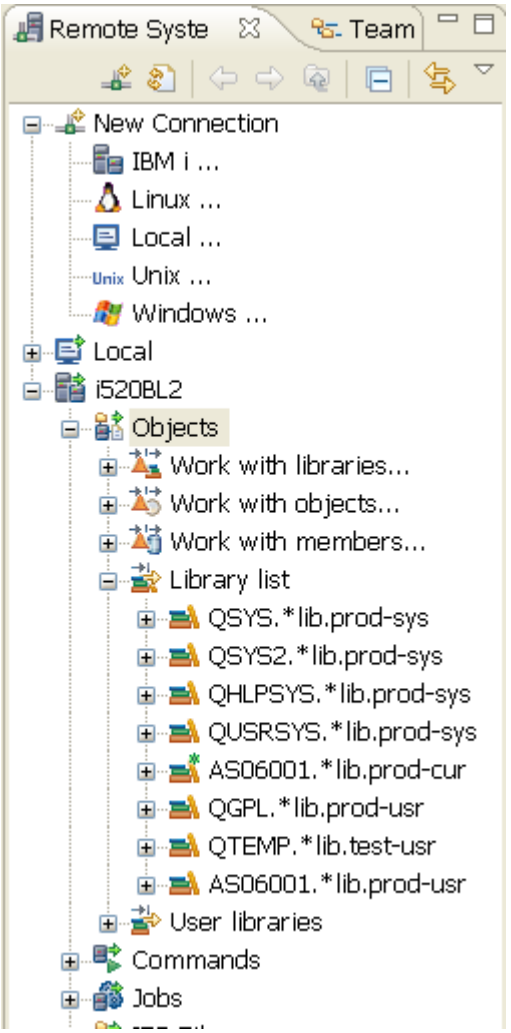
Now, the connection will be activated, and you will be prompted for a user ID and password.

__ 25. Enter **<*i_userid*>** and **<*i_password*>**.

Make sure that the **Save user ID** checkbox is unchecked.
Make sure that the **Save password** checkbox is unchecked.
Click **OK**.

As you know, you can use the properties of any of the subsystems to set connection information such as adding a library to a library list. Back in the Workbench in the

Remote Systems view when you click the **Refresh** icon, you will see the libraries in your job's library list.

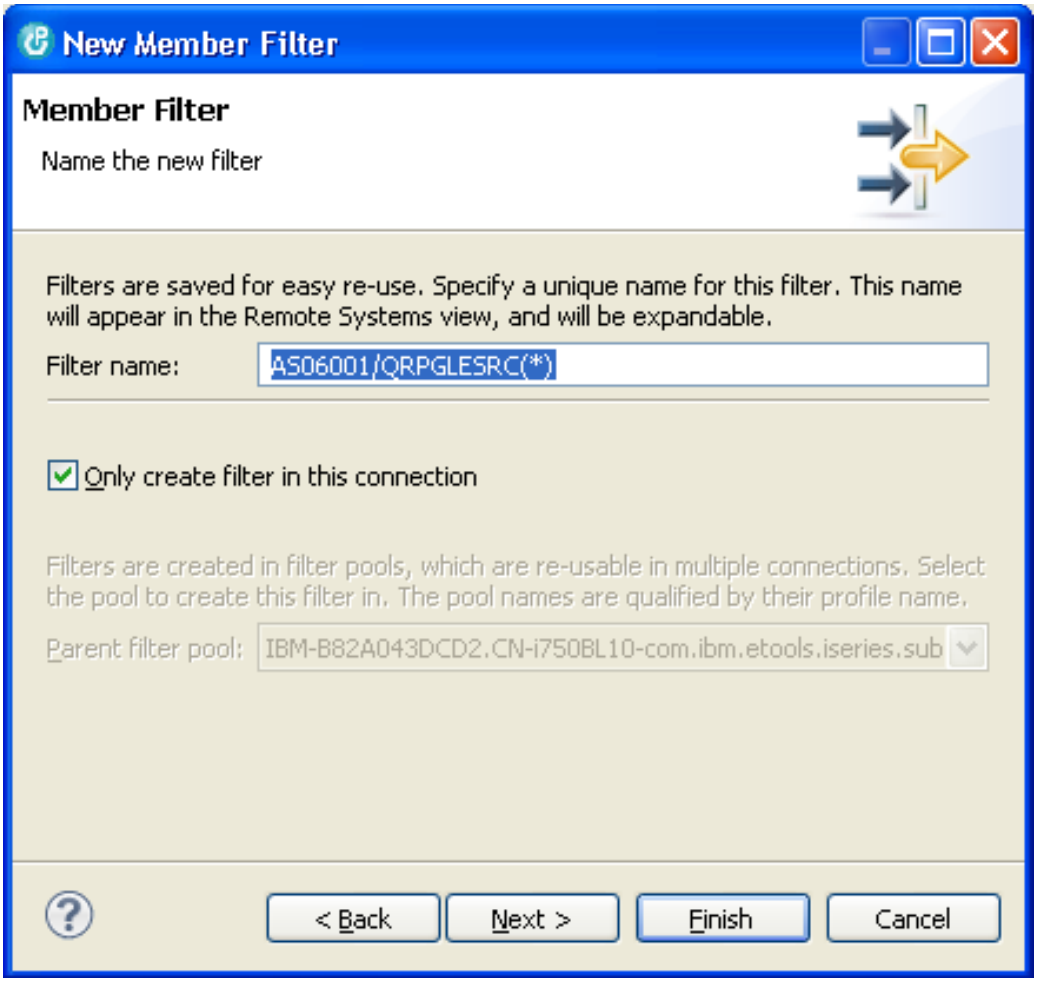


__ 26. Click the **+** beside **Work with members**...

__ 27. In the Library entry field, type **`AS06nnn`**.

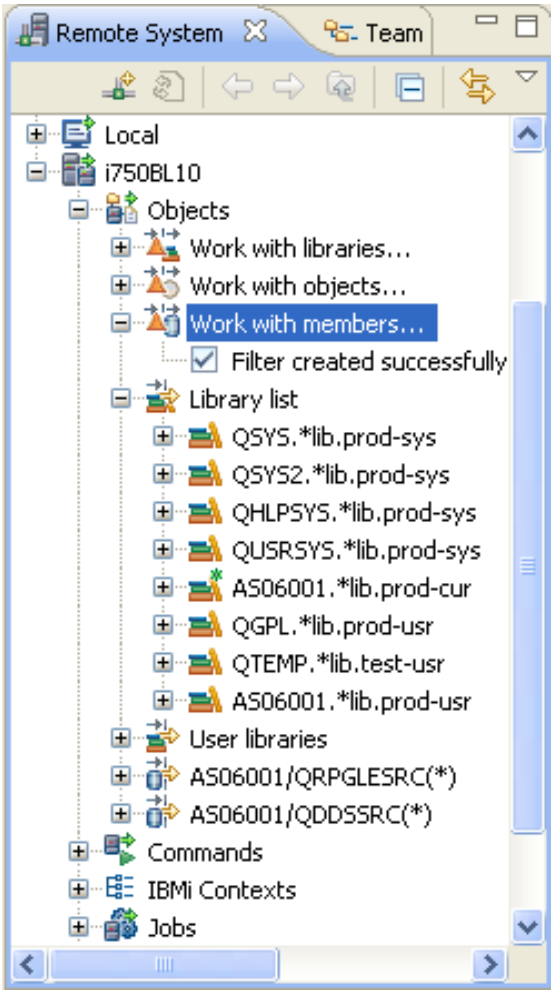
__ 28. In the File entry field, type **`QRPGLESRC`**.

__ 29. Click the **Next** button, and on the subsequent filter name display, type `QRPGLESRC` as the name of this new filter. Click **Finish**.



__ 30. Repeat the above steps to add a filter for your `QDDSSRC` source file in your library.

__ 31. Your RSE Workbench should now look similar to the one below.



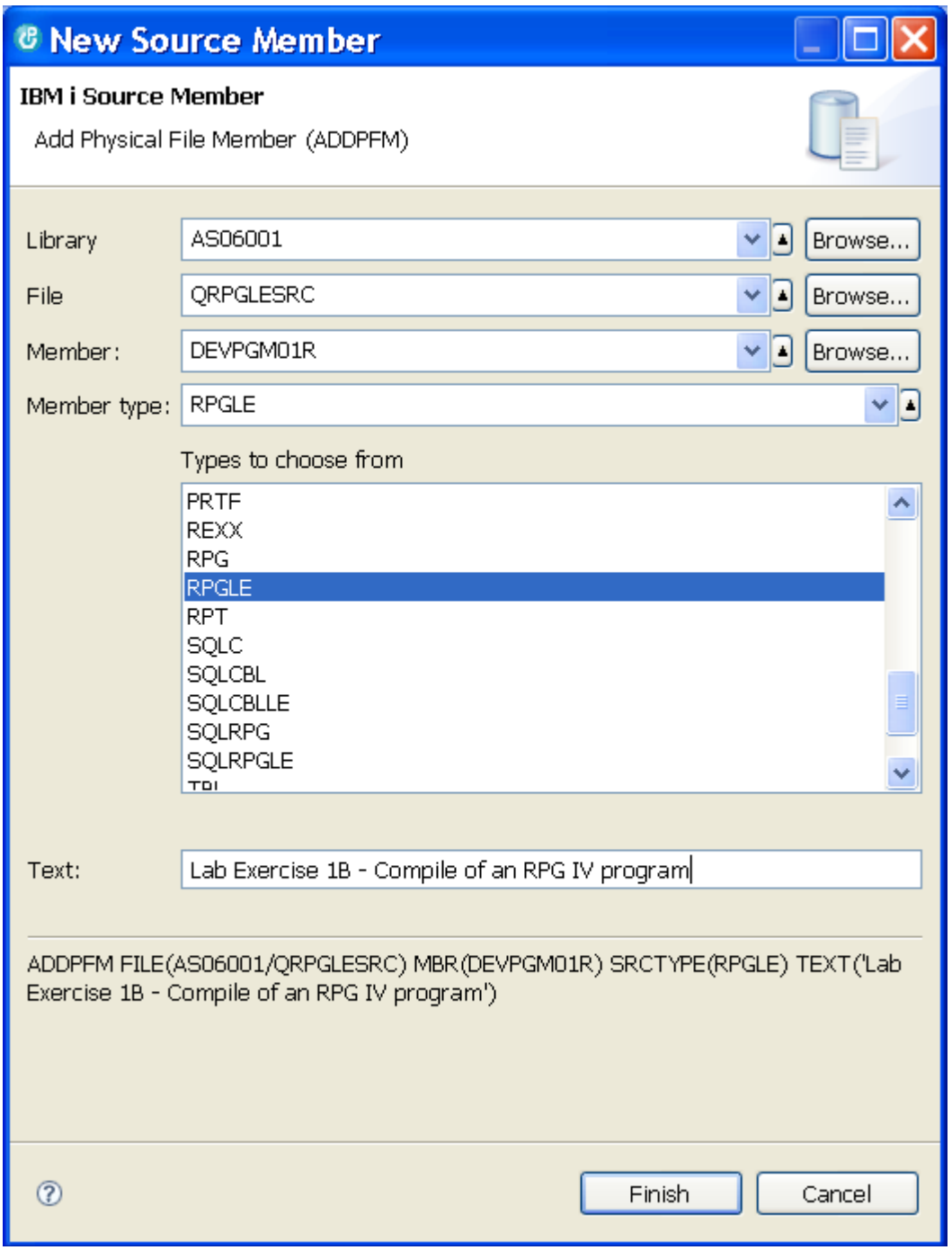
Part 3: Start the LPEX editor

__ 32. Right-click your **QRPGLESRC** filter icon and select **New > member**.

__ 33. On the **IBM i Source Member** display, verify that your library AS06*nnn* and source file `QRPGLESRC` are already completed. For member name, type `DEVPGM01R`. For member type, use the scroll bar to select **RPGLE**. For the text prompt, type the following:

```
Lab Exercise 1B Compile of an RPG IV program.
```

Click **Finish**.



__ 34. Once the editor window opens, enter the **code** that follows into your source member. Notice that you can use prompting (**F4**) to assist you in order to enter data correctly into the RPG IV specification templates.

__ 35. Here is the code that you enter into your QRPGLESRC file, member DEVPGM01. Note that the source templates are included for your reference only. Enter only the RPG IV code:

```
Do NOT enter the template line immediately below:
DName++++++++++ETDsFrom+++To/L+++IDc.Keywords++++++++Comments
DMessage          s             30    inz('The sum of 2 plus 2 is')
DSums             s              3  0 inz
Do NOT enter the template line immediately below:
 /..1....+....2....+....3....+....4....+....5....+....6....+....
 /free
  sum = 2 + 2;
  message = %trimr(Message) + ' ' + %char(sum);
  Dsply message '*REQUESTER';
  *InLr = *on;
 /end-free
```

__ 36. After pressing **F4** (Prompt), scroll through the **Source Prompter** window (one of the tabs at the bottom) to find the appropriate formats for keying the statements.

__ 37. Save the file as it is to your QRPGLESRC file on the i by selecting **File > Save**. The member is then written to your QRPGLESRC source file with a member name of **DEVPGM01R**. Do not close the member. In The next step, you will create (**compile**) the program.

Part 4: Compile your RPG IV program

__ 38. Open the source member **DEVPGM01R** in QRPGLESRC in your library AS06nnn (by clicking that item in the **RSE workspace**) if you have previously closed it. Your RPGLE program for this source file member is now ready to compile. Click **Compile > Compile >** CRTBNDRPG (without prompting). There is an error window (tab named **Error List**) if your program has any warnings or errors. The **Command Log** tab will display the status of your compile.

Part 5: Run your RPG IV program

__ 39. Sign on to a **5250** session on the i server. Issue the command DSPLIB AS06***nnn*** and check that your program **DEVPGM01R** exists as a *PGM. Assuming it does, on the command line, enter CALL DEVPGM01R. You should see the message.