RPG IV Programming Intermediate Workshop for i

(Course code AS07)

Student Exercises

ERC 5.0

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®	Balance®	DB2®
i5/OS®	Integrated Language Environment®	iSeries®
Language Environment®	Notes®	Operating System/400®
OS/400®	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.

Java and all Java-based trademarks and logos are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

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

UNIX® is a registered trademark of The Open Group in the United States and other countries.

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

February 2009 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®	Balance®	DB2®
i5/OS®	Integrated Language Environment®	iSeries®
Language Environment®	Notes®	Operating System/400®
OS/400®	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.

Java and all Java-based trademarks and logos are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

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

UNIX® is a registered trademark of The Open Group in the United States and other countries.

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

Exercises description

Exercise instructions: This section contains what it is 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.

Your instructor assigns you a team number, **nnn**. This team number is used for your userid, **AS07nnn**, and your library, **AS07nnn**. Your library contains all the objects you need to perform the exercises. At times, you are asked to copy objects from **AS07XXX,** which is the student master library.

When you are prompted to sign on to the i server, your userid will be **AS07nnn** and your password will be **AS07**. The password is set to expire; so enter a new one that you can remember easily. Read the exercise instructions carefully as you proceed through the exercise.

Which editor to use: You can use the PC-based Remote Systems LPEX editor of RDi or you can use SEU as your editor. The Remote System LPEX editor is the recommended editor. It is straightforward and easy to use.

You can open the Remote Systems LPEX editor by **Start -> Programs -> IBM Software Development Platform -> IBM Rational Developer for System i -> IBM Rational Developer for System i**.

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

Exercise 1. Using OVERLAY and PUTOVR in display files

What this exercise is about

This exercise provides an opportunity to modify a display file and program that you are given to include use of the OVERLAY and PUTOVR keywords.

What you should be able to do

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

- Code a display file with multiple formats that incorporates the OVERLAY and PUTOVR keywords
- Write an RPG IV program that interfaces with a multiple display format DSPF that includes the OVERLAY keyword

Introduction

In this exercise, you modify a program that displays information from the Vendor file. You are given a working program, including the display file.

You notice that certain information in the two display file formats is redundant.

You include the common information in separate formats and use the OVERLAY keyword to display the common information and the variable information.

The display formats, PROMPT_FMT and DSPLY_FMT, are modified so that they are coresident on the display. Both formats are present at the same time, as well as adding the new formats HEADER_FMT and FKEYS_FMT. The reasons for doing this, and making the application more complicated, are to:

- Provide a more user-friendly screen interaction. It is not necessary to toggle between the two formats, as the program currently does.
- Facilitate improved screen response, especially noticeable when the workstations are connected remotely, resulting in reduced *traffic* between the system and workstation.

Required materials

- Student Notebook
- Userid (**AS07nnn**) and password **AS07**

Exercise instructions

You can perform this exercise using either the Remote Systems LPEX editor or the SEU editor. Both are available to you on your desktop. Use the editor that you prefer.

Step 1. Getting started

__ 1. When prompted, sign on using your **AS07nnn** ID. Replace **nnn** with your student number. Your password is **AS07**. It is set to expire at first signon. Change it to something meaningful.

__ 2. In your library, you find source members for a **DSPF** named **VNDINQ** and an **RPG IV** program named **VNRINQ**. Compile both.

__ 3. On the i, call your compiled **VNRINQ** program and test it using vendor numbers ranging in value from **10001 through 10050**. Any other vendor number results in an error message.

__ 4. Review the copies of the display file and the RPG IV program that follow:

VNDINQ DDS:

```
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                                      COLOR(BLU)
A          R DSPLY_FMT
A                                      CA12(12 'Return to previous display-
A                                      ')
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)
```

VNRINQ RPG IV Program:

```
// Vendor master File
FVendor_PF IF   E           K Disk
 // Display File
FVndinq 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;
       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
```

Step 2. Copy, examine, and modify the DSPF source of VNDINQ

__ 1. Make a copy of your DDS member, **VNDINQ**, naming it **VNDINQOV**.

__ 2. Notice that the heading and footing information is common to both screen formats in the existing display file.

__ 3. The DDS currently includes two formats, **PROMPT_FMT** and **DSPLY_FMT**. As we showed you in the lecture, modify the DDS and break up the two formats into four formats in total; **HEADER_FMT, PROMPT_FMT, DSPLY_FMT** and **FKEYS_FMT**.

__ 4. Examine the existing display file's formats and use them to help you to design the four formats. Remember to use **OVERLAY** and also make sure that the formats do not overlap. Also, use the **PUTOVR** keyword to reduce the amount of data that is transferred between the system and the display. You also need to change the usage of the search field, **VNDNBR_INQ**, to both (**B**) with OVRDTA rather than input.

__ 5. **F3** is enabled at the file level currently. You should change it to be enabled in the **PROMPT_FMT** only.

__ 6. **F12** is enabled in the **DSPLY_FMT**. When you modify the DSPF and use OVERLAY, F12 will no longer be necessary. Remove it.

__ 7. Make sure that the error message continues to be displayed for an invalid vendor number.

Step 3. Copy and modify your RPG program VNRINQ

__ 1. Make a copy of your RPG IV source member, **VNRINQ**, naming it **VNRINQOV**.

__ 2. You must modify your logic when you have changed your DSPF to support the four formats rather than two. You also need to modify your F-spec to reference your VNDINQOV DSPF rather than VNDINQ.

__ 3. Use your student notebook to help you to modify the logic of your program using similar techniques as discussed in the lecture.

Step 4. Compile, test, and debug your RPG program VNRINQOV

__ 1. Use valid (**10001 - 10050**) and invalid vendor numbers to test your program.

__ 2. When your program runs correctly, you are finished with the lab.

END OF LAB

Exercise 2. Using DDS Windows

What this exercise is about

In this exercise you add a pop- up window to the program and display file you modified in the first exercise.

What you should be able to do

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

- Code a display file to support a pop-up window
- Write an RPG IV program that displays a pop-up over an existing display without erasing the current window

Introduction

In this exercise, you modify the program and display file from the first exercise to display a pop-up window over the existing window on the display.

Required materials

- Student Notebook
- Userid (**AS07nnn**) and password.

Exercise Instructions

The management team was impressed with the work you did with OVERLAY and PUTOVR. Response time has improved. One of the sales representatives has requested a change. Please add a pop-up window using the same techniques shown in the lecture. This pop up window should display the vendor name and the MTD Purchases by that vendor.

Use either the Remote Systems LPEX editor or the SEU editor for your coding.

Step 1. Copy, Examine and Modify the DSPF source of VNDINQ

__ 1. Make a copy of your DDS member, **VNDINQOV**, naming it **VNDINQPU**.

__ 2. In this member, enable F4 in the PROMPT_FMT, as you did with F3.

__ 3. Also, add the descriptive information, '**F4 = Display Vendor Detail**' to the right of the '**F3=Exit**' in your FKEYS_FMT.

__ 4. Using your student notebook to assist you, further modify your VNDINQPU DSPF to display a popup that looks similar to this:

```
***************Vendor Detail*******************
*                                               *
*   Vendor Name:   Federal Paper                *
*   MTD Purchased:    60,600.00                 *
*                                               *
*                                               *
*   Press F12 to Return                         *
*                                               *
***********************************************
```

You can make the border any character you like. Center the title as shown. The information shown on the format is available from the database record that you have already read. Allocate enough space to hold the data above.

If necessary, review the field definitions in the DICTIONARY Field Reference file. This member is in your QDDSSRC file, and, is also listed in the appendix.

Enable CA12 in the window and use it to return to the main application display.

__ 5. Compile the DSPF, **VNDINQPU** and make any necessary corrections.

Step 2. Copy and Modify your RPG Program VNRINQOV

__ 1. Make a copy of your RPG IV source member, **VNRINQOV**, naming it **VNRINQPU**.

__ 2. Modify the F-spec to reference your new file, **VNDINQPU**.

__ 3. In your D-specs, where the named indicators are mapped, add a new indicator, **Details** and map it to indicator number 4 as was done with the others.

__ 4. In your calculations, you need to test for the user's having pressed F4. This should be performed as part of processing a record that was found. When F4 is pressed, you should display the pop-up window. Be sure that you only do this for a valid vendor number.

__ 5. Compile your VNRINQPU program. Correct any compilation errors. Ask your instructor for assistance as needed.

__ 6. Test your pop-up window by calling your main program, **VNRINQPU**. Enter a valid vendor number; then, press **F4** to display details (MTD purchase data) for this vendor. Exit your window by pressing **F12**. Your original screen from **VNRINQPU** should be restored.

__ 7. Further test your pop-up window by entering an invalid vendor number. When you get the error message, reset the display and press F4. The pop-up should **not** be displayed. Enter a valid number and F4; your popup should display.

END OF LAB

Exercise 3. Array processing

What this exercise is about

This exercise provides an opportunity to work with an array.

What you should be able to do

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

- Define a compile-time array
- Store and access data in array elements

Introduction

In this exercise, you modify the display file and the program you wrote in the **OVERLAY**/**PUTOVR** exercise. Rather than displaying the date in edited numeric format, you display the character value of the month and edit the rest of the date value.

To do this, make changes to your display file as well as your RPG IV program.

Required materials

- Student Notebook
- Userid (**AS07nnn**) and password

Exercise instructions

Step 1 Understand the requirement

__ 1. Review the Display formats below. Notice the format of the date. The month is displayed as a character value. Trailing blanks of the value of month are removed. The day and year are separated by a comma and a space.

```
RJSLANEY                    Vendor Inquiry                          AECAUX
                                                 March 29, 2002     10:50:11
 Vendor number. . . . :   10050




 Please press enter to continue
  F3 = Exit
```

```
RJSLANEY                    Vendor Inquiry                          AECAUX
                                                 March 29, 2002     10:50:11
 Vendor number. . . . :   10050




 Name . . . . . :   Genessee Office Products
 Address  . . . . :   1313 Koday Rd.
                    Binghamton              NY  34506
 Telephone. . . . :    ( 816 )  555-1313
 Sales Person . . :   Antonio Souchak
 Purchases YTD  . :        50,000.00
 Balance Owed . . :        15,000.00









 Please press enter to continue
  F3 = Exit
```

Step 2 Copy, Modify and Compile the Display File

__ 1. Make a copy of your **VNDINQOV** display file from the first exercise and name it **VNDINQAR**.

__ 2. Modify your **VNDINQAR** display file by removing the **DATE** keyword that you included in your earlier solution.

__ 3. Define a new **18**-character field named **TODAY** that holds the formatted value of today's date supplied by your RPG IV program. You need to reposition the date field (**TODAY**) so that it does not overlay the time field.

__ 4. Compile your display file, **VNDINQAR**.

Step 3 Getting Values for the Date

__ 1. RPG IV supports some keywords that provide you with signed numeric values for day, month and year of the job date. The keywords are based on the ***DATE** (job date). The keywords to use are:

- ***DAY** - numeric value of day of job date
- ***MONTH** - numeric value of month of job date
- ***YEAR** - numeric value of year of job date

__ 2. These values are available to your RPG IV program. You do not need to define them to use them.

Step 4 Modify your RPG IV Program

__ 1. Make a copy of your solution program, **VNRINQOV**, from the first exercise and name it **VNRINQAR**.

__ 2. Modify your program, **VNRINQAR**, to use your new display file, **VNDINQAR**.

__ 3. Define a compile time array that holds the character values of each the twelve months.

__ 4. Enter the values of each of the twelve months in compile time format at the end of your program.

__ 5. Add the logic to format the job date in character format using the special keywords in the step above to extract the job date's values.

Hint: Use an expression to concatenate the three components into a single character value. Use BIFs to handle removal of unnecessary blanks and to ensure that all components of the expression are compatible data types.

Step 5 Compile and Test your Program

__ 1. Compile your program, **VNRINQAR**.

__ 2. Test values of the vendor number (**10001 - 10050**). Your date should be displayed in character format. If the date today is 03/29/2002, your date should display as **March 29, 2002**.

END OF LAB

Optional exercises

In this part of the exercise, you modify your date field to display a suffix to the day portion of the date. For example, 03-29-2002 should display as **March 29th, 2002**.

Step 1 Create display file VNDINQARE

__ 1. Make a copy of your display file and name it **VNDINQARE**.

__ 2. You need to make your date field **TODAY** two positions longer.

__ 3. Make sure that no fields overlap after having made this change.

__ 4. Compile your display file.

Step 2 Code and Compile VNRINQARE

__ 1. Make a copy of your program and name it **VNRINQARE**.

__ 2. Define a second array, **DAYSUFF**, that holds 20 2-character fields, **stndrdthththththththththththththththth**. These values cover the suffixes for the first 20 days. Days 21 through 31 can be handled using the first 10 elements of **DAYSUFF**.

__ 3. Add a new field to your D-specs, **INDEX**, that holds the calculated index to the **DAYSUFF** array.

__ 4. Use the following formula to calculate value of the index to the **DAYSUFF** array that will give you the suffix to be added to the day portion of the date:

Index = (Day/21) + (Remainder of (Day/21))

where Day is based on today's date (for example, May 23 is day 23).
However, this formula does not handle the 31st day of a month (Index = 11 when day is 31). Can you think of a simple solution?

Add the logic to include the suffix in the **TODAY** field on the display.

__ 5. Perform another test. Run the CHGJOB command and change the job date. Call your program again. The date should be the same as the one you entered and the date suffix should be correct for that date.

END OF LAB

Exercise 4. Data structures and data areas

What this exercise is about

This exercise provides an opportunity to create a data area and retrieve its contents from an RPG IV program.

What you should be able to do

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

- Create a data area
- Retrieve the contents of a data area in an RPG IV program
- Modify and rewrite the contents of a data area from an RPG IV program

Introduction

In this lab, you use your results from the “Array processing” lab. You modify the display file **VNDINQAR** to include our company name and tax rate. You modify the RPG program **VNRINQAR** to fetch the company name and tax rate from the data area.

Exercise instructions

Step 1. Display output

__ 1. Review this sample display. Your output should be similar to it. Notice the company name in the display.

```
                    Programming Supplies, International

RJSLANEY                    Vendor Inquiry                          AECAUX
                                              March 29, 2002        11:55:54

 Vendor number. . . . :   10050









 Please press enter to continue
  F3 = Exit
```

```
                    Programming Supplies, International

RJSLANEY                    Vendor Inquiry                          AECAUX
                                                 March 29, 2002     11:55:54

 Vendor number. . . . :   10050

 Name . . . . . . . :   Genessee Office Products
 Address  . . . . . :   1313 Koday Rd.
                       Binghamton              NY  34506
 Telephone. . . . . :   ( 816 )  555-1313
 Sales Person . . . :   Antonio Souchak
 Purchases YTD  . . :        50,000.00
 Balance Owed . . . :        15,000.00

 Tax Owed . . . . . :           975.00




 Please press enter to continue
  F3 = Exit
```

Step 2. Copy existing members

__ 1. Make copies of your previous DSPF (**VNDINQAR**) and RPG IV (**VNRINQAR**) member so that you can modify them in this lab. Name your new members **VNDINQDT** and **VNRINQDT,** respectively.

Step 3. Create a data area

Our company name was the **RPG Supply Company**. We are merging with another company named **Programming Supplies, International**. We need to change all headings on our reports and displays to reflect this new name. Management has asked that you write our company name in a data area so that we can access the company name without having to change and recompile programs in the future.

In addition, the VP of Finance has requested that we display the tax owed on any outstanding balance. In speaking with the VP, we learn that the current tax rate is **6.5%** and that the rate is expected to change sometime this year.

__ 1. Create a data area, COMPANY, of 50 characters in length and place Programming Supplies, International in it as well as the 6.5% tax rate. The tax rate should be specified as 065 in the first three positions of the data area.

We are making the data area 50 bytes long just in case corporate decides to place more information in it.

When you run **CRTDTAARA**, you need to use F4 to prompt. You should specify **&** in the CRTDTAARA **VALU**E keyword (initial value) when prompting in order to extend the entry area for complete input of the contents required.

Step 4. Modify your DSPF VNDINQDT

__ 1. Add the company name variable at the top of the existing display formats as shown in the sample. If it is necessary, modify line numbers to avoid potential overlap.

__ 2. Define a new field, **TAX (7,2)**, in the display file that holds the tax calculated based on the balance owed.

Step 5. Modify your RPG member VNRINQDT

__ 1. Use a data structure to hold the information from your data area and to separate the company name (character value) from the tax rate to be used in the program, which must be numeric to be valid.

__ 2. Modify it so that the company name variable is retrieved from your data area and is available to print at the top of the report.

__ 3. Also retrieve the value of the tax rate from the data area.

__ 4. Calculate the tax to be displayed as the product of **VNDBALANCE** and the tax rate stored in your data area.

Step 6. Compile and test

__ 1. If you have problems, check your compiler messages closely.

END OF LAB

Exercise 5. Inquiry subfiles

What this exercise is about

This exercise provides an opportunity to use inquiry subfiles.

What you should be able to do

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

- Create a display file that defines a subfile for inquiry
- Write an RPG program that uses a subfile for inquiry

Introduction

The purchasing agents in the RPG Office Supply Company want a simple display of all vendors. They need it today!

You are given screen design information and a description of the program logic.

Code and compile a subfile display file and an RPG IV program.

Use the example in your student notebook as a guide to coding your program.

Exercise instructions

Step 1. Analyze the requirements

__ 1. The purchasing department wants the vendors displayed in Vendor Name sequence. Use the file **Vndnam_LF** in your library for this exercise.

__ 2. Produce output similar to the following sample:

```
RJSLANEY                    Vendor Name Display                    AECAUX
                                                          8/22/03  13:49:15




Vend     Vendor Name
 No                            Telephone              Sales Person
10046 AAA Pencils, Etc.
10044 ABC Office Supply       316 345-6788           Ronald Mc Casky
10019 ACE Distributors        718 374-9728           Bob Quick
10036 Allen, Allen and Allen  701 645-3344           David Humphries
10006 Andy Glover Corp.       716 253-4580           Hayes M. Odem
10017 Best Furniture          205 600-7000           Carl Roberts
10031 Brand Names             404 261-1257           Janice Firestone
Electronics                   612 386-4318           Dave Treeman
10010 Clifford Distribution   702 722-4585           Roy Wayne
10040 Collier Office          616 458-6312           Colin Collier
Products                      213 889-0022           Connie Fusion
10034 Digital Analog          208 321-1000           Tucker Morgan
10033 Electronic Paper        301 658-4216           Kenneth Updike
10021 Ethan Albright Co.      914 256-3431           Davis Young
10025 Federal Paper           315 782-4618           Tom Wooley
10022 Fetzner & Fetzner
                                                                More...
  F3 = Exit
```

Step 2. Create display file VNDSUBFILE

__ 1. Code the DDS source that you use to create the display file.

__ 2. Name your source member **VNDSUBFILE** in file **AS07nnn/QDDSSRC**.

__ 3. Review the layout carefully and code it in logical formats using the techniques that have been discussed in class. You should design and code five formats for the

display headings, the function keys, the column headings for the subfile data, the subfile data record(s), and the subfile control record.

__ 4. Name the formats, **PROMPT_FMT, HEADER_FMT, FKEY_FMT, VSEARCHDTA, VSEARCHCTL**.

__ 5. Condition the **SFLEND** keyword using indicator 40. Set this indicator in your program when **EOF** of the database file in your program is encountered. If the **EOF** indicator is on for the data file, all records have been read, and the subfile has been filled and is ready to display.

__ 6. Use the documentation in the appendix for the vendor files **Vendor_PF** and the **Dictionary** file to get the information you need for the fields.

__ 7. Review the DDS for logical file **VNDNAM_LF**:

```
A                                       ALTSEQ(QSYSTRNTBL)
A           R VENDOR_FMT                PFILE(VENDOR_PF)
A           K VNDNAME
```

Step 3. Understand the program

Use the example in your student notebook to assist you in coding the RPG IV subfile program.

__ 1. As you read each record from the **VndNam_LF** file, write it to the subfile data record.

__ 2. Remember to increment the subfile RRN.

__ 3. Once you have reached **EOF** of the **VndNam_LF** file, write the subfile control record using *in03 (named indicator only) to control exit from your loop.

Step 4. Code and compile VNRSUBFILE

__ 1. Use the following conventions:

- Indicator 03: F3 key pressed.
- Indicator 40: Set when End of file VNDNAM_LF reached (use %eof BIF) or subfile **full**.
- Field for subfile relative record number: RRN.

Step 5. Test and debug VNRSUBFILE

__ 1. Compile your program. When it compiles successfully, call it and match your output to the sample at the beginning of this exercise.

__ 2. Next, test your program using an empty copy of the VENDOR_PF file as described in the steps that follow.

__ 3. You will need an empty VENDOR_PF to test.

__ a. Copy your VENDOR_PF file to another file named VENDOR_PFD using this command:

```
CRTDUPOBJ OBJ(VENDOR_PF)
          FROMLIB(AS07nnn)
          OBJTYPE(*FILE)
          NEWOBJ(VENDOR_PFD)
          DATA(*YES)
```

__ b. Clear the data from your VENDOR_PF using the following command:

```
CLRPFM FILE(AS07nnn/vendor_pf)
```

__ 4. Call your program. Do you get an error? Can you explain why the error occurred?

__ 5. Save a copy of your work so far. Make a new copy of the DDS (**VNDSUBNR**) and the RPG program (**VNRSUBNR**) if you decide to modify both. Modify your program to handle the situation where the subfile is empty and thus avoid the error. What in your program would tell you that the subfile is empty? For now, simply issue a message to your message queue that states, `No records to display`.

__ 6. Compile and test your modified program.

__ 7. When your program is working, restore the data to your copy of Vendor_PF using the following command:

```
CPYF FROMFILE(AS07nnn/VENDOR_PFD)
     TOFILE(AS07nnn/VENDOR_PF)
     MBROPT(*REPLACE)
```

END OF LAB

Exercise 6. Inquiry subfile with search

What this exercise is about

This exercise provides a second opportunity to use an inquiry subfile.

What you should be able to do

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

- Create a display file that defines a subfile for inquiry with a search argument
- Write an RPG program that uses a subfile for inquiry using a search argument to position the file cursor

Introduction

The purchasing agents in the RPG Office Supply Company liked the work you did in the previous exercise. They want you to enhance the program by allowing them to search using a partial vendor name. For example, if a `C` is entered by the user as a search argument, your program should load the subfile and then display it beginning with the first vendor name beginning with a `C, Clifford Distribution`.

You are provided with information to assist you to design the display file formats and with a description of the program logic.

You can use, modify, and enhance the display file and RPG IV program that you coded in the previous exercise if you prefer. Do not be concerned if the test against an empty data file did not work. You can use the version of the program you saved before you tried this test.

Exercise instructions

Step 1. Analyze the requirements

__ 1. Modify the DDS and the RPG program from the previous exercise to prompt for the vendor name and enable F3 key as the exit option. Review the sample that follows. First, you can see that we now prompt for a search key (partial vendor name):

```
RJSLANEY                    Vendor Name Search                    AECAUX
                                                          8/25/03 09:29:16
Enter partial vendor name: _________________________
Press enter to continue




  F3 = Exit
```

__ 2. The program should use the partial vendor name to set the file cursor and to load the subfile. Display those vendors starting at the first one that satisfies the search argument.

```
RJSLANEY                    Vendor Name Display                    AECAUX
                                                          8/22/03  13:49:15

Enter partial vendor name: _____________________
Press enter to continue

Vend     Vendor Name           Telephone           Sales Person
 No
10046 AAA Pencils, Etc.        316 345-6788        Ronald Mc Casky
10044 ABC Office Supply        718 374-9728        Bob Quick
10019 ACE Distributors         701 645-3344        David Humphries
10036 Allen, Allen and Allen   716 253-4580        Hayes M. Odem
10006 Andy Glover Corp.        205 600-7000        Carl Roberts
10017 Best Furniture           404 261-1257        Janice Firestone
10031 Brand Names Electronics  612 386-4318        Dave Treeman
10010 Clifford Distribution    702 722-4585        Roy Wayne
10040 Collier Office Products  616 458-6312        Colin Collier
10034 Digital Analog           213 889-0022        Connie Fusion
10033 Electronic Paper         208 321-1000        Tucker Morgan
10021 Ethan Albright Co.       301 658-4216        Kenneth Updike
10025 Federal Paper            914 256-3431        Davis Young
10022 Fetzner & Fetzner        315 782-4618        Tom Wooley
                                                              More...
  F3 = Exit
```

Step 2. Create display file VNDSEARCH

This display file is very similar to the DDS for the display file that you coded in the previous exercise.

__ 1. Copy the DDS that you coded in the previous exercise. Name your copied source member **VNDSEARCH** in file **AS07nnn/QDDSSRC**.

__ 2. Modify the DDS you copied from your previous display file to create the display file, **VNDSEARCH**.

__ 3. Add a search argument. Include an error message `No vendors found` to be displayed when indicator 96 is set by your program.

__ 4. As before, name your formats: **FKEY_FMT, VSEARCHCTL, VSEARCHDTA, PROMPT_FMT,** and **HEADER_FMT**.

__ 5. Use the following indicators in your DDS:

- Indicator 40: SFLEND
- Indicator 75: SFLCLR

- Indicator 85: SFLDSPCTL
- Indicator 95: SFLDSP
- Indicator 96: Conditions the error message `No vendors found`.

__ 6. Use the documentation in the appendix for the vendor files **VndNam_LF** and **Vendor_PF** and the **Dictionary** file to obtain any additional information you need for naming any fields.

Step 3. Create VNRSEARCH

__ 1. Make a copy of your **VNRSUBFILE** program from your previous exercise and name it **VNRSEARCH**.

__ 2. Use the following indicators in the RPG program in addition to those required by the subfile control record (to manage the subfile - Indicators 40, 75, 85 and 95). Map the numbered indicators to named conventions:

- Indicator 03: F3 key pressed.
- Indicator 96: Error - SETLL set file cursor beyond EOF.

__ 3. Continue using the field for subfile relative record number: **RRN**.

__ 4. This program is different from the inquiry subfile program you wrote. Use your lecture notebook example program as a reference while you write this program.

Add the logic to process the search key against the **VndNam_LF** file.

__ 5. Compile and test your program **VNRSEARCH**.

Step 4. Test program VNRSEARCH

__ 1. Call **VNRSEARCH**.

__ 2. Enter a partial name and note the results.

__ 3. Try to page up from the top of the subfile. It should not work because you are trying to page above the first record of the existing subfile.

__ 4. Do not enter a new search argument. Simply press Enter. You should see the first record in the logical file.

Step 5. Recognizing upper and lower case

__ 1. Review the DDS for logical file VNDNAM_LF that follows:

```
A                                     ALTSEQ(QSYSTRNTBL)
A          R VENDOR_FMT               PFILE(VENDOR_PF)
A          K VNDNAME
```

__ 2. The alternating collating sequence keyword (ALTSEQ) above specifies that QSYSTRNTBL, the system-provided translate table, translates lower case to upper case. By doing this, the user's entry for the search argument is not case sensitive when searching the DB file.

END OF LAB

Exercise 7. Modularize vendor subfile search

What this exercise is about

This lab is the first of a series of five exercises where you make a series of enhancements to your subfile search program.

What you should be able to do

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

- Change a program that is written using inline logic to use subroutines

Introduction

In this exercise, you make your vendor subfile search program that you wrote in the previous exercise modular by changing the code to use subroutines.

Make a copy of the DDS and RPG IV program from the previous exercise and modify them.

Exercise instructions

Step 1. Make copies of your existing DDS

__ 1. Make sure that you successfully completed the previous exercise. If you were not successful, ask your instructor for assistance.

__ 2. Make a copy of your DDS, **VndSearch**, naming it **VndSchS1**.

Step 2. Modify your copy of the DDS, VndSchS1

__ 1. Make the following change to your copy of the DDS:

__ a. Rather than using the ERRMSG keyword and indicator for an error in the search argument, add a separate record format named MSG. Delete the ERRMSG coding.

__ b. Define a constant with a value of `No vendors found`. This record format is displayed when the search argument positions the file cursor at end of file.

__ c. Code the message to display high intensity in row 12, beginning in position 32.

__ 2. Compile your copy of the DDS, **VndSchS1**.

Step 3. Code your RPG IV program, VnrSchS1

__ 1. Using the lecture sample program as a guide, and your previous program **VnrSearch**, code your **VnrSchS1** program to use subroutines to perform all functions. Add a stand-alone indicator, EmptySfl, which is set when there are no subfile records to display (as in the lecture example).

__ 2. Because we use this program as a basis for subsequent exercises, name your subroutines as follows:

- ***InzSR** - performs initial processing
- **SearchRtn** - positions file cursor and drives all other subroutines (except *InzSR)
- **FilI** - fills subfile based on Search
- **Prompt** - displays subfile and prompts for a new partial search code
- **SflClear** - clears existing subfile content before filling with new records based on search code

__ 3. Your mainline performs the **SearchRtn** subroutine as long as F3 is not pressed.

Step 4. Compile and test your program, VnrSchS1

__ 1. When your program object has been created, test using various search codes. The program should perform exactly as your **VnrSearch** program did.

__ 2. Because SFLSIZ > SFLPAG in the DSPF, the system handles scrolling automatically. But, you can only scroll up to the top of the subfile. The record displayed at the *top of the subfile* varies depending on the value you enter for the **SEARCH** (search argument) field. In a later exercise, you learn how to scroll above the *top* and are not limited by the value of the **SEARCH** field.

END OF LAB

Exercise 8. Page + 1 and Pagedown

What this exercise is about

This lab gives you the opportunity to modify your previous lab program and DDS to implement **Page + 1** processing and to code a **PageDown** subroutine.

What you should be able to do

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

- Implement **Page + 1** techniques
- Code a subroutine that performs **PageDown** processing

Introduction

In this lab exercise, you modify your program and DDS from the previous lab to add **Page + 1** and **PageDown** processing.

Exercise instructions

Step 1. Make copies of your previous exercise DDS and program

__ 1. Make copies of your DDS, **VndSchS1,** and your RPG IV program, **VnrSchS1**, from your previous exercise, naming the copies **VndSchS2** and **VnrSchS2,** respectively.

Step 2. Modify your DDS, VndSchS2

__ 1. Change your DDS, adding **PAGEDOWN** support. Think about where you should specify this keyword. You need to assign an indicator that is set when the **PAGEDOWN** key is pressed. This keyword needs to be associated with the indicator used with the **SFLEND** keyword.

__ 2. Add and define a hidden field **SFLRRN(4,0)** to the **Subfile Control** format that is associated with the **SFLRCDNBR** keyword. **SFLRCDNBR** specifies that the page of the subfile to be displayed is the page containing the record whose relative record number is in **SFLRCDNBR**.

__ 3. Add a variable **SFLSIZE(5,0)** to the **Subfile Control** format that will be used in the program to set the subfile size. This field is known as a **program-to-system** field. Specify this usage type in your DDS. Remember to add a parameter to the **SFLSIZ** keyword to reference this variable which is set in the RPG IV program.

__ 4. Create your display file, **VndSchS2**. Correct any errors.

Step 3. Modify your RPG IV program, VnrSchS2

__ 1. Refer to the lecture notes. Using the **ITEMSUBS2** program as a guide, change your program to support **Page + 1** and **PageDown**. Some key points to consider follow:

   __ a. Add an named indicator, **PageDown**, mapped to the indicator you used with the PAGEDOWN keyword in your DDS.

   __ b. Add another variable, **RrnCount**, to keep track of subfile records. We have been using **Rrn** to track the number of records read from the **VndNam_LF** file.

   __ c. Your mainline changes. Use a **Select group** to check for **NextPage** being pressed and to perform the **SearchRtn** subroutine (as your mainline did before).

   __ d. Modify your **Fill** subroutine to initialize **RrnCount** before filling the subfile. **RrnCount** counts the number of subfile records loaded.

   __ e. Modify the Do loop condition of your **Fill** subroutine to add a check for end of file for the **SflPag having** been filled. Use your **RrnCount** variable.

   __ f. In the **Fill** subroutine, increment **RrnCount** in the loop as you write records to fill the subfile.

   __ g. Also in the **Fill** subroutine, add logic after the loop to set your indicators whether you have loaded any records in the subfile. Set your **SflEnd** indicator and your **EmptySfl** indicator.

__ h. Initialize the variable, **SflSize**, defined in the DSPF in your ***InzSR** subroutine. It should be set to a value of your **SflPag + 1**.

__ i. Write a subroutine to perform **PageDown**. Name the subroutine **NextPage**. When the user presses **PageDown**, you should fill and display the subfile. Make sure that the user is prompted for a new search code.

Step 4. Create and test your RPG IV program, VnrSchS2

__ 1. Test your program as you have before and compare it to your **VnrSchS1** program. There should be no difference in the behavior of the two programs.

END OF LAB

Exercise 9. Add PageUp

What this exercise is about

This lab gives you the opportunity to add **PageUp** processing to your program.

What you should be able to do

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

- Add **PageUp** capability to a display file
- Write the subroutine to enable **PageUp** in RPG IV

Introduction

Copy your DDS and RPG IV program and then modify them to support **PageUp** processing.

Exercise instructions

Step 1. Make copies of your DDS and RPG IV source

__ 1. Copy **VndSchS2** and **VnrSchS2** and name them **VndSchS3** and **VnrSchS3,** respectively.

Step 2. Modify your DDS, VndSchS3

__ 1. Change your DDS, adding **PageUp** support. You need to assign an indicator that is set when the **PageUp** key is pressed. This keyword needs to be associated with an indicator used to indicate beginning of file (set in your program).

__ 2. Create your display file once you are finished making this change.

Step 3. Modify your RPG IV program, VnrSchS3

__ 1. Use your lecture notes and pattern your changes after the program in your notes.

__ 2. Following are the key features that you add. Note that not everything is documented. Review the lecture example carefully:

- __ a. Define indicators to check for the **PageUp** key having been pressed and to check for the top of the subfile.
- __ b. Initialize the indicator that you defined to be set when you reach the beginning of the subfile.
- __ c. In your mainline, add the code to check whether **PageUp** was pressed and to exit to a subroutine to handle it. Name the subroutine **PrevPage**.
- __ d. Add code in the **SearchRtn** subroutine to check whether you have reached BOF of the data file.
- __ e. Remember that you need to manage the indicator you assign for the beginning of the subfile that works with the **PageUp** key (**PageUp** is enabled when you execute the **Prompt** subroutine, but should be disabled when you execute the initialization subroutine). In the **Prompt** subroutine, **PageUp** should be disabled when there are no records to display in the subfile.
- __ f. Code the **PrevPage** subroutine. You can write it the way we did in lecture, using the first record in the current subfile to position the cursor in the **VndNam_LF** file and then use **ReadP** to read backwards. Check whether you have reached BOF of the DB file by executing a subroutine named **CheckBOF**.
- __ g. Code a subroutine named **CheckBOF** check if we are at the beginning of the DB file. Use a READP to see whether we reach the beginning of file. If we are at BOF, **CheckBOF** then positions the file cursor at the first record of the DB file and then reads that record.

Step 4. Compile and test your RPG IV program, VnrSchS3

__ 1. Test your program as you did before. Also press the **PageUp** key. Test its function by positioning the subfile using a search key of **F**. Press the **PageUp** key. Vendor names with names earlier in sequence than **F** should be displayed. Keep pressing the PageUp key until you have reach the top of the file. The first vendor is **AAA Pencils, Etc**.

END OF LAB

Exercise 10. Add SFLPAG = SFLSIZ

What this exercise is about

This lab gives you the opportunity to implement **SFLPAG = SFLSIZ** processing to your program.

What you should be able to do

At the end of the lab, you should be able to add the logic necessary to implement **SFLPAG = SFLSIZ.**

Introduction

Copy your DDS and RPG IV program and then modify them to support **SFLPAG = SFLSIZ** processing.

Exercise instructions

Step 1. Make copies of your DDS and RPG IV program

__ 1. Copy **VndSchS3** and **VnrSchS3** and name them **VndSchS4** and **VnrSchS4,** respectively.

__ 2. Your DDS is not changed as you already have a variable defined (**&SFLSIZE**) to get the value of **SLFSIZ** set by RPG IV program.

Step 2. Modify your RPG IV program, VnrSchS4

The main change in the program is that now you manage all scrolling in your program.

__ 1. In the initialization subroutine, set the value of the RPG IV variable **SflSize** equal to the value of **SflPag** as specified in your DDS.

__ 2. Manage **SflRrn** (associated with the **SFLRCDNBR** keyword) and initialize it in your initialization subroutine.

__ 3. Because **SflSiz =SflPag**, check all your conditions that check **SflSize** to make sure that they reflect this.

__ 4. The **NextPage** subroutine must be changed to reset the value of **Rrn** (counter for data file records read), and the subfile should be cleared.

__ 5. In the SflClear subroutine, reset the **SflBegin** indicator.

Step 3. Compile and test

__ 1. When you have compiled your program, test various search keys and check that scrolling works as it did before. The program should perform the same as your **VnrSchS3** program.

Note: Remember, you are now managing the subfile completely. The system no longer automatically extends the size of the subfile which you have fixed at 14 records.

END OF LAB

Exercise 11. Add maintenance

What this exercise is about

In the lab, you make final modifications to your DDS and RPG IV program by adding the capability to perform additions, changes and deletions to the database file by using the subfile to select the record.

What you should be able to do

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

- Modify DDS to enable maintenance
- Modify an RPG IV program to perform file maintenance using modular techniques

Introduction

You can use any of the subfile search exercises as a basis for adding maintenance. You might find it easiest to use the first search exercise as it does not have all the formats broken up into smaller formats.

In this exercise, you change the DDS and the program from the previous exercise. You add an input field in the subfile record that is used to indicate what type of transaction the user wants to perform. You do not write all the code to perform the specific transaction. As we did in the lecture example, you code calls to programs that perform the specific maintenance tasks.

Exercise instructions

Step 1. Make copies of your source members

Note: Maintenance can be added to your most recent working subfile search program. You can use any of your earlier solutions as a basis. Select a working solution to Exercises 7 through 10 as the basis for this exercise. (**VnrSchS1** through **VnrSchS4** may be used).

Whatever you decide, you should use an already working solution as a basis.

__ 1. Copy the working DDS source member and the corresponding RPG IV source member of your choice, naming your copies **VndSchS5** and **VnrSchS5**, respectively.

Step 2. Modify your DDS, VndSchS5

__ 1. Add a field named **Option** in position 2 of your subfile record format. Only accept values of **1**, **2** and **4**. This field should be one character in length. When you do this, shift all other fields right two positions to avoid overlaying fields.

__ 2. Add a heading **Opt** starting in position 1 of the heading line of your subfile control record format. When you do this, shift all other fields right three positions to avoid overlaying fields.

__ 3. Change your **Msg** record format. Delete the line that displays the constant message *No vendors found* and add a line to display a field named **Message** that is 25 characters in size in the same position. This field is set by the program.

__ 4. Move the prompting fields from your **PROMPT_FMT** and the heading fields from your **HEADER_FMT** for the subfile to the subfile control record format. Comment out or delete these two record format specifications as they are not used in this maintenance application. This is the same redesign that we discussed in the lecture.

__ 5. Compile and create your display file. Correct any errors.

Step 3. Modify your RPG IV member, VnrSchS5

Use your lecture sample program to guide you on the changes to make.

__ 1. Make changes to your program to **Write** or **Exfmt** the subfile control record rather than the **HEADER_FMT** and **PROMPT_FMT** record formats as we discussed in lecture. Consider the setting of the subfile display control indicator before you perform the I/O to the subfile control record.

__ 2. In your **Prompt** subroutine, add an **Exsr** to your new **Changes** subroutine that you write. You should only execute the **Changes** subroutine if there are records in the subfile to display. The **Changes** subroutine look at the value of **Option** and then execute the appropriate subroutine. As well, you need to assign the value of `No Vendors to Display` to the **Message** field in the appropriate location before you write the **Msg** record format.

__ 3. Write the **Changes** subroutine.

__ a. The first thing you must do is to read the subfile record in which the user entered the value of **Option**. Use a **ReadC** to do this. ReadC reads any changed subfile records. In our case, we should only be concerned about single changes.

__ b. Code a **Select** group that checks the values of the **Option** field and then executes the appropriate code. You can code inline logic as we did in class or you can code your transactions in subroutines.

__ c. Code your calls to the various maintenance programs exactly as we did in the lecture program. When you want to change or delete a record, you modify the physical file, **Vendor_PF** using the logical view, so you have to pass a parameter, **VndNbr**. We discuss calls further in a later unit. But, for now, just copy how the calls are coded in the lecture example.

__ d. As you review the lecture program, notice that the each routine of the maintenance logic that we discussed in lecture is very similar to the others. You can code one routine and then copy and modify it to handle the other maintenance options in order to reduce keystrokes.

__ e. When the value of the **Option** field is **1**, for example, you should set the **SflDsp** and **SflDspCtl** indicators to permit the **Msg** to be displayed rather than the subfile records.

Next, call the program named **ADDPROGRAM** using the same syntax as we showed in lecture. This type of call is known as a *dynamic call*. The program name must be enclosed in quotes and coded in uppercase as it is a literal value. The **CALL** opcode is not supported in free format. We spend time discussing the prototyped call using the CALLP opcode that is supported by free format.

For the add transaction, we do not need to pass any parameters. The **ADDPROGRAM** prompts the user for the new Vendor Number to be added. Remember, you are working with the **Vendor_PF** physical file in this program.

__ f. The **ADDPROGRAM** might not have been written yet and might not exist. Add the error handling using the **E** opcode extender and use the **%error** BIF as we did in the lecture example. We discuss error handling further later.

__ g. Add the logic to call the **CHGPROGRAM**. Again, use the lecture example as a guide. In this case, we need to pass a parameter, the key to the **Vendor_PF** physical file. When we perform the **ReadC** opcode, we not only read the **Option** field, but all the other fields in the subfile record, including the record key.

The **CHGPROGRAM** may not been written yet and may not exist. Add the error handling using the **E** opcode extender and use the **%error** BIF as we did in the lecture example.

__ h. Add the logic to call the **DLTPROGRAM** as well, patterning your code after what we did in lecture. Pass the record key as a parameter to the **DLTPROGRAM**.

__ i. The **DLTPROGRAM** has been written and a ***PGM** object exists in the course master library. Include the error handling using the **E** opcode extender and use the **%error** BIF as we did in the lecture example.

__ j. Now that you have processed the transactions, code a **ReadC** just before the **EndDo**. We are in a **DoW** loop testing for EOF. The ReadC read s the next changed subfile record or set an EOF condition. This is exactly like a normal read loop. If EOF is encountered, the subfile is displayed and repositioned if the user has changed the search key.

Step 4. Compile and test your program

__ 1. When your program has compiled, test it. The **Option** field should now be available as an input field.

__ 2. Test different values.

__ 3. For a **1**, the `Calling Add Program` message is displayed.

__ 4. For a **2**, the `Calling Change Program` message should be displayed.

__ 5. For a **4**, the `Calling Delete Program` message should be displayed. When you press Enter, you should see a window that tells you that the record you selected was deleted. When you press Enter, the subfile is displayed. Notice that the record that you deleted is no longer in the subfile. To refresh your Vendor_PF file, copy the data from the copy you made in the earlier exercise as follows:

```
CPYF FROMFILE(AS07XXX/VENDOR_PF)
     TOFILE(AS07nnn/VENDOR_PF)
     MBROPT(*REPLACE)
```

__ 6. Any other value of **Option** is an error. This should be handled in your display file DDS.

__ 7. Use the debugger to help you to resolve any problems during testing.

END OF LAB

Exercise 12. Error-handling BIFs

What this exercise is about

This lab covers using Built-in Functions to avoid exceptions. You also compare the difference in behavior between avoiding an exception and the default error handler.

What you should be able to do

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

- Explain how the program behaves without BIFs using the default error handler
- Code error handling in RPG IV programs to manage errors and terminate programs gracefully

Introduction

You work with a basic file maintenance program and modify it to study how the default error handler behaves compared to coding Built-in-Functions. You might want to integrate this program with the subfile maintenance program you completed in an earlier exercise.

Exercise instructions

Step 1. Understand the task

__ 1. In your library is a DDS member, **VndAdd**. Review this member, shown as follows. This member uses DDS windows to prompt for the vendor number to add to the **Vendor_PF** file:

```
A                                      REF(VENDOR_PF)
A          R ADDWIN
A                                      WINDOW(07 4 10 70)
A                                      WDWTITLE((*TEXT 'Add Program') +
A                                      (*COLOR BLU) (*DSPATR RI) +
A                                      *LEFT *TOP)
A                                      WDWBORDER((*COLOR BLU))
A                                  3  2'Enter vendor number:'
A            VNDNBR    R     D  I  3 31

A          R MSGWIN
A                                      WINDOW(07 4 10 70)
A                                      WDWTITLE((*TEXT 'Add Program') +
A                                      (*COLOR BLU) (*DSPATR RI) +
A                                      *LEFT *TOP)
A                                      WDWBORDER((*COLOR BLU))
A            MESSAGE       50       3  2

 ** Dummy format to prevent display clearing
A          R DUMMY
A                                      ASSUME
A                                  2  4' '
```

__ 2. Compile the **VndAdd** DDS source to create a new DSPF.

__ 3. You are also given an existing RPG IV program, **ADDPROGRAM**. This program exists in your library. A copy is shown as follows. Notice that the program simply prompts for a vendor number and writes it to the **Vendor_PF** file. No error or exception handling is performed:

```
FVendor_PF UF A E           K Disk
FVndAdd    CF   E             Workstn

 /FREE
  ExFmt AddWin; // Prompt for the vendor number to add
  Write Vendor_Fmt;
  Message = 'Vendor number ' + %char(VndNbr) +
            ' added successfully';

  ExFmt MsgWin;
  *inLR = *on;
 /End-free
```

__ 4. Compile your copy of **ADDPROGRAM**.

__ 5. Test **ADDPROGRAM** by calling it from the command line. You see a window with a prompt for a vendor number to add. Enter vendor number **10099**. Press Enter. You

should see a window with a message that the vendor was successfully added. Press Enter.

You can confirm the record was added to **Vendor_PF** by running the following command:

```
RUNQRY *N *CURLIB/VENDOR_PF
```

Additional: If you completed the subfile maintenance program, you can also test your **ADDPROGRAM** by calling your **VnrSchS5** program. In the **Option** field, enter a '1' to add a new record **(10098)**. You see the window with the message `Calling Add Program`. Press Enter. You should now be prompted as shown previously. When you have added another new record, your subfile should now display the new vendor in addition to all the existing ones. Notice that the new vendor is first in the list as it has blanks as the vendor name. We only added a vendor number and no other data to the record.

__ 6. Add the same vendor number again **(10099)**. You should receive a message from the default error handler regarding your attempt to add a duplicate key:

```
RNQ1021 - Attempt to write a duplicate record to file VENDOR_PF
```

Step 2. Modify the AddProgram

__ 1. Now, enhance your copy of the **ADDPROGRAM** by including logic that anticipates and avoids the duplicate key error. Use opcodes and BIFs that will avoid the error being handled by the **Default Error Handler**. If the error occurs, use the **Message** field to inform the user that an attempt was made to add a vendor number that was already on file. For example:

```
Error - Vendor (10099) already exists in file
```

__ 2. This message can be displayed using the **MSGWIN** format of the **VndAdd DSPF**.

__ 3. When you have modified your program, compile and test it as you did before. This time, you should not receive an exception message from the default error handler. The error should be handled in your program.

__ 4. Restore your copy of the **Vendor_PF** file by copying the data from **AS07XXX/Vendor_PF** file to **Vendor_PF** in your library. (Remember to replace the existing data.)

END OF LAB

Exercise 13. Using monitor groups

What this exercise is about

This lab lets you test monitor groups and compare them.

What you should be able to do

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

- Code monitor groups
- Contrast the use of monitor groups to % error and **E** extender

Introduction

Modify your subfile maintenance program again and experiment with monitor groups.

A second program is provided to give you more opportunity to try monitor groups.

Exercise instructions

Step 1 Copy your program, VnrSchSErr

__ 1. Make a copy of your program, **VnrSchS5**. Name it **VnrSchSMon**.

If you did not complete the subfile maintenance program **VnrSchS5** in the earlier exercises, you can copy the sample solution from the course library:

Display File (DDS): **AS07V4LIB/QDDSSRC(VNDSCHS5S)** copy to, **AS07nnn/QDDSSRC(VNDSCHS5S)**

RPG IV Program: **AS07V4LIB/QRPGLESRC(VNRSCHS5S)** copy to, **AS07nnn/QRPGLESRC(VNRSCHSMON)**

Compile the new source members and call your **VnrSchSMon** program to verify it functions correctly.

Step 2 Modify VnrSchSMon

__ 1. Define a field named **ErrorMsg** with a length of 30 characters.

__ 2. Find the logic in **VnrSchMon** that processes a value of **2** for the **Option** field (Change transaction).

__ 3. Remove the existing error handling in this portion of code (E-extender/%error) and replace it with a monitor group. Handle the program not found error by issuing a message to your message queue using the **ErrorMsg** field. This field should contain the text `Change Program not found ssss` where *ssss* is the status code (use your reference manual to determine the code).

Step 3 Compile and Test your Program, VnrSchSMon

__ 1. Enter a value of *1* for the **Option** field as you did before. The add should work as it did in the previous exercise.

__ 2. Enter a value of *2* for the **Option** field. End the program and examine your message queue. What happened? Is there any difference in behavior when using a Monitor Group versus the E-extender/%error method? _____

_____

_____

__ 3. Restore the data in your **Vendor_PF** file if necessary.

Step 4 Try anotherp program

__ 1. In your library, you find a DDS source member for a display file named **LOANPAYD**. Review it and compile it.

__ 2. You also find an RPG IV source member named **LOANPAYLP**. Browse your copy. This program calculates the effective periodic interest rate and payment for a loan amount and loan term that you input.

__ 3. Compile your source, **LOANPAYLP**, using option 14 of PDM to execute the CRTBNDRPG command.

__ 4. When you have successfully created the program, **LOANPAYLP**, test it by calling it.

Use any additional test data you want, but enter these values initially:

100000.00 for loan amount

7.25 for annual interest

12 payments per year

360 as the number of payments

You will see the result shown below:

```
9/05/03                 Loan Payment Calculator          System:  AECAUX
13:31:09                                                 User:    RJSLANEY



Type values, press Enter.

               Loan amount . . . . . .   100,000.00

               Annual interest % . . . . . .  7.250

               Payments per year . . . . . . . 12

               Number of payments  . . . . . .   360

               Periodic interest . . . . . .   .00604166666

               Periodic payment amount . . . .           682.17




F3=Exit
```

__ 5. Next, leave all fields as they are except the annual interest rate field. Enter zeros for the interest rate. Press Enter.

__ 6. What error message do you see? _____

__ 7. Respond to the error message with the 'D' option (dump) to produce a program dump listing.

__ 8. Modify your source code to monitor for this error. How is this error code supported in RPG IV? _____ (***Hint:*** To determine the Status Code, review the dump listing you produced in the previous step)

__ 9. Define the field, **ErrMsg**, in your DDS source as a 40-character field. Place the error anywhere below the existing information on the display.

__ 10. Modify your RPG IV source to assign a value to the field, **Message**, that is to be displayed when the error occurs. The RPG program should assign the following value to the **ErrMsg** field:

```
Incorrect value for Annual Interest
```

Allow the program to continue.

Step 5 Test your modified program

__ 1. Test your program using the same test as you did above. Do you see the message? Did your program allow you to continue? _____

Step 6 Implementing monitor groups

__ 1. Now that your programs are working, please think about how monitor groups could be implemented throughout these programs. Should they be coded specifically for one logical section of code or should they be generic and should the whole program be one monitor group?

_____

_____

__ 2. We only tried a single I/O exception in the first program. We could use an E-extender and %error to handle the error as you know. Can you suggest where you might use monitor groups where no error handling can be easily implemented?

_____

_____

_____

_____

_____

_____

__ 3. The instructor leads a class discussion about the different methods of error handling. Be ready to contribute your ideas.

END OF LAB

Exercise 14. Using dates

What this exercise is about

This exercise provides an opportunity to experiment with date BIFs, such as **%Date**, **%Diff**, and **%SubDt**.

What you should be able to do

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

- Use the **%Date** BIF to convert from a character to a date data type
- Use the **%SubDt** BIF to extract the year, month, and day portion from a date
- Use the **%Diff** BIF to determine the duration between two dates expressed in days, months, or years

Introduction

You are given the DSPF DDS and you write a date program that uses date processing BIFs.

Exercise instructions

Step 1. Review the DDS, DATEDSPF

__ 1. In your library, you find the DDS for the display file. A copy follows:

```
A                                      REF(*LIBL/ITEM_PF)
A                                      INDARA
A                                      CA03(03 'Exit')
 **
A          R HEADER
A                                  3 35'Date Exercise' COLOR(WHT)
A                                  3 55'Today''s Date'
A            TODAY           L  O  3 70DATFMT(*ISO)
 **
A          R PROMPT                    OVERLAY
A                                  8 10'Enter any date as YYYY-MM-DD:'
A            CHARDATE      10A  B  8 45
A  40                                  ERRMSG('Invalid date entered' 40)
 **
A          R DETAIL                    OVERLAY
A                                 10 20'The year entered was. .:'
A            YEAR           4  0O 10 46EDTCDE(L)
A                                 11 20'The month entered was .:'
A            MONTH          2  0O 11 48EDTCDE(L)
A                                 12 20'The day entered was . .:'
A            DAY            2  0O 12 48EDTCDE(L)
A                                 14 20'The date entered is . .:'
A            DAYS           5  0O 14 45EDTCDE(Z)
A  50                             14 51'days from now'
A N50                             14 51'days ago'
 **
A          R FOOTER                    OVERLAY
A                                 20  7'Press Enter to continue'
A                                 21  7'F3=Exit'
A                                      COLOR(BLU)
```

__ 2. Notice that the fields to be entered and displayed are defined in the DDS.

__ 3. Compile and create the display file, **DATEDSPF**.

Step 2. Study the display output

__ 1. Review this sample output at the top of the next page:

```
                          Date Exercise      Today's Date   2003-09-10



    Enter any date as YYYY-MM-DD:      2003-12-19

              The year entered was. .:  2003
              The month entered was .:    12
              The day entered was . .:    19

              The date entered is . .:   100 days from now




  Press Enter to continue
  F3=Exit
```

__ 2. The user can enter a date that, when it is valid, is separated into the year, month, and day components. Also, the duration between the date entered and the job date is calculated in days and displayed as in the future *days from now* or in the past *days ago*.

__ 3. If the date is invalid the message *Invalid date entered* is displayed.

Step 3. Write the program, DATERPG

__ 1. Use named indicators.

__ 2. The program should determine today's date based on the job date and display it in the field **Today**.

__ 3. When the user enters a value for the field **CharDate**, check whether it is valid. If it is not, you should turn on the named indicator that equates to indicator 40. If the date is valid, you should extract the year, month, and days components, and assign the values to the appropriate display file fields.

__ 4. Determine the difference in days between the date entered and the job date. If the date is in the future, you should set the indicator appropriately (see your display file). Also handle the opposite situation.

Step 4. Compile and test DATERPG

__ 1. Compile your program and correct and errors.

__ 2. When you have created a *PGM, test it using various valid as well as invalid dates. If you get a size error for the field **Days**, you can make it larger in the DSPF or try a different value for **Days**. **Days** in the DSPF that you are given is five digits in size.

END OF LAB

Exercise 15. Prototyping

What this exercise is about

This lab covers practice writing a program that you call using prototyping.

What you should be able to do

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

- Code a program that can be called from another program
- Code a procedure interface (PI) in the called program
- Code a prototype (PR) in the calling program
- Use the CALLP operation code

Introduction

In this exercise, you modify your subfile maintenance program so that it uses **CALLP** and a **prototype** to call one of the maintenance programs that you also write, including the procedure interface.

Exercise instructions

In this exercise, you modify a program that you have already written, **VNRSCHSMON**. You recall that there are **CALLs** (dynamic calls) to several programs that can add a new record, change an existing record, or delete an existing record from the **Vendor_PF** file. We are using a logical view by vendor name in the subfile maintenance program. At this point the called programs do not really exist.

Write the program that deletes a record based on an **option code of 4**. As you recall, the subfile maintenance program reads the subfile record where you enter the option and knows the record key (**VnrNbr**) for that record. Pass this key as a parameter to your **VNRDLT**. Write the prototype and the **CALLP** to the **VNRDLT** in your **VNRSCHSPR** program, a copy of **VNRSCHSMON**.

Step 1. Make copies of your existing code

__ 1. Before you start, be sure that you have completed the subfile maintenance exercise. Make copies of your subfile maintenance **RPG IV** program. Copy **VnrSchsMon**, naming it **VNRSCHSPR**.

If you did not complete the subfile maintenance program **VnrSchSMon** in the earlier exercises, you can copy the sample solution from the course library:

Display File (DDS): **AS07V4LIB/QDDSSRC(VNDSCHS5S)** copy to **AS07nnn/QDDSSRC(VNDSCHS5S)**

RPG IV Program: **AS07V4LIB/QRPGLESRC(VNRSCHSMNS)** copy to **AS07nnn/QRPGLESRC(VNRSCHSPR)**

Compile the new source members and Call your **VnrSchSPR** program to verify that it functions correctly.

Step 2. Review the code that you are given

__ 1. In your library, you find the DDS source for the called program. The DDS is named **VndDlt**. Use this display file in conjunction with the **VNRDLT** program that you write.

__ 2. A copy of **VndDlt** follows:

```
A          R MSGWIN
A                                     WINDOW(07 4 10 70)

A                                     WDWTITLE((*TEXT 'Delete Pgm') +
A                                     (*COLOR BLU) (*DSPATR RI) +
A                                     *LEFT *TOP)

A                                     WDWBORDER((*COLOR BLU))


A            MESSAGE       50      3  2
```

```
     ** Dummy format to prevent display clearing
A          R DUMMY
A                                      ASSUME
A                                  2  4' '
```

Step 3. Review the existing program

__ 1. For review, browse the source for your **VNRSCHSPR** program. Execute it again to refresh your memory of how it works, as needed.

Step 4. Write the called program, VNRDLT

(**Hint:** You might want to use your existing **AddProgram** as a basis for the following code.)

__ 1. Your new **VNRDLT** program receives a parameter from the caller, the **VndNbr**. You need to define the **Vendor_PF** file.

__ 2. Also, code the procedure interface and the prototype for this program.

__ 3. Define the DSPF named **VndDlt**.

__ 4. Code the logic to delete the record of Vendor_PF that matches the VndNbr key.

__ 5. After deleting the record, build the field **Message** (50 characters) with something meaningful that tells the user that `the record with key nnnnn was successfully deleted.`

__ 6. Using the **MsgWin** record format, display the message window.

__ 7. Add logic to handle the situation where the record key does not match a record to be deleted. This program is called by many other programs and they might pass an invalid record key.

__ 8. Use the LR indicator to terminate and end the called program.

Step 5. Modify the caller, VNRSCHSPR

__ 1. Your DDS for the subfile does not require enhancement.

__ 2. Add the prototype definition for the call to **VNRDLT**.

__ 3. Find the existing CALL to DLTPROGRAM. Delete it and the line of code that follows, the PARM.

__ 4. Replace the lines you just deleted with a prototyped call, remembering to pass the VndNbr parameter with the call.

__ 5. Optionally, add code after the **If %Error** to better handle the situation where the program that is called is not found.

Step 6. Compile and test your programs

__ 1. Compile your DDS, **VndDlt**.

__ 2. Compile your two RPG IV programs, **VnrDlt** and **VNRSCHSPR**. Correct compilation errors as necessary.

__ 3. Test your programs by calling **VNRSCHSPR**. Position the subfile by entering the search key **F**.

__ 4. Enter the option for a delete, **4**, next to the **Federal Paper** record. As before, you see the message `Calling Delete Program`. We could remove this message now as it is really redundant. Leave it as it is for now.

__ 5. Press Enter. You should see a window pop up (this is the format in the **VndDlt** DSPF) that states that `record nnnn was deleted` (depends on what you placed in the Message field).

__ 6. To exit the window, press Enter. You are returned to the subfile display and should see that `Federal Paper` is no longer displayed (it has been deleted!)

__ 7. Exit your program.

__ 8. Refresh your copy of the **Vendor_PF** file by copying the data from the student master library, **AS07XXX**.

END OF LAB

Exercise 16. Subprocedures

What this exercise is about

This exercise provides an opportunity to code a subprocedure which can be used like an RPG IV built-in function (BIF). Your subprocedure receives input and then returns a value to the caller.

What you should be able to do

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

- Code a subprocedure that returns a value to the caller
- Code a prototype for a subprocedure
- Code a procedure interface for a subprocedure
- Code a local variable for a subprocedure

Introduction

Given a complete loan payment calculator application, your task is to modify certain inline calculations to create subprocedures that return a value to the caller.

You are not required to write any new logic. All you are required to do is to move existing calculations to subprocedures. Then, you modify the current procedure so that it calls the subprocedures.

Exercise instructions

Step 1. Code a local subprocedure

In this portion of the exercise, you modify an existing program by moving some inline calculations into two subprocedures and writing the code necessary to call them. You code the procedure interfaces to define the parameters that are passed to the subprocedures.

__ 1. Create a new QRPGLESRC source member, **RATPER. RATPER**, a subprocedure, receives as parameters the annual interest rate and number of payments per year, returning the periodic interest rate.

__ 2. You do not have to code the prototype at this time. It is coded in a separate source member in a later step.

__ 3. Code the procedure interface for RATPER on D specs. At some point your complete subprocedure is an independent compile unit. The PI must precisely define all parameter attributes in the expected sequence that they are passed from the caller.

Use the following display file to help you to define your fields correctly:

```
A                                      INDARA
A                                      CA03(03)
A          R PAYFMT
A                                  1  2DATE
A                                      EDTCDE(Y)
A                                  1 29'Loan Payment Calculator'
A                                      DSPATR(HI)
A                                  1 61'System:'
A                                  1 70SYSNAME
A                                  2  2TIME
A                                  2 61'User:'
A                                  2 70USER
A                                  4  2'Type values, press Enter.'
A                                      COLOR(BLU)
A                                  6 18'Loan amount . . . . . .'
A            PRINCIPAL      9Y 2B  6 50EDTWRD(' ,   , 0 .  ')
A                                      TEXT('LOAN AMOUNT')
A                                      DSPATR(MDT)
A                                      COMP(GT .00)
A                                      CHECK(FE)
A                                  8 18'Annual interest % . . . . . .'
A            RATEPCANN      5Y 3B  8 50EDTWRD('0 .   ')
A                                      TEXT('ANNUAL INTEREST %')
A                                      DSPATR(MDT)
A                                      RANGE(.000 50.000)
A                                      CHECK(FE)
A                                 10 18'Payments per year . . . . . .'
A            NBRPAYYR       2Y 0B 10 50EDTWRD(' 0')
A                                      TEXT('NUMBER OF PAYMENTS PER YEA-
A                                      R')
A                                      DSPATR(MDT)
A                                      RANGE(1 52)
```

```
A                                      CHECK(FE)
A                                 12 18'Number of payments  . . . . . .'
A            NBRPAYTOT      4Y 0B 12 50EDTWRD(' ,   ')
A                                      TEXT('TOTAL NUMBER OF PAYMENTS')
A                                      DSPATR(MDT)
A                                      RANGE(1 1600)
A                                      CHECK(FE)
A                                 14 18'Periodic interest . . . . . .'
A            RATEPERIOD    13Y11O 14 50EDTCDE(4)
A                                      TEXT('DECIMAL INTEREST RATE PER-
A                                      IOD')
A                                 16 18'Periodic payment amount . . . .'
A            PAYMENTAMT    13Y 2O 16 50EDTWRD('  ,   ,   ,  0.  ')
A                                      TEXT('PAYMENT AMOUNT')
A                                      DSPATR(HI)
A            ERRMSG        40     21 35
A                                 22  2'F3=Exit'
A                                      COLOR(BLU)
```

__ 4. If you have not already done so, create the DSPF, **LOANPAYD**. (You should have created the display file as part of the Monitor Groups exercise.)

__ 5. Review **LOANPAYSP** as follows. A copy is in your library.

```
FLoanPayD  CF   E             WorkStn IndDS(LoanPDS)
D LoanPDS         DS
D  Exit                   3      3N

 /free
      ExFmt PayFmt;

      DoW NOT Exit;
          RatePeriod = ( RatePCAnn * 0.01 ) / NbrPayYr;
          PaymentAmt = (Principal*RatePeriod) /
                       (1-(1/((1+RatePeriod)**NbrPayTot)));
          ExFmt PayFmt;
      EndDo;

      *InLR = *On;
      Return;
 /End-free
```

__ 6. Code the calculations for the **RATPER** subprocedure that calculate the periodic interest rate for a loan. If you like, you can simply copy the calculation from your copy of LOANPAYSP to RATPER.

__ 7. Code the P specifications for your RATPER subprocedure.

__ 8. Code the PI for your RATPER subprocedure.

__ 9. Code a RETURN operation in your RATPER subprocedure.

__ 10. Exit and add an appropriate text description before saving your new source member.

Step 2. Code a subprocedure prototype

__ 1. Create new QRPGLESRC source member **RATPER_PR**. The **PR** suffix is an abbreviation for prototype. By coding a subprocedure prototype in a source member separate from the rest of the procedure, you can /COPY the prototype into modules that call your subprocedure, as well as modules that include it.

__ 2. Code statements in this member for a prototype for the RATPER subprocedure. Be sure that the PR statements match the requirements for the RATPER subprocedure. If necessary, refer to the PI in RATPER for help.

__ 3. Exit and add appropriate descriptive text before saving your new source member.

Step 3. Code another subprocedure and prototype

__ 1. Create another subprocedure member **PAYMNT** in your source file QRPGLESRC.

__ 2. This subprocedure receives the principal amount, rate per period, and total number of payments. It returns the actual payment amount. See the above program LOANPAYSP for the calculation of amount of the monthly payment. You can copy the calculation from your LOANPAYSP to PAYMNT.

__ 3. Exit and add an appropriate text description before saving your new source member.

__ 4. Now create another subprocedure prototype member **PAYMNT_PR** in your source file QRPGLESRC. This prototype describes the input to your PAYMNT subprocedure. If necessary, refer to the PI in PAYMNT for help.

Some programmers prefer to include all subprocedure prototypes in a single source member. This approach is acceptable, but you might find including so many unused prototypes unnecessarily cumbersome. Others prefer to reduce the number of *unreferenced* compiler messages by including prototypes for only the subprocedures that are to be included in the program. By coding each prototype in a separate source member, you can eliminate unnecessary prototypes.

__ 5. Exit and add appropriate descriptive text before saving your new source member. You have created code that can be used and reused to create local and exportable subprocedures.

Step 4. Include subprocedures in a main program

__ 1. Modify your copy of LOANPAYSP. You might want to make a backup copy of the program.

__ 2. Code statements necessary to perform the loan payment calculation using your two new subprocedures and the prototyped source members.

__ 3. Be sure that the rate and payment calculations are not performed within LOANPAYSP, but rather in your subprocedures.

__ 4. Code the necessary /**COPY** statements to include both prototype and subprocedure source members at the appropriate places in LOANPAYSP. This includes your prototypes and subprocedures in the compile unit.

__ 5. Exit and add appropriate descriptive text before saving your new source member.

__ 6. You should now have five new source members. LOANPAYSP is the member to be compiled. Using the /COPY compiler directive, it directs the compiler to include the other four source members at the appropriate points.

__ 7. Compiling program LOANPAYSP is a slightly different process. Compile your program specifying that the DFTACTGRP = *NO. If you do not specify this parameter as *NO, your compilation will fail.

__ 8. Test LOANPAYSP. Notice that this copy of the program does not have a monitor group in it; so you might experience an error if you do not enter a valid interest rate. If you like, add a monitor group to check for an interest rate error as you did earlier for LOANPAYLP. You need to include the monitor group in the appropriate subprocedure.

__ 9. If it operates correctly, you have created an RPG IV program using local subprocedures.

END OF LAB

Exercise 17. Creating ILE objects

What this exercise is about

This exercise familiarizes you with the commands that support modular programming in the Integrated Language Environment.

What you should be able to do

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

- Create modules
- Create programs

Introduction

You are given the source code of an RPG IV procedure and a display file. Create a module and a program object from this source member.

Exercise Instructions

Step 1. Using Remote Systems LPEX Editor to create ILE objects

In this exercise you can use the Remote Systems LPEX Editor or PDM to create the objects. As always, you can switch between rsLPEX and 5250 emulation easily.

In your library, you notice several data files, **EMPMST**, **PRJMST**, and **RSNMST**. You also notice a display file, **MSTDSP**. These files are used by the RPG IV procedure, **PAYROLLG**.

If you choose to use 5250 emulation for the exercise, skip to the next step, “Using 5250 Emulation.”

__ 1. Open an edit session for your RPGLE source member **PAYROLLG**. We will use this member to reacquaint you with the various commands that can be executed from the Remote Systems LPEX Editor.

__ 2. From the editor's **Compile** menu, select **Compile Prompt**. Then select **CRTBNDRPG**.

__ 3. The Create RPG Module (CRTRPGMOD) window opens.

__ 4. Notice the generation severity level check value of 10. You could change it but we won’t.

__ 5. Check other parameters of the Create RPG Module (CRTRPGMOD) window. Specifically, notice the value of the **Debugging Views** parameter. You might need to set this to the level of debugging you need, for example ***ALL**.

__ 6. Click the **OK** button to submit your compile and close the window.

__ 7. You should see your compile messages in the lower portion of the window under the **Error List** tab.

__ 8. Switch to 5250 emulation. Look in your library for an object named **PAYROLLG**. Do you see an object type ***Module**?

__ 9. Enter option 5 next to the module to display module information. You can scroll down to view additional information for a specific item and press Enter to see different data. If you press Enter several times, for example, you notice that PAYROLLG is a procedure.

__ 10. Create a program from this module. Take option 26 in PDM and press Enter to view the CRTPGM command. Enter PAYROLLG as the program name. Notice that you can enter a plus sign (+) if this program contains more than a single module. The PAYROLLG program contains only one module, PAYROLLG.

__ 11. Check your messages that the command executed successfully.

__ 12. You should see your new *PGM object, **PAYROLLG**. You have just created an ILE module and an ILE program.

Step 2. Using 5250 Emulation

__ 1. In your library, find an RPGLE source member, PAYROLLG.

__ 2. Next to your member, enter option 15 and press F4. You see the display for the CRTRPGMOD command.

__ 3. Press Enter. Check your spool file to be sure that the module was created.

__ 4. Use WRKOBJPDM and look in your library for an object named **PAYROLLG**. Do you see an object type ***Module**?

__ 5. Enter option 5 next to the module to display module information. You can scroll down to view addition information for a specific item and press Enter to see different data. If you press Enter several times, for example, you will notice that PAYROLLG is a procedure.

__ 6. Create a program from this module. Take option 26 in PDM and press Enter to view the CRTPGM command. Enter PAYROLLG as the program name. Notice that you can enter a plus sign (+) if this program contains more than a single module. The PAYROLLG program will contain only one module, PAYROLLG.

__ 7. Check your messages that the command executed successfully.

__ 8. You should see your new *PGM object, **PAYROLLG**. You have just created an ILE module and an ILE program.

END OF LAB

Exercise 18. Bind by copy

What this exercise is about

This exercise provides an opportunity to use static binding, specifically bind by copy, wherein needed procedures are bound into an ILE program by copying the executable code from the modules containing the procedures to a program object.

What you should be able to do

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

- Edit an RPG IV source member to change a dynamic call to a static call in the prototype
- Create RPG IV modules
- Create a multi-module ILE program using bind by copy
- Reuse existing tested modules in multiple ILE programs using bind by copy

Introduction

In this exercise, you modify your copy of VNRDLT and VNRSCHSPR to use a bound call rather than a dynamic call.

Exercise instructions

Step 1. Make copies of your existing source members

__ 1. Make copies of your **VnrDlt** and **VnrSchSPR** source members, naming them **VNRDLTPROC** and **VNRSCHMAIN**.

If you did not complete the subfile maintenance program **VnrSchSPR** in the earlier exercises, you can copy the sample solution from the course library:

Display File (DDS): **AS07V4LIB/QDDSSRC(VNDSCHS5S)** copy to **AS07nnn/QDDSSRC(VNDSCHS5S)**

RPG IV Programs: **AS07V4LIB/QRPGLESRC(VNRSCHSPR)** copy to **AS07nnn/QRPGLESRC(VNRSCHMAIN)** and **AS07V4LIB/QRPGLESRC(VNRDLTS)** copy to **AS07nnn/QRPGLESRC(VNRDLTPROC)**

__ 2. Previously, each was compiled as an individual program. **VnrSchSPR** calls **VnrDlt** using a dynamic call.

__ 3. What in the code of each program makes this a dynamic call?

_____

_____

List the changes that you have to make to each program:

In procedure VNRDLTPROC:

_____

_____

In procedure VNRSCHMAIN:

_____

_____

Step 2. Modify VNRDLTPROC and VNRSCHMAIN

__ 1. Using your documented changes above, modify the source members so that VNRSCHMAIN will use a bound call to call VNRDLTPROC.

Step 3. Create modules VNRDLTPROC and VNRSCHMAIN

__ 1. Compile each source member, creating modules.

__ 2. When you have compiled successfully, confirm that the two modules, **VNRDLTPROC** and **VNRSCHMAIN** have been created.

__ 3. Display the module information for each module. Does either module know about the other yet? _____

Step 4. Create VNRSCHMAIN *PGM

__ 1. Run the ILE CRTPGM command and prompt it. If you use LPEX, you should click **Actions** from the editor menu and select **Create Program**.

__ 2. For the program name, we use the same name as the *MODULE, VNRSCHMAIN.

__ 3. Notice that the modules to be included can be expanded using the plus sign (+). Enter a plus sign (+) and press Enter.

__ 4. Enter VNRDLTPROC and your library for the second module. Notice that more modules could be entered to be included in the VNRSCHMAIN program.

__ 5. Press F10 or in LPEX, look for the entry module box. In PDM, you see that the default is *FIRST. This means that the first module in the list is the PEP for the program. Which module is this? _____

__ 6. Press Enter to create the program.

Step 5. Test the program, VNRSCHMAIN

__ 1. As before, test that a delete option works.

__ 2. Remember to refresh your copy of the Vendor_PF file after testing, copying from the master copy in AS07XXX.

Step 6. Explore the *PGM, VNRSCHMAIN

__ 1. Find your new VNRSCHMAIN program.

__ 2. Use the DSPPGM command and prompt with F4. For the DETAIL parameter, specify *ALL. Press Enter.

Notice that the phrase `More` appears near the lower right corner of the panel, but you are also prompted to `Press Enter to continue`. To avoid missing any information, scroll your display forward through each section until `More` is replaced by `Bottom`. Then press Enter to advance to the next display to view other information.

__ 3. What is the program entry procedure module?

_____

__ 4. What is the program attribute?

_____

__ 5. What is the type of program?

_____

__ 6. Press Enter to go to the display that lists modules. How many modules are bound into this program?

_____

__ 7. The service programs are listed next. Explore the remaining information. How many service programs are bound to this program?

_____

END OF LAB

Exercise 19. Bind by reference

What this exercise is about

This exercise provides an opportunity to use static binding, specifically bind by reference, wherein one or more procedures needed by an ILE program are centrally contained in a service program, and bound by reference to the ILE program during the CRTPGM binding process.

What you should be able to do

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

- Create a service program
- Bind by reference to modules in a service program object

Introduction

In the Bind by Copy exercise, module VNRDLTPROC was bound by copy into ILE program, VNRSCHMAIN.

It is often desirable to make commonly used modules available to the application by including them in a service program object. You now bind the VNRDLTPROC module into a new service program object, and then bind by reference to your new service program from the ILE procedure VNRSCHMAIN.

Reuse the modules from the Bind by copy exercise, and bind them together differently.

Exercise instructions

Step 1. Create service program, MySrvPgm

__ 1. Use the CRTSRVPGM command to create a service program that contains the module VNRDLTPROC. Specify the parameter **EXPORT(*ALL)**:

```
CRTSRVPGM   SRVPGM(MYSRVPGM)
            MODULE(VNRDLTPROC)
            EXPORT(*ALL)
```

Export indicates that all export capable symbols can be referenced beyond the scope of the object.

__ 2. What type of object did you create?

_____

Service programs usually contain more than one module. You just created a service program with only one module.

__ 3. What happens if you try to execute the command:

```
CALL MYSRVPGM
```

_____

A stand-alone service program cannot be called dynamically.

Step 2. Create and test a new program, VNRSCHREF

__ 1. Now create a new program that functions like VNRSCHMAIN in the Bind by copy exercise. Instead of binding the VNRDLTPROC module as we did in the bind by copy, all you need to do is bind the service programs that you just created. When you run CRTPGM, specify the program name as VNRSCHREF and bind the VNRSCHMAIN as you did before. You need to press F10 to see the parameter for binding of the service program.

__ 2. Test VNRSCHREF as you have before.

Step 3. Explore your service program and program object

__ 1. Run the DSPPGM command to explore the information available for the VNRSCHREF. Press the Enter key to move from display to display.

Which module is the program entry procedure? Why?

_____

__ 2. Press Enter and stop when you reach the Modules display. VNRDLTPROC was not specified in the CRTPGM command prompt. Rather, it was bound by reference to a service program.

__ 3. Press Enter and stop at the screen that displays information about service programs. Enter a 5 beside your MYSRVPGM and press Enter until you see the module VNRDLTPROC.

__ 4. Explore more. Make a note of any points of interest (Signature, Exports, and so on). Make a note of any questions and review them with the instructor and the rest of the class at the end of the exercise.

END OF LAB

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: Using OVERLAY and PUTOVR in display files

DDS: VNDINQOV

```
A                                      REF(*LIBL/VENDOR_PF)
A                                      INDARA
A          R HEADER_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          R PROMPT_FMT                PUTOVR
A                                      OVERLAY
A                                      CA03(03 'End Program')
A                                   3  3'Vendor number. . . . :'
A            VNDNBR_INQR     D  B   3 28COLOR(WHT) OVRDTA
A                                      REFFLD(VNDNBR DICTIONARY)
A  96                                  ERRMSG('Invalid vendor number' 96)
A          R DSPLY_FMT                 PUTOVR
A                                      OVERLAY
A                                   8  3'Name . . . . . . :'
A                                   9  3'Address  . . . . :'
A            VNDNAME    R        O  8 24OVRDTA
A            VNDSTREET  R        O  9 24OVRDTA
A            VNDCITY    R        O 10 24OVRDTA
A            VNDSTATE   R        O 10 49OVRDTA
A            VNDZIPCODER         O 10 53OVRDTA
A                                  11  3'Telephone. . . . :'
A            VNDAREACD  R        O 11 26OVRDTA
A                                  11 24'('
A                                  11 30')'
A            VNDTELNO   R        O 11 33OVRDTA
A                                       EDTWRD('0  -    ')
A                                  12  3'Sales Person . . :'
A            VNDSALES   R        O 12 24OVRDTA
A                                  13  3'Purchases YTD  . :'
A            VNDPRCHYTDR           13 24EDTCDE(J) OVRDTA
A                                  14  3'Balance Owed . . :'
A            VNDBALANCER           14 26EDTCDE(J) OVRDTA
A  60                                  DSPATR(HI)
A  60                                  COLOR(RED)
A          R FKEYS_FMT
A                                      OVERLAY
A                                  22  3'Please press enter to continue'
A                                  23  4'F3 = Exit'
A                                      COLOR(BLU)
```

RPG IV Program: VNRINQOV

```
 // Vendor master File
FVendor_PF IF   E           K Disk
 // Display File
FVndinqOvS 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

  Write     Header_Fmt;
  Write     Fkeys_Fmt;

  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
        // Check whether balance owed is greater than 5000.00
        HighBalance = VndBalance > 5000.00;
        // Display details
        Write Dsply_fmt;

    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
```

Exercise 2: Using DDS windows

DDS: VndInqPup

```
A                                      REF(*LIBL/VENDOR_PF)
A                                      INDARA
A          R HEADER_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          R PROMPT_FMT                PUTOVR
A                                      OVERLAY
A                                      CA03(03 'End Program')
A                                      CA04(04 'Display Vendor Details'
A                                   3  3'Vendor number. . . . :'
A            VNDNBR_INQR     D  I 3 28COLOR(WHT)
A                                      REFFLD(VNDNBR DICTIONARY)
A  96                                  ERRMSG('Invalid vendor number' 96)
A          R DSPLY_FMT                 PUTOVR
A                                      OVERLAY
A                                   8  3'Name . . . . . . :'
A                                   9  3'Address  . . . . :'
A            VNDNAME   R        O  8 24OVRDTA
A            VNDSTREET R        O  9 24OVRDTA
A            VNDCITY   R        O 10 24OVRDTA
A            VNDSTATE  R        O 10 49OVRDTA
A            VNDZIPCODER        O 10 53OVRDTA
A                                  11  3'Telephone. . . . :'
A            VNDAREACD R        O 11 26OVRDTA
A                                  11 24'('
A                                  11 30')'
A            VNDTELNO  R        O 11 33OVRDTA
A                                      EDTWRD('0  -    ')
A                                  12  3'Sales Person . . :'
A            VNDSALES  R        O 12 24OVRDTA
A                                  13  3'Purchases YTD  . :'
A            VNDPRCHYTDR           13 24EDTCDE(J) OVRDTA
A                                  14  3'Balance Owed . . :'
A            VNDBALANCER           14 26EDTCDE(J) OVRDTA
A  60                                  DSPATR(HI)
A  60                                  COLOR(RED)
A          R FKEYS_FMT
A                                      OVERLAY
A                                  22  3'Please press enter to continue'
A                                  23  4'F3 = Exit F4 = Display Vendor Deta-
A                                       il'
A                                      COLOR(BLU)
A** Pop-Up window format
A          R WINDOWFMT
A                                      CA12(12 'Press F12 to Return'
A                                      WINDOW(*DFT 7 50)
```

```
A                                       WDWBORDER((*COLOR WHT) (*DSPATR RI)-
A                                        (*CHAR '         '))

A                                       WDWTITLE((*TEXT 'Vendor Detail') -
A                                       *CENTER)

A                                   2  2'Vendor Name:'
A                                   3  2'MTD Purchased:'
A            VNDNAME    R           2 17
A            VNDPRCHMTDR            3 17EDTCDE(J)
A                                   6  2'Press F12 to Return'
A                                       COLOR(BLU)

 ** Dummy format to prevent display clearing
A          R DUMMY
A                                       ASSUME
A                                   2  4' '
```

RPG IV program: VNRINQPUP

```
 // Vendor master File
FVendor_PF IF   E           K Disk
 // Display File
FVndInqPup CF   E             Workstn IndDS(WkIndicators)

// Indicator Data Structure
D WkIndicators    DS
D Exit                    3      3N
D Details                 4      4N
D Cancel                 12     12N
D HighBalance            60     60N
D NotFound               96     96N
 /FREE


 Write     Header_Fmt;
 Write     Fkeys_Fmt;

 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
       // Check whether balance owed is greater than 5000.00
       HighBalance = VndBalance > 5000.00;
       // Display details
       Write Dsply_fmt;
```

```
    If Details;

       Exfmt WindowFmt; // Display Popup window

    EndIf;

   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
```

Exercise 3: Array processing

DDS:VNDINQAR

```
A                                      REF(*LIBL/DICTIONARY)
A                                      CA03(03 'End Program')
A                                      INDARA
 ** Prompt Format
A          R PROMPT_FMT                OVERLAY
A                                    1  2USER
A                                    1 30'Vendor Name Search'
A                                      DSPATR(HI)
A                                      COLOR(WHT)
A                                    1 71SYSNAME
A                                    2 61DATE
A                                      EDTCDE(Y)
A                                    2 71TIME
A                                    3  2'Enter partial vendor name:  '
A            SEARCH        25A  I  3 31
A  96                                  ERRMSG('No vendors found' 96)
A                                    4  2'Press enter to continue'
A                                      COLOR(BLU)
 ** Subfile data record
A          R VSEARCHDTA                SFL
A            VNDNBR    R        O  9  2
A            VNDNAME   R        O  9  8
A            VNDAREACD R        O  9 34
A            VNDTELNO  R        O  9 38EDTWRD('   -    ')
A            VNDSALES  R        O  9 51
 ** Subfile Control Format
A          R VSEARCHCTL                SFLCTL(VSEARCHDTA)
A                                      SFLSIZ(0050)
A                                      SFLPAG(0014)
A  40                                  SFLEND(*MORE)
A  75                                  SFLCLR
A  85                                  SFLDSPCTL
A  95                                  SFLDSP
A                                      OVERLAY
 ** Headings for Subfile
A          R HEADER_FMT                OVERLAY
A                                    7  2'Vend'
A                                    8  3'No'
A                                    7 11'Vendor Name'
A                                    7 34'Telephone'
A                                    7 56'Sales Person'
 ** Function Keys
A          R FKEY_FMT
A                                   24  4'F3 = Exit'
A                                      COLOR(BLU)
```

RPG IV Program: VNRINQARS

```
 // Vendor master File
FVendor_PF IF   E           K Disk
 // Display File
```

```
FVndinqAR  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

D Months          S              9A   Dim(12) CtData PerRcd(4)
 /FREE
  // Convert month number to month name and format date
  Today = %trimr(months(*month)) + ' '
           + %char(*day) + ', ' + %char(*year);

  Write     Header_Fmt;
  Write     Fkeys_Fmt;

  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
        // Check whether balance owed is greater than 5000.00
        HighBalance = VndBalance > 5000.00;
        // Display details
        Write Dsply_fmt;

    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

**ctdata months
January  February March    April
May      June     July     August
SeptemberOctober  November December

DDS (Optional Lab - VNDINQARE)

A                                      REF(*LIBL/VENDOR_PF)
A                                      INDARA
A          R HEADER_FMT
A                                  1  2USER
A                                  1 30'Vendor Inquiry'
A                                      COLOR(WHT)
```

```
A                                      1 71SYSNAME
A            TODAY         20   O  2 45
A*                                     2 61DATE
A*                                         EDTCDE(Y)
A                                      2 71TIME
A          R PROMPT_FMT                    PUTOVR
A                                          OVERLAY
A                                          CA03(03 'End Program')
A                                      3  3'Vendor number. . . . :'
A            VNDNBR_INQR     D  B  3 28COLOR(WHT) OVRDTA
A                                          REFFLD(VNDNBR DICTIONARY)
A  96                                      ERRMSG('Invalid vendor number' 96)
A          R DSPLY_FMT                     PUTOVR
A                                          OVERLAY
A                                      8  3'Name . . . . . . :'
A                                      9  3'Address  . . . . :'
A            VNDNAME   R           O  8 24OVRDTA
A            VNDSTREET R           O  9 24OVRDTA
A            VNDCITY   R           O 10 24OVRDTA
A            VNDSTATE  R           O 10 49OVRDTA
A            VNDZIPCODER           O 10 53OVRDTA
A                                     11  3'Telephone. . . . :'
A            VNDAREACD R           O 11 26OVRDTA
A                                     11 24'('
A                                     11 30')'
A            VNDTELNO  R           O 11 33OVRDTA
A                                          EDTWRD('0  -    ')
A                                     12  3'Sales Person . . :'
A            VNDSALES  R           O 12 24OVRDTA
A                                     13  3'Purchases YTD  . :'
A            VNDPRCHYTDR           13 24EDTCDE(J) OVRDTA
A                                     14  3'Balance Owed . . :'
A            VNDBALANCER           14 26EDTCDE(J) OVRDTA
A  60                                      DSPATR(HI)
A  60                                      COLOR(RED)
A          R FKEYS_FMT
A                                          OVERLAY
A                                     22  3'Please press enter to continue'
A                                     23  4'F3 = Exit'
A                                          COLOR(BLU)
```

RPG IV Program (Optional - VNRINQARE):

```
 // Vendor master File
FVendor_PF IF   E           K Disk
 // Display File
FVndinqARe 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

D Months          S              9A   Dim(12) CtData PerRcd(4)
D Index           S              3P 0
```

```
D DaySuff         S              2A   dim(20) ctdata perrcd(10)

 /FREE
  // Determine the suffix of the day
  Index = %Div(*Day:21) + (%Rem(*Day:21));
  // special handling for the 31st
  If *day = 31;
     Index = Index - 10;
  EndIf;
  // Convert month number to month name and format date
  Today = %trimr(months(*month)) + ' '
          + %char(*day) + DaySuff(Index)
          + ', ' + %char(*year);

  Write Header_Fmt;
  Write Fkeys_Fmt;

  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
       // Check whether balance owed is greater than 5000.00
       HighBalance = VndBalance > 5000.00;
       // Display details
       Write Dsply_fmt;

   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


**ctdata months
January  February March    April
May      June     July     August
SeptemberOctober  November December
**CtData DaySuff
stndrdthththththththth
ththththththththththth
```

Exercise 4: Data structures and data areas

DDS: VNDINQDT

```
A                                      REF(*LIBL/VENDOR_PF)
A                                      INDARA
A          R HEADER_FMT
A            CONAME        40   O  1 25
A                                    3  2USER
A                                    3 30'Vendor Inquiry'
A                                       COLOR(WHT)
A                                    3 71SYSNAME
A            TODAY         18   O  4 50
A*                                   4 61DATE
A*                                      EDTCDE(Y)
A                                    4 71TIME
A          R PROMPT_FMT                 PUTOVR
A                                       OVERLAY
A                                       CA03(03 'End Program')
A                                    6  3'Vendor number. . . . :'
A            VNDNBR_INQR     D  B  6 28COLOR(WHT) OVRDTA
A                                       REFFLD(VNDNBR DICTIONARY)
A  96                                   ERRMSG('Invalid vendor number' 96)
A          R DSPLY_FMT                  PUTOVR
A                                       OVERLAY
A                                    8  3'Name . . . . . . :'
A                                    9  3'Address  . . . . :'
A            VNDNAME   R          O  8 24OVRDTA
A            VNDSTREET R          O  9 24OVRDTA
A            VNDCITY   R          O 10 24OVRDTA
A            VNDSTATE  R          O 10 49OVRDTA
A            VNDZIPCODER          O 10 53OVRDTA
A                                   11  3'Telephone. . . . :'
A            VNDAREACD R          O 11 26OVRDTA
A                                   11 24'('
A                                   11 30')'
A            VNDTELNO  R          O 11 33OVRDTA
A                                       EDTWRD('0  -    ')
A                                   12  3'Sales Person . . :'
A            VNDSALES  R          O 12 24OVRDTA
A                                   13  3'Purchases YTD  . :'
A            VNDPRCHYTDR            13 24EDTCDE(J) OVRDTA
A                                   14  3'Balance Owed . . :'
A            VNDBALANCER            14 26EDTCDE(J) OVRDTA
A  60                                   DSPATR(HI)
A  60                                   COLOR(RED)
A                                   16  3'Tax Owed . . . . :'
A            TAX            7  2O 16 29EDTCDE(J) OVRDTA
A          R FKEYS_FMT
A                                       OVERLAY
A                                   22  3'Please press enter to continue'
A                                   23  4'F3 = Exit'
A                                       COLOR(BLU)
```

RPG IV program: VNRINQDT

```
H ExPrOpts(*ResDecpos)
 // Vendor master File
FVendor_PF IF   E           K Disk
 // Display File
FVndinqDT  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

D Months          S              9A   Dim(12) CtData PerRcd(4)
  // Data Structure to hold company name
D Company         DS                   DTAARA
D  TaxrateA                       3
D   Taxrate                       3  3 OVERLAY(TaxRateA)
D  CoName                        40
D  Empty                          7

 /FREE
  // get company name from Data area
  // Write headings on first page of report
  In Company;
  // Convert month number to month name and format date
  Today = %trimr(months(*month)) + ' '
           + %char(*day) + ', ' + %char(*year);

  Write     Header_Fmt;
  Write     Fkeys_Fmt;

  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
        // Check whether balance owed is greater than 5000.00
        HighBalance = VndBalance > 5000.00;
        // Calcualte tax owed on balance
        Tax = VndBalance * TaxRate;
        // Display details
        Write Dsply_fmt;

    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


**ctdata months
January  February March    April
May      June     July     August
SeptemberOctober  November December
```

Exercise 5: Inquiry subfiles

DDS:VNDSUBFILE

```
A                                      REF(*LIBL/DICTIONARY)
A                                      CA03(03 'End Program')
A                                      INDARA
 ** Prompt Format
A          R PROMPT_FMT                OVERLAY
A                                   1  2USER
A                                   1 30'Vendor Name Display'
A                                      DSPATR(HI)
A                                      COLOR(WHT)
A                                   1 71SYSNAME
A                                   2 61DATE
A                                      EDTCDE(Y)
A                                   2 71TIME
 ** Data record
A          R VSEARCHDTA                SFL
A            VNDNBR    R        O  9  2
A            VNDNAME   R        O  9  8
A            VNDAREACD R        O  9 34
A            VNDTELNO  R        O  9 38EDTWRD('   -    ')
A            VNDSALES  R        O  9 51
 ** Subfile Control Record
A          R VSEARCHCTL                SFLCTL(VSEARCHDTA)
A                                      SFLDSPCTL
A                                      SFLDSP
A                                      SFLSIZ(0050)
A                                      SFLPAG(0014)
A  40                                  SFLEND(*MORE)
A                                      OVERLAY
 ** Heading Format
A          R HEADER_FMT                OVERLAY
A                                   7  2'Vend'
A                                   8  3'No'
A                                   7 11'Vendor Name'
A                                   7 34'Telephone'
A                                   7 56'Sales Person'
 ** Function Keys
A          R FKEY_FMT
A                                  24  4'F3 = Exit'
A                  COLOR(BLU)
```

RPG IV program: VNRSUBFILE

```
fVndnam_lf if   e           k disk
fVndsubfilecf   e             workstn      Sfile(VSearchDta:RRN)
F                                          IndDS(VndIndic)
D VndIndic        DS
D   Exit                   3      3N
D   SflEnd                40     40N

D RRN             s              3  0

 /FREE
  Write FKey_Fmt;
  Write Prompt_Fmt;

  Read VndNam_LF;
  Rrn = 1;
  // Load Subfile
  DoW NOT %eof(VndNam_LF);
    Write VsearchDta;
    Read VndNam_lf;
    RRN=RRN+1;
  EndDo;
  // Display Subfile
  Dow not Exit;
    Write Header_Fmt;
    SflEnd = %eof(VndNam_LF);
    Exfmt VsearchCtl;
  EndDo;

  *inlr=*on;
  Return;
 /END-FREE
```

DDS:VNDSUBNR

```
A                                      REF(*LIBL/DICTIONARY)
A                                      CA03(03 'End Program')
A                                      INDARA
 ** Prompt Format
A          R PROMPT_FMT                OVERLAY
A                                    1  2USER
A                                    1 30'Vendor Name Display'
A                                       DSPATR(HI)
A                                       COLOR(WHT)
A                                    1 71SYSNAME
A                                    2 61DATE
A                                       EDTCDE(Y)
A                                    2 71TIME
 ** Data record
A          R VSEARCHDTA                SFL
A            VNDNBR    R        O  9  2
A            VNDNAME   R        O  9  8
```

```
A            VNDAREACD R        O  9 34
A            VNDTELNO  R        O  9 38EDTWRD('   -    ')
A            VNDSALES  R        O  9 51
 ** Subfile Control Record
A          R VSEARCHCTL                SFLCTL(VSEARCHDTA)
A                                      SFLDSPCTL
A                                      SFLDSP
A                                      SFLSIZ(0050)
A                                      SFLPAG(0014)
A  40                                  SFLEND(*MORE)
A                                      OVERLAY
 ** Heading Format
A          R HEADER_FMT                OVERLAY
A                                    7  2'Vend'
A                                    8  3'No'
A                                    7 11'Vendor Name'
A                                    7 34'Telephone'
A                                    7 56'Sales Person'
 ** Function Keys
A          R FKEY_FMT
A                                   24  4'F3 = Exit'
A                   COLOR(BLU)
```

RPG IV: VNRSUBNR with enhancements to handle empty DB file

```
fVndnam_lf if   e           k disk
fVndsubnr  cf   e             workstn      Sfile(VSearchDta:RRN)
F                                          IndDS(VndIndic)
D VndIndic        DS
D   Exit                  3      3N
D   SflEnd               40     40N

D RRN             s              3  0
D EmptyMsg        S             25A
 /FREE
  Write FKey_Fmt;
  Write Prompt_Fmt;

  Read VndNam_LF;
  Rrn = 1;
  // Load Subfile
  DoW NOT %eof(VndNam_LF);
    Write VsearchDta;
    Read VndNam_lf;
    RRN=RRN+1;
  EndDo;
  // Display Subfile
  Dow not Exit;
    Write Header_Fmt;
    // Add logic to handle empty subfile
    If Rrn = 1;
       EmptyMsg = 'No records to display';
       Dsply EmptyMsg '*REQUESTER¬';
       Leave;
```

```
    EndIf;
    SflEnd = %eof(VndNam_LF);
    Exfmt VsearchCtl;
  EndDo;

  *inlr=*on;
  Return;
 /END-FREE
```

Exercise 6: Inquiry subfiles with search

DDS: VNDSEARCH

```
A                                      REF(*LIBL/DICTIONARY)
A                                      CA03(03 'End Program')
A                                      INDARA
 ** Prompt Format
A          R PROMPT_FMT                OVERLAY
A                                   1  2USER
A                                   1 30'Vendor Name Search'
A                                      DSPATR(HI)
A                                      COLOR(WHT)
A                                   1 71SYSNAME
A                                   2 61DATE
A                                      EDTCDE(Y)
A                                   2 71TIME
A                                   3  2'Enter partial vendor name:  '
A            SEARCH        25A  I  3 31
A  96                                  ERRMSG('No vendors found' 96)
A                                   4  2'Press enter to continue'
A                                      COLOR(BLU)
 ** Subfile data record
A          R VSEARCHDTA                SFL
A            VNDNBR    R        O  9  2
A            VNDNAME   R        O  9  8
A            VNDAREACD R        O  9 34
A            VNDTELNO  R        O  9 38EDTWRD('   -    ')
A            VNDSALES  R        O  9 51
 ** Subfile Control Format
A          R VSEARCHCTL                SFLCTL(VSEARCHDTA)
A                                      SFLSIZ(0050)
A                                      SFLPAG(0014)
A  40                                  SFLEND(*MORE)
A  75                                  SFLCLR
A  85                                  SFLDSPCTL
A  95                                  SFLDSP
A                                      OVERLAY
 ** Headings for Subfile
A          R HEADER_FMT                OVERLAY
A                                   7  2'Vend'
A                                   8  3'No'
A                                   7 11'Vendor Name'
A                                   7 34'Telephone'
A                                   7 56'Sales Person'
 ** Function Keys
A          R FKEY_FMT
A                                  24  4'F3 = Exit'
A                                      COLOR(BLU)
```

RPG IV Program: VNRSEARCH

```
fVndnam_lf if   e           k disk
```

```
fVndsearch cf   e             workstn      Sfile(VSearchDta:RRN)
F                                          IndDS(VndIndic)

D VndIndic        DS
D  Exit                  03     03N
D  SflEnd                40     40N
D  SflClr                75     75N
D  SflDspCtl             85     85N
D  SflDsp                95     95N
D  NotFound              96     96N

d RRN             s              3  0      INZ

 /FREE
  DoW not Exit;
    // Setup search key and position file cursor
    Search = %TrimL(Search);
    SetLL Search VndNam_lf;
    Read VndNam_lf;
    Rrn = 1;
    // Load Subfile and read rest of records unless at End of File
    DoW Not %Eof(VndNam_LF);
      Write VsearchDta;
      Read VndNam_lf;
      Rrn = Rrn+1;
    EndDo;
    // Display Subfile - do we have any records to display?
    If Rrn <= 1;
      NotFound = *on;
    Else;
      SflDsp = *on;
    EndIF;
    // Display records in subfile
    Write Header_Fmt;
    SflDspCtl = *on;
    Write VsearchCtl;
    SflDspCtl = *off;
    SflDsp = *off;
    Exfmt Prompt_fmt;
    NotFound = *off;
    // Start a new search - clear subfile and reset RRN
    SflClr = *on;
    Write VSearchCtl;
    SflClr = *off;
    RRN = 0;
  EndDo;
  *inlr=*on;
  Return;

  BegSR *InzSR;
    Write FKey_Fmt;
    Exfmt Prompt_fmt;
    // Set SFLEND indicator
    SflEnd = *on;
  Endsr;
 /END-FREE
```

Exercise 7: Modularize vendor subfile search

DDS: VNDSCHS1

```
A                                      REF(*LIBL/DICTIONARY)
A                                      CA03(03 'End Program')
A                                      INDARA
 ** Prompt Format
A          R PROMPT_FMT                OVERLAY
A                                   1  2USER
A                                   1 30'Vendor Name Search'
A                                      DSPATR(HI)
A                                      COLOR(WHT)
A                                   1 71SYSNAME
A                                   2 61DATE
A                                      EDTCDE(Y)
A                                   2 71TIME
A                                   3  2'Enter partial vendor name:  '
A            SEARCH        25A  I  3 31
A                                   4  2'Press enter to continue'
A                                      COLOR(BLU)
 ** Subfile data record
A          R VSEARCHDTA                SFL
A            VNDNBR    R        O  9  2
A            VNDNAME   R        O  9  8
A            VNDAREACD R        O  9 34
A            VNDTELNO  R        O  9 38EDTWRD('   -    ')
A            VNDSALES  R        O  9 51
 ** Subfile Control Format
A          R VSEARCHCTL                SFLCTL(VSEARCHDTA)
A                                      SFLSIZ(0050)
A                                      SFLPAG(0014)
A  40                                  SFLEND(*MORE)
A  75                                  SFLCLR
A  85                                  SFLDSPCTL
A  95                                  SFLDSP
A                                      OVERLAY
 ** Headings for Subfile
A          R HEADER_FMT                OVERLAY
A                                   7  2'Vend'
A                                   8  3'No'
A                                   7 11'Vendor Name'
A                                   7 34'Telephone'
A                                   7 56'Sales Person'
 ** Function Keys
A          R FKEY_FMT
A                                  24  4'F3 = Exit'
A                                      COLOR(BLU)
 ** Message for empty subfile
A          R MSG
A                                      OVERLAY
A                                  12 32'No Records Found'
A                                      DSPATR(HI)
```

RPG program: VNRSCHS1S

```
FVndNam_LF IF   E           K Disk
FVndSchS1  CF   E             Workstn Sfile(VSearchDta:Rrn)
F                                     IndDS(WkStnIndics)
D WkStnIndics     DS
D  Exit                   3      3N
D  SflEnd                40     40N
D  SflDspCtl             85     85N
D  SflDsp                95     95N
D  SflClr                75     75N

D Rrn             S              4  0 INZ
D EmptySfl        S               N


 /FREE

  DoW not Exit;
    ExSR SearchRtn;
  EndDo;

  *InLr = *on;

   // subroutines
  BegSR *InzSR;     // Initialization subroutine
    Write FKey_Fmt;

    ExFmt Prompt_Fmt;
  Endsr;

  BegSR SearchRtn;
    ExSr SFLClear;          // Clear subfile for new search

    SetLL Search VndNam_LF; // Position file cursor
                            // using search key
    Read VndNam_LF;         // Read first record after cursor

    Rrn = 1;                // Rrn set to 1 even if we are at EOF
    Exsr Fill;              // Fill Subfile
    Exsr Prompt;            // Prompt for new search
  EndSR;

  Begsr Fill;

   // Load entire subfile
   // If already at EOF, will ll not enter loop
    Dow (NOT %EOF(VndNam_LF)) AND (Rrn <= 9999);
      Write VSearchDta;
      Read VndNam_LF;
      Rrn = Rrn + 1;
    Enddo;

    EmptySfl = (Rrn <= 1);  // No records to display?
    SflEnd = %EOF(VndNam_LF); // Have reached EOF of Item_PF?
```

```
  Endsr;

  Begsr Prompt;

    If NOT EmptySfl;

      SflDspCtl = *ON; // Display Subfile
      SflDsp = *ON;

   Else;

     SflDspCtl = *ON;  // No records to view - display message
     SflDsp = *OFF;
     Write Msg;
   Endif;

   Write Header_Fmt;
   Write VSearchCtl;
   ExFmt Prompt_Fmt;
 Endsr;

 Begsr SFLCLear;   // Subfile clear subroutine
   SflClr = *on;
   SflDsp = *OFF;
   SflDspCtl = *OFF;
    Write VSearchCtl;  // New search - clear subfile
    SflClr = *off;
  EndSR;
 /END-FREE
```

Exercise 8: Page + 1 and Pagedown

DDS: VNDSCHS2

```
     A                                      REF(*LIBL/DICTIONARY)
     A                                      CA03(03 'End Program')
     A                                      INDARA
      ** Prompt Format
     A          R PROMPT_FMT                OVERLAY
     A                                   1  2USER
     A                                   1 30'Vendor Name Search'
     A                                      DSPATR(HI)
     A                                      COLOR(WHT)
     A                                   1 71SYSNAME
     A                                   2 61DATE
     A                                      EDTCDE(Y)
     A                                   2 71TIME
     A                                   3  2'Enter partial vendor name:  '

     A            SEARCH        25A  I  3 31
     A                                   4  2'Press enter to continue'
     A                                      COLOR(BLU)
      ** Subfile data record
     A          R VSEARCHDTA                SFL
     A            VNDNBR    R        O  9  2
     A            VNDNAME   R        O  9  8
     A            VNDAREACD R        O  9 34
     A            VNDTELNO  R        O  9 38EDTWRD('   -    ')
     A            VNDSALES  R        O  9 51
      ** Subfile Control Format
     A          R VSEARCHCTL                SFLCTL(VSEARCHDTA)
>>   A                                      SFLSIZ(&SFLSIZE)
>>   A N40                                  PAGEDOWN(30)
     A                                      SFLPAG(0014)
     A  40                                  SFLEND(*MORE)
     A  75                                  SFLCLR
     A  85                                  SFLDSPCTL
     A  95                                  SFLDSP
     A                                      OVERLAY
>>   A            SFLSIZE        5S 0P
>>   A            SFLRRN         4S 0H      SFLRCDNBR
      ** Headings for Subfile
     A          R HEADER_FMT                OVERLAY
     A                                   7  2'Vend'
     A                                   8  3'No'
     A                                   7 11'Vendor Name'
     A                                   7 34'Telephone'
     A                                   7 56'Sales Person'
      ** Function Keys
     A          R FKEY_FMT
     A                                  24  4'F3 = Exit'
     A                                      COLOR(BLU)
      ** Message for empty subfile
     A          R MSG
```

```
     A                                      OVERLAY
     A                                  12 32'No vendors Found'
     A                                      DSPATR(HI)
```

RPG IV: VNRSCHS2S

```
     FVndNam_LF IF   E           K Disk
     FVndSchS2S CF   E             Workstn Sfile(VSearchDta:Rrn)
     F                                     IndDS(WkStnIndics)

     D WkStnIndics     DS
     D  Exit                   3      3N
>>   D  PageDown              30     30N
     D  SflEnd                40     40N
     D  SflDspCtl             85     85N
     D  SflDsp                95     95N
     D  SflClr                75     75N

     D Rrn             S              4  0 INZ
>>   D RrnCount        S                   Like(Rrn)
>>   D EmptySfl        S               N


      /FREE
       DoW not Exit;
>>       Select;
>>       When PageDown;
>>         Exsr NextPage;

>>       Other;
           ExSR SearchRtn;
>>       Endsl;

       EndDo;

       *InLr = *on;

        // subroutines
       BegSR *InzSR;     // Initialization subroutine

>>       SflSize = 15;
>>       SflEnd = *On;
         Write FKey_Fmt;
         ExFmt Prompt_Fmt;
       Endsr;

       BegSR SearchRtn;
         ExSr SFLClear;          // Clear subfile for new search

         SetLL Search VndNam_LF; // Position file cursor
                                 // using search key
         Read VndNam_LF;         // Read first record after cursor

         Rrn = 1;                // Rrn set to 1 even if we are at EOF
         Exsr Fill;              // Fill Subfile
         Exsr Prompt;            // Prompt for new search
```

```
       EndSR;

       Begsr Fill;

        // Load entire subfile
>> 7   RrnCount = 1;
        // If already at EOF, will not enter loop
>>       Dow (NOT %EOF(VndNam_LF))
>>           AND (RrnCount <= (SflSize-1));
           Write VSearchDta;
           Read VndNam_LF;
           Rrn = Rrn + 1;
>>         RrnCount = RrnCount + 1;
         Enddo;

         EmptySfl = (Rrn <= 1);  // No records to display?
         SflEnd = %EOF(VndNam_LF); // Have reached EOF of Item_PF?

       Endsr;

       Begsr Prompt;

         If NOT EmptySfl;

           SflDspCtl = *ON; // Display Subfile
           SflDsp = *ON;
>>         SflRrn = Rrn - 1;

         Else;

           SflDspCtl = *ON;  // No records to view - display message
           SflDsp = *OFF;
           Write Msg;
         Endif;

         Write Header_Fmt;
>>       Write Prompt_Fmt;
>>       Exfmt VSearchCtl;
>>       Read  Prompt_Fmt;
       Endsr;

>>     Begsr NextPage;

       // Load next block of records
>>       Exsr Fill;
>>       Exsr Prompt;
>>     Endsr;

       Begsr SFLCLear;   // Subfile clear subroutine
         SflClr = *on;
         SflDsp = *OFF;
         SflDspCtl = *OFF;
         Write VSearchCtl;  // New search - clear subfile
         SflClr = *off;
       EndSR;
      /END-FREE
```

Exercise 9: Add PageUp

DDS: VNDSCHS3

```
     A                                      REF(*LIBL/DICTIONARY)
     A                                      CA03(03 'End Program')
     A                                      INDARA
      ** Prompt Format
     A          R PROMPT_FMT                OVERLAY
     A                                    1  2USER
     A                                    1 30'Vendor Name Search'
     A                                       DSPATR(HI)
     A                                       COLOR(WHT)
     A                                    1 71SYSNAME
     A                                    2 61DATE
     A                                       EDTCDE(Y)
     A                                    2 71TIME
     A                                    3  2'Enter partial vendor name:  '
     A            SEARCH        25A  I  3 31
     A                                    4  2'Press enter to continue'
     A                                       COLOR(BLU)
      ** Subfile data record
     A          R VSEARCHDTA                SFL
     A            VNDNBR    R        O  9  2
     A            VNDNAME   R        O  9  8
     A            VNDAREACD R        O  9 34
     A            VNDTELNO  R        O  9 38EDTWRD('   -    ')
     A            VNDSALES  R        O  9 51
      ** Subfile Control Format
     A          R VSEARCHCTL                SFLCTL(VSEARCHDTA)
     A N40                                  PAGEDOWN(30)
>>   A N41                                  PAGEUP(31)
>>   A                                      SFLSIZ(&SFLSIZE)
     A                                      SFLPAG(0014)
     A  40                                  SFLEND(*MORE)
     A  75                                  SFLCLR
     A  85                                  SFLDSPCTL
     A  95                                  SFLDSP
     A                                      OVERLAY
>>   A            SFLSIZE        5S 0P
>>   A            SFLRRN         4S 0H      SFLRCDNBR
      ** Headings for Subfile
     A          R HEADER_FMT                OVERLAY
     A                                    7  2'Vend'
     A                                    8  3'No'
     A                                    7 11'Vendor Name'
     A                                    7 34'Telephone'
     A                                    7 56'Sales Person'
      ** Function Keys
     A          R FKEY_FMT
     A                                   24  4'F3 = Exit'
     A                                       COLOR(BLU)
      ** Message for empty subfile
     A          R MSG
```

```
     A                                      OVERLAY
     A                                  12 32'No vendors Found'
     A                                      DSPATR(HI)
```

RPG IV:VNRSCHS3

```
     FVndNam_LF IF   E           K Disk
>>   FVndSchS3S CF   E             Workstn Sfile(VSearchDta:Rrn)
     F                                     IndDS(WkStnIndics)

     D WkStnIndics     DS
     D  Exit                   3      3N
     D  PageDown              30     30N
>>   D  PageUp                31     31N
     D  SflEnd                40     40N
>>   D  SflBegin              41     41N
     D  SflDspCtl             85     85N
     D  SflDsp                95     95N
     D  SflClr                75     75N

     D Rrn             S              4  0 INZ
     D RrnCount        S                   Like(Rrn)
     D EmptySfl        S               N


      /FREE
       DoW not Exit;
         Select;
         When PageDown;
           Exsr NextPage;
>>       When PageUp;
>>         Exsr PrevPage;

         Other;
           ExSR SearchRtn;
         Endsl;

       EndDo;

       *InLr = *on;

        // subroutines
       BegSR *InzSR;     // Initialization subroutine

         SflSize = 15;
         SflEnd = *On;
>>       SflBegin = *ON;
         Write FKey_Fmt;
         ExFmt Prompt_Fmt;
       Endsr;

       BegSR SearchRtn;
         ExSr SFLClear;          // Clear subfile for new search

         SetLL Search VndNam_LF; // Position file cursor
                                 // using search key
```

```
>>       Exsr CheckBOF;
>>       // Read VndNam_LF;        // Read first record after cursor

         Rrn = 1;               // Rrn set to 1 even if we are at EOF
         Exsr Fill;             // Fill Subfile
         Exsr Prompt;           // Prompt for new search
       EndSR;

>>     Begsr CheckBOF;

       // Check if this is the first record in the file
>>       ReadP VndNam_LF;
>>       SflBegin = %EOF(VndNam_LF);

>>       If %EOF(VndNam_LF);
>>         Setll *Start VndNam_LF;
>>       Endif;

>>       Read VndNam_LF;
>>     Endsr;

       Begsr Fill;

        // Load entire subfile
       RrnCount = 1;
        // If already at EOF, will not enter loop
         Dow (NOT %EOF(VndNam_LF))
             AND (RrnCount <= (SflSize-1));
           Write VSearchDta;
           Read VndNam_LF;
           Rrn = Rrn + 1;
           RrnCount = RrnCount + 1;
         Enddo;

         EmptySfl = (Rrn <= 1);  // No records to display?
         SflEnd = %EOF(VndNam_LF); // Have reached EOF of VndNam_LF?

       Endsr;

       Begsr Prompt;

         If NOT EmptySfl;

           SflDspCtl = *ON; // Display Subfile
           SflDsp = *ON;
           SflRrn = Rrn - 1;

         Else;

           SflDspCtl = *ON;  // No records to view - display message
           SflDsp = *OFF;
>>         SflBegin = *ON;
           Write Msg;
         Endif;

         Write Header_Fmt;
```

```
          Write Prompt_Fmt;
          Exfmt VSearchCtl;
          Read  Prompt_Fmt;
        Endsr;

        Begsr NextPage;

        // Load next block of records
          Exsr Fill;
          Exsr Prompt;
        Endsr;

>>      Begsr PrevPage;
          // Find out where I am now in the DB
>>        Chain 1 VSearchDta; // Chain to first record in subfile
>>        Exsr SflClear;

>>        Setll VndName VndNam_LF;// Load previous block of records
>>        ReadP VndNam_LF;
>>        RrnCount  = 1;

          // Read back through the file one page
>>        DOW (NOT %EOF(VndNam_LF)) AND
>>            (RrnCount <= (SflSize-1));
>>          ReadP VndNam_LF;
>>          RrnCount = RrnCount + 1;
>>        Enddo;

           // Prevent PAGEUP if first record
>>        Exsr CheckBOF;
>>        Rrn = 1;
>>        Exsr Fill;
>>        Exsr Prompt;
>>      Endsr;

        Begsr SFLCLear;   // Subfile clear subroutine
          SflClr = *on;
          SflDsp = *OFF;
          SflDspCtl = *OFF;
          Write VSearchCtl;  // New search - clear subfile
          SflClr = *off;
>>        SflBegin = *OFF; // Set BOF of subfile
        EndSR;
       /END-FREE
```

Exercise 10: Add SFLPAG = SFLSIZ

DDS: VNDSUBS4

```
A                                      REF(*LIBL/DICTIONARY)
A                                      CA03(03 'End Program')
A                                      INDARA
 ** Prompt Format
A          R PROMPT_FMT                OVERLAY
A                                    1  2USER
A                                    1 30'Vendor Name Search'
A                                       DSPATR(HI)
A                                       COLOR(WHT)
A                                    1 71SYSNAME
A                                    2 61DATE
A                                       EDTCDE(Y)
A                                    2 71TIME
A                                    3  2'Enter partial vendor name:  '
A            SEARCH        25A  I  3 31
A                                    4  2'Press enter to continue'
A                                       COLOR(BLU)
 ** Subfile data record
A          R VSEARCHDTA                SFL
A            VNDNBR    R        O  9  2
A            VNDNAME   R        O  9  8
A            VNDAREACD R        O  9 34
A            VNDTELNO  R        O  9 38EDTWRD('   -    ')
A            VNDSALES  R        O  9 51
 ** Subfile Control Format

A          R VSEARCHCTL                SFLCTL(VSEARCHDTA)
A N40                                  PAGEDOWN(30)
A N41                                  PAGEUP(31)
A                                      SFLSIZ(&SFLSIZE)
A                                      SFLPAG(0014)
A  40                                  SFLEND(*MORE)
A  75                                  SFLCLR
A  85                                  SFLDSPCTL
A  95                                  SFLDSP
A                                      OVERLAY
A            SFLSIZE        5S 0P
A            SFLRRN         4S 0H      SFLRCDNBR
 ** Headings for Subfile
A          R HEADER_FMT                OVERLAY
A                                    7  2'Vend'
A                                    8  3'No'
A                                    7 11'Vendor Name'
A                                    7 34'Telephone'
A                                    7 56'Sales Person'
 ** Function Keys
A          R FKEY_FMT
A                                   24  4'F3 = Exit'
A                                       COLOR(BLU)
 ** Message for empty subfile
```

```
     A          R MSG
     A                                      OVERLAY
     A                                   12 32'No vendors Found'
     A                                      DSPATR(HI)
```

RPG IV: VNRSUBS4

```
     FVndNam_LF IF   E           K Disk
>>   FVndSchS4S CF   E             Workstn Sfile(VSearchDta:Rrn)
     F                                     IndDS(WkStnIndics)

     D WkStnIndics     DS
     D  Exit                   3      3N
     D  PageDown              30     30N
     D  PageUp                31     31N
     D  SflEnd                40     40N
     D  SflBegin              41     41N
     D  SflDspCtl             85     85N
     D  SflDsp                95     95N
     D  SflClr                75     75N

     D Rrn             S              4  0 INZ
     D RrnCount        S                   Like(Rrn)
     D EmptySfl        S               N


      /FREE
       DoW not Exit;
         Select;
         When PageDown;
           Exsr NextPage;
         When PageUp;
           Exsr PrevPage;

         Other;
           ExSR SearchRtn;
         Endsl;

       EndDo;

       *InLr = *on;

        // subroutines
       BegSR *InzSR;     // Initialization subroutine

>>       SflRrn = 1;
         SflSize = 14;
         SflEnd = *On;
         SflBegin = *ON;
         Write FKey_Fmt;
         ExFmt Prompt_Fmt;
       Endsr;

       BegSR SearchRtn;
         ExSr SFLClear;          // Clear subfile for new search
```

```
          SetLL Search VndNam_LF; // Position file cursor
                                 // using search key
          Exsr CheckBOF;
          // Read VndNam_LF;        // Read first record after cursor

          Rrn = 1;               // Rrn set to 1 even if we are at EOF
          Exsr Fill;             // Fill Subfile
          Exsr Prompt;           // Prompt for new search
        EndSR;

        Begsr CheckBOF;

        // Check if this is the first record in the file
          ReadP VndNam_LF;
          SflBegin = %EOF(VndNam_LF);

          If %EOF(VndNam_LF);
            Setll *Start VndNam_LF;
          Endif;

          Read VndNam_LF;
        Endsr;

        Begsr Fill;

         // Load entire subfile
        RrnCount = 1;
         // If already at EOF, will not enter loop
          Dow (NOT %EOF(VndNam_LF))
>>            AND (RrnCount <= (SflSize));
            Write VSearchDta;
            Read VndNam_LF;
            Rrn = Rrn + 1;
            RrnCount = RrnCount + 1;
          Enddo;

          EmptySfl = (Rrn <= 1);  // No records to display?
          SflEnd = %EOF(VndNam_LF); // Have reached EOF of VndNam_LF?

        Endsr;

        Begsr Prompt;

          If NOT EmptySfl;

            SflDspCtl = *ON; // Display Subfile
            SflDsp = *ON;
            SflRrn = Rrn - 1;

          Else;

            SflDspCtl = *ON;  // No records to view - display message
            SflDsp = *OFF;
>>          SflBegin = *ON;
            Write Msg;
```

```
         Endif;

         Write Header_Fmt;
         Write Prompt_Fmt;
         Exfmt VSearchCtl;
         Read  Prompt_Fmt;
       Endsr;

       Begsr NextPage;

       // Load next block of records
>>       Exsr SflClear;
>>       Rrn = 1;
         Exsr Fill;
         Exsr Prompt;
       Endsr;

       Begsr PrevPage;
         // Find out where I am now in the DB
         Chain 1 VSearchDta; // Chain to first record in subfile
         Exsr SflClear;

         Setll VndName VndNam_LF;// Load previous block of records
         ReadP VndNam_LF;
         RrnCount  = 1;

         // Read back through the file one page
         DOW (NOT %EOF(VndNam_LF)) AND
>>           (RrnCount <= (SflSize));
           ReadP VndNam_LF;
           RrnCount = RrnCount + 1;
         Enddo;

          // Prevent PAGEUP if first record
         Exsr CheckBOF;
         Rrn = 1;
         Exsr Fill;
         Exsr Prompt;
       Endsr;

       Begsr SFLCLear;   // Subfile clear subroutine
         SflClr = *on;
         SflDsp = *OFF;
         SflDspCtl = *OFF;
         Write VSearchCtl;  // New search - clear subfile
         SflClr = *off;
         SflBegin = *OFF; // Set BOF of subfile
       EndSR;
      /END-FREE
```

Exercise 11: Add maintenance

The solution that follows is based on modifying the source of Exercise 10.

DDS:VNDSCHS5

```
     A                                      REF(*LIBL/DICTIONARY)
     A                                      CA03(03 'End Program')
     A                                      INDARA
** Subfile data record
     A          R VSEARCHDTA                SFL
>>   A            OPTION         1A  I  9  2VALUES(' ' '1' '2' '4')
     A            VNDNBR    R        O  9  4
     A            VNDNAME   R        O  9 10
     A            VNDAREACD R        O  9 36
     A            VNDTELNO  R        O  9 40EDTWRD('   -    ')
     A            VNDSALES  R        O  9 53
      ** Subfile Control Format
     A          R VSEARCHCTL                SFLCTL(VSEARCHDTA)
     A N40                                  PAGEDOWN(30)
     A N41                                  PAGEUP(31)
     A                                      SFLSIZ(&SFLSIZE)
     A                                      SFLPAG(0014)
     A  40                                  SFLEND(*MORE)
     A  75                                  SFLCLR
     A  85                                  SFLDSPCTL
     A  95                                  SFLDSP
     A                                      OVERLAY
     A            SFLSIZE        5S 0P
     A            SFLRRN         4S 0H      SFLRCDNBR
>>    ** Prompt Format
>>   A*         R PROMPT_FMT                OVERLAY
     A                                   1  2USER
     A                                   1 30'Vendor Name Search'
     A                                      DSPATR(HI)
     A                                      COLOR(WHT)
     A                                   1 71SYSNAME
     A                                   2 61DATE
     A                                      EDTCDE(Y)
     A                                   2 71TIME
     A                                   3  2'Enter partial vendor name:  '
     A            SEARCH        25A  I  3 31
     A                                   4  2'Press enter to continue'
     A                                      COLOR(BLU)
>>    ** Headings for Subfile
>>   A*         R HEADER_FMT                OVERLAY
>>   A                                   6  1'Opt'
     A                                   7  4'Vend'
     A                                   8  5'No'
     A                                   7 13'Vendor Name'
     A                                   7 36'Telephone'
     A                                   7 58'Sales Person'
      ** Function Keys
     A          R FKEY_FMT
     A                                  24  4'F3 = Exit'
```

```
     A                                      COLOR(BLU)
      ** Message for empty subfile
     A          R MSG
     A                                      OVERLAY
>>   A*                                  12 32'No vendors Found'
>>   A            MESSAGE       25       12 32
     A                                        DSPATR(HI)
```

RPG IV: VNRSCHS5

```
     FVndNam_LF IF   E           K Disk
>>   FVndSchS5S CF   E             Workstn Sfile(VSearchDta:Rrn)
     F                                     IndDS(WkStnIndics)

     D WkStnIndics     DS
     D  Exit                   3      3N
     D  PageDown              30     30N
     D  PageUp                31     31N
     D  SflEnd                40     40N
     D  SflBegin              41     41N
     D  SflDspCtl             85     85N
     D  SflDsp                95     95N
     D  SflClr                75     75N

     D Rrn             S              4  0 INZ
     D RrnCount        S                   Like(Rrn)
     D EmptySfl        S               N


      /FREE
       DoW not Exit;
         Select;
         When PageDown;
           Exsr NextPage;
         When PageUp;
           Exsr PrevPage;

         Other;
           ExSR SearchRtn;
         Endsl;

       EndDo;

       *InLr = *on;

        // subroutines
       BegSR *InzSR;     // Initialization subroutine

         SflRrn = 1;
         SflSize = 14;
         SflEnd = *On;
         SflBegin = *ON;
         SflDspCtl = *On;
         Write FKey_Fmt;
>>       ExFmt VSearchCtl;
```

```
Endsr;

BegSR SearchRtn;
  ExSr SFLClear;          // Clear subfile for new search

  SetLL Search VndNam_LF; // Position file cursor
                          // using search key
  Exsr CheckBOF;
  // Read VndNam_LF;        // Read first record after cursor

  Rrn = 1;               // Rrn set to 1 even if we are at EOF
  Exsr Fill;             // Fill Subfile
  Exsr Prompt;           // Prompt for new search
EndSR;

Begsr CheckBOF;

// Check if this is the first record in the file
  ReadP VndNam_LF;
  SflBegin = %EOF(VndNam_LF);

  If %EOF(VndNam_LF);
    Setll *Start VndNam_LF;
  Endif;

  Read VndNam_LF;
Endsr;

Begsr Fill;

 // Load entire subfile
RrnCount = 1;
 // If already at EOF, will not enter loop
  Dow (NOT %EOF(VndNam_LF))
      AND (RrnCount <= (SflSize));
    Write VSearchDta;
    Read VndNam_LF;
    Rrn = Rrn + 1;
    RrnCount = RrnCount + 1;
  Enddo;

  EmptySfl = (Rrn <= 1);  // No records to display?
  SflEnd = %EOF(VndNam_LF); // Have reached EOF of VndNam_LF?

Endsr;

Begsr Prompt;

  If NOT EmptySfl;

    SflDspCtl = *ON; // Display Subfile
    SflDsp = *ON;
    SflRrn = Rrn - 1;

  Else;
```

```
            SflDspCtl = *ON;  // No records to view - display message
            SflDsp = *OFF;
>>          SflBegin = *On;
>>          Message = 'No vendors found';
            Write Msg;
          Endif;

>>        // Write Header_Fmt;
>>        ExFmt VSearchCtl;
>>         // ExFmt Prompt_Fmt;
          If Rrn > 1; // Process changes only if subfile has records
>>           Exsr Changes;
          EndIf;
        Endsr;

        Begsr NextPage;

        // Load next block of records
          Exsr SflClear;
>>        Rrn = 1;
          Exsr Fill;
          Exsr Prompt;
        Endsr;

        Begsr PrevPage;
          // Find out where I am now in the DB
          Chain 1 VSearchDta; // Chain to first record in subfile
          Exsr SflClear;

          Setll VndName VndNam_LF;// Load previous block of records
          ReadP VndNam_LF;
          RrnCount  = 1;

          // Read back through the file one page
          DOW (NOT %EOF(VndNam_LF)) AND
              (RrnCount <= (SflSize));
            ReadP VndNam_LF;
            RrnCount = RrnCount + 1;
          Enddo;

           // Prevent PAGEUP if first record
          Exsr CheckBOF;
          Rrn = 1;
          Exsr Fill;
          Exsr Prompt;
        Endsr;

        Begsr SFLCLear;   // Subfile clear subroutine
          SflClr = *on;
          SflDsp = *OFF;
          SflDspCtl = *OFF;
          Write VSearchCtl;  // New search - clear subfile
          SflClr = *off;
          SflBegin = *OFF; // Set BOF of subfile
        EndSR;
```

```
>>     Begsr Changes;
>>       ReadC VSearchDta;

       // Process subfile changes
>> 5     Dow NOT %EOF(VndSchs5s);
>>         Select;

       // Add new Vendor
>>         When Option = '1';
>>           SflDspCtl = *ON;  // display message
>>           SflDsp = *OFF;
>>           SflBegin = *ON;
>>           Message = 'Calling Add Program';
>>           Write Msg;
>>           Exfmt VSearchCtl;
>>    /End-free
>>   C                   Call(E)   'ADDPROGRAM'
>>    /Free
>>           If %ERROR;
       // Check if Add function completed successfully
>>           Endif;

       // Change Vendor details
>>         When Option = '2';
>>           SflDspCtl = *ON;  // display message
>>           SflDsp = *OFF;
>>           SflBegin = *ON;
>>           Message = 'Calling Change Program';
>>           Write Msg;
>>           Exfmt VSearchCtl;
>>    /End-free
>>   C                   Call(E)   'CHGPROGRAM'
>>   C                   Parm                    VndNbr
>>    /Free
>>           If %ERROR;
       // Check Change function completed successfully
>>           Endif;

       // Delete existing Vendor
>>         When Option = '4';
>>           SflDspCtl = *ON;  // display message
>>           SflDsp = *OFF;
>>           SflBegin = *ON;
>>           Message = 'Calling Delete Program';
>>           Write Msg;
>>           Exfmt VSearchCtl;
>>    /End-free
>>   C                   Call(E)   'DLTPROGRAM'
>>   C                   Parm                    VndNbr
>>    /Free
>>           If %ERROR;
       // Check Delete function completed successfully
>>           Endif;
>>         Endsl;

>>         ReadC VSearchDta;
```

```
>>       Enddo;
         Write FKey_Fmt;
>>     Endsr;
      /END-FREE
```

The solution that follows is based on modifying the source of exercise 7 VNxSCHS1.

DDS:VNDSCHS5

```
     A                                      REF(*LIBL/DICTIONARY)
     A                                      CA03(03 'End Program')
     A                                      INDARA
      ** Subfile data record
     A          R VSEARCHDTA                SFL
>>   A            OPTION         1A  I  9  2VALUES(' ' '1' '2' '4')
     A            VNDNBR    R        O  9  4
     A            VNDNAME   R        O  9 10
     A            VNDAREACD R        O  9 36
     A            VNDTELNO  R        O  9 40EDTWRD('   -    ')
     A            VNDSALES  R        O  9 53
      ** Subfile Control Format
     A          R VSEARCHCTL                SFLCTL(VSEARCHDTA)
     A                                      SFLSIZ(0050)
     A                                      SFLPAG(0014)
     A  40                                  SFLEND(*MORE)
     A  75                                  SFLCLR
     A  85                                  SFLDSPCTL
     A  95                                  SFLDSP
     A                                      OVERLAY
      ** Prompt Format
     A*         R PROMPT_FMT                OVERLAY
     A                                   1  2USER
     A                                   1 30'Vendor Name Search'
     A                                      DSPATR(HI)
     A                                      COLOR(WHT)
     A                                   1 71SYSNAME
     A                                   2 61DATE
     A                                      EDTCDE(Y)
     A                                   2 71TIME
     A                                   3  2'Enter partial vendor name:  '
     A            SEARCH        25A  I  3 31
>>   A* 96                                  ERRMSG('No vendors found' 9
     A                                   4  2'Press enter to continue'
     A                                      COLOR(BLU)
>>    ** Headings for Subfile
>>   A*         R HEADER_FMT                OVERLAY
>>   A                                   6  1'Opt'
     A                                   7  4'Vend'
     A                                   8  5'No'
     A                                   7 13'Vendor Name'
     A                                   7 36'Telephone'
     A                                   7 58'Sales Person'
      ** Function Keys
```

```
     A          R FKEY_FMT
     A                                  24  4'F3 = Exit'
     A                                      COLOR(BLU)
      ** Message for empty subfile
     A          R MSG
     A                                      OVERLAY
>>   A*                                 12 32'No vendors Found'
>>   A            MESSAGE       25      12 32
     A                                      DSPATR(HI)
```

RPG IV: VNRSCHS5

```
     FVndNam_LF IF   E           K Disk
>>   FVndSchS5  CF   E             Workstn Sfile(VSearchDta:Rrn)
     F                                     IndDS(WkStnIndics)

     D WkStnIndics     DS
     D  Exit                   3      3N
     D  SflEnd                40     40N
     D  SflDspCtl             85     85N
     D  SflDsp                95     95N
     D  SflClr                75     75N

     D Rrn             S              4  0 INZ
>>   D EmptySfl        S               N


      /FREE
       DoW not Exit;
         ExSR SearchRtn;
       EndDo;

       *InLr = *on;

        // subroutines
       BegSR *InzSR;     // Initialization subroutine
         Write FKey_Fmt;
        //  ExFmt Prompt_Fmt;
>>       SflDspCtl = *On;
>>       ExFmt VSearchCtl;
       Endsr;

       BegSR SearchRtn;
         ExSr SFLClear;          // Clear subfile for new search

         SetLL Search VndNam_LF; // Position file cursor
                                 // using search key
         Read VndNam_LF;         // Read first record after cursor

         Rrn = 1;                // Rrn set to 1 even if we are at EOF
         Exsr Fill;              // Fill Subfile
         Exsr Prompt;            // Prompt for new search
       EndSR;
```

```
       Begsr Fill;

        // Load entire subfile
        // If already at EOF, will ll not enter loop
         Dow (NOT %EOF(VndNam_LF)) AND (Rrn <= 9999);
           Write VSearchDta;
           Read VndNam_LF;
           Rrn = Rrn + 1;
         Enddo;

         EmptySfl = (Rrn <= 1);  // No records to display?
         SflEnd = %EOF(VndNam_LF); // Have reached EOF of Item_PF?

       Endsr;

       Begsr Prompt;

         If NOT EmptySfl;

           SflDspCtl = *ON; // Display Subfile
           SflDsp = *ON;

         Else;

           SflDspCtl = *ON;  // No records to view - display message
           SflDsp = *OFF;
>>         Message = 'No vendors found';
>>         Write Msg;
         Endif;

         // Write Header_Fmt;
         ExFmt VSearchCtl;
         // ExFmt Prompt_Fmt;
         If Rrn > 1; // Process changes only if subfile has records
>>          Exsr Changes;
         EndIf;
       Endsr;
       Begsr SFLCLear;   // Subfile clear subroutine
         SflClr = *on;
         SflDsp = *OFF;
         SflDspCtl = *OFF;
         Write VSearchCtl;  // New search - clear subfile
         SflClr = *off;
       EndSR;

>>     Begsr Changes;
>>       ReadC VSearchDta;

       // Process subfile changes
>>       Dow NOT %EOF(VndSchs5);
>>         Select;

       // Add new Vendor
>>         When Option = '1';
>>           SflDspCtl = *ON;  // display message
>>           SflDsp = *OFF;
```

```
>>           Message = 'Calling Add Program';
>>           Write Msg;
>>           Exfmt VSearchCtl;
>>     /End-free
>>    C                   Call(E)    'ADDPROGRAM'
>>     /Free
>>           If %ERROR;
       // Check if Add function completed successfully
>>           Endif;

       // Change Vendor details
>>         When Option = '2';
>>           SflDspCtl = *ON;  // display message
>>           SflDsp = *OFF;
>>           Message = 'Calling Change Program';
>>           Write Msg;
>>           Exfmt VSearchCtl;
>>     /End-free
>>    C                   Call(E)    'CHGPROGRAM'
>>    C                   Parm                    VndNbr
>>     /Free
>>           If %ERROR;
       // Check Change function completed successfully
>>           Endif;

       // Delete existing Vendor
>>         When Option = '4';
>>           SflDspCtl = *ON;  // display message
>>           SflDsp = *OFF;
>>           Message = 'Calling Delete Program';
>>           Write Msg;
>>           Exfmt VSearchCtl;
>>     /End-free
>>    C                   Call(E)    'DLTPROGRAM'
>>    C                   Parm                    VndNbr
>>     /Free
>>           If %ERROR;
       // Check Delete function completed successfully
>>           Endif;
>>         Endsl;

>>         ReadC VSearchDta;

>>       Enddo;

     Write FKey_Fmt;
>>     Endsr;
      /End-Free
```

Exercise 12: Error handling and BIFs

RPG IV - AddProgram:

```
FVendor_PF UF A E           K Disk
FVndAdd    CF   E             Workstn

 /FREE
  ExFmt AddWin;
  Setll(E) VndNbr Vendor_PF;

  If not %error;
     If not %equal(Vendor_PF);
        Write Vendor_Fmt;
        Message = 'Vendor number ' + %char(VndNbr) +
                  ' added successfully';
        ExFmt MsgWin;
     Else;
        Message = 'Error - Vendor ' +
                  '(' + %char(VndNbr) + ')' +
                  ' already exists in file';
        ExFmt MsgWin;
     EndIf;
  EndIf;

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

Exercise 13: Using monitor groups

RPG IV: VnrSchSErr

```
     FVndNam_LF IF   E           K Disk
     FVndSchS5  CF   E             Workstn Sfile(VSearchDta:Rrn)
     F                                     IndDS(WkStnIndics)

     D WkStnIndics     DS
     D  Exit                   3      3N
     D  PageDown              30     30N
     D  PageUp                31     31N
     D  SflEnd                40     40N
     D  SflBegin              41     41N
     D  SflDspCtl             85     85N
     D  SflDsp                95     95N
     D  SflClr                75     75N

     D Rrn             S              4  0 INZ
     D RrnCount        S                   Like(Rrn)
     D EmptySfl        S               N
>>   D ErrorMsg        S              30A

      /FREE
       DoW not Exit;
         Select;
         When PageDown;
           Exsr NextPage;
         When PageUp;
           Exsr PrevPage;

         Other;
           ExSR SearchRtn;
         Endsl;

       EndDo;

       *InLr = *on;

        // subroutines
       BegSR *InzSR;     // Initialization subroutine

         SflRrn = 1;
         SflSize = 14;
         SflEnd = *On;
         SflBegin = *ON;
         SflDspCtl = *On;
         Write FKey_Fmt;
         ExFmt VSearchCtl;
       Endsr;

       BegSR SearchRtn;
         ExSr SFLClear;          // Clear subfile for new search
```

```
    SetLL Search VndNam_LF; // Position file cursor
                           // using search key
    Exsr CheckBOF;
    // Read VndNam_LF;        // Read first record after cursor

    Rrn = 1;               // Rrn set to 1 even if we are at EOF
    Exsr Fill;             // Fill Subfile
    Exsr Prompt;           // Prompt for new search
  EndSR;

  Begsr CheckBOF;

  // Check if this is the first record in the file
    ReadP VndNam_LF;
   SflBegin = %EOF(VndNam_LF);

   If %EOF(VndNam_LF);
     Setll *Start VndNam_LF;
   Endif;

   Read VndNam_LF;
 Endsr;

 Begsr Fill;

  // Load entire subfile
 RrnCount = 1;
  // If already at EOF, will not enter loop
   Dow (NOT %EOF(VndNam_LF))
       AND (RrnCount <= (SflSize));
      Write VSearchDta;
      Read VndNam_LF;
      Rrn = Rrn + 1;
      RrnCount = RrnCount + 1;
    Enddo;

    EmptySfl = (Rrn <= 1);  // No records to display?
    SflEnd = %EOF(VndNam_LF); // Have reached EOF of VndNam_LF?

  Endsr;

  Begsr Prompt;

    If NOT EmptySfl;

      SflDspCtl = *ON; // Display Subfile
      SflDsp = *ON;
      SflRrn = Rrn - 1;

    Else;

      SflDspCtl = *ON;  // No records to view - display message
      SflDsp = *OFF;
      SflBegin = *On;
      Message = 'No vendors found';
```

```
      Write Msg;
    Endif;

    // Write Header_Fmt;
    ExFmt VSearchCtl;
     // ExFmt Prompt_Fmt;
    If Rrn > 1; // Process changes only if subfile has records
      Exsr Changes;
  EndIf;
Endsr;

Begsr NextPage;

// Load next block of records
  Exsr SflClear;
  Rrn = 1;
  Exsr Fill;
  Exsr Prompt;
Endsr;

Begsr PrevPage;
  // Find out where I am now in the DB
  Chain 1 VSearchDta; // Chain to first record in subfile
  Exsr SflClear;

  Setll VndName VndNam_LF;// Load previous block of records
  ReadP VndNam_LF;
  RrnCount  = 1;

  // Read back through the file one page
  DOW (NOT %EOF(VndNam_LF)) AND
      (RrnCount <= (SflSize));
    ReadP VndNam_LF;
    RrnCount = RrnCount + 1;
  Enddo;

   // Prevent PAGEUP if first record
  Exsr CheckBOF;
  Rrn = 1;
  Exsr Fill;
  Exsr Prompt;
Endsr;

Begsr SFLCLear;   // Subfile clear subroutine
  SflClr = *on;
  SflDsp = *OFF;
  SflDspCtl = *OFF;
  Write VSearchCtl;  // New search - clear subfile
  SflClr = *off;
  SflBegin = *OFF; // Set BOF of subfile
EndSR;

Begsr Changes;
  ReadC VSearchDta;
```

```
      // Process subfile changes
        Dow NOT %EOF(VndSchs5);
          Select;

      // Add new Vendor
          When Option = '1';
            SflDspCtl = *ON;  // display message
            SflDsp = *OFF;
            SflBegin = *ON;
            Message = 'Calling Add Program';
            Write Msg;
            Exfmt VSearchCtl;
     /End-free
    C                   Call(E)   'ADDPROGRMS'
     /Free
             If %ERROR;
       // Check if Add function completed successfully
             Endif;

       // Change Vendor details
           When Option = '2';
>>           Monitor;
              SflDspCtl = *ON;  // display message
              SflDsp = *OFF;
              SflBegin = *ON;
              Message = 'Calling Change Program';
              Write Msg;
              Exfmt VSearchCtl;
       /End-free
>>   C                   Call      'CHGPROGRAM'
     C                   Parm                    VndNbr
      /Free
>>           On-error 00211;
>>     // Check Change function completed successfully
>>             ErrorMsg = 'Change Program not found '
>>                      + %editw(%status:'0    ');
>>             Dsply ErrorMsg '*REQUESTER';
>>           EndMon;

       // Delete existing Vendor
           When Option = '4';
             SflDspCtl = *ON;  // display message
             SflDsp = *OFF;
             SflBegin = *ON;
             Message = 'Calling Delete Program';
             Write Msg;
             Exfmt VSearchCtl;
      /End-free
     C                   Call(E)   'DLTPROGRAM'
     C                   Parm                    VndNbr
      /Free
             If %ERROR;
       // Check Delete function completed successfully
             Endif;
           Endsl;
```

```
          ReadC VSearchDta;

        Enddo;
        Write FKey_Fmt;
       Endsr;
      /END-FREE
```

Step 3-2 You should see a message in your message queue informing you that there was an error. The e-extender and %error method could have been coded to do this as well. Which method you use depends on the situation, what will work best for you, and what standards are in force in your organization.

RPG IV: LOANPAYLP

```
FLoanPayD  CF   E             WorkStn IndDS(LoanPDS)
D LoanPDS         DS
D  Exit                   3      3N

 /free
      ExFmt PayFmt;

      DoW NOT Exit;
          Monitor;
          RatePeriod = ( RatePCAnn * 0.01 ) / NbrPayYr;
          PaymentAmt = (Principal*RatePeriod) /
                       (1-(1/((1+RatePeriod)**NbrPayTot)));
          On-error 00907;
             ErrMsg = 'Incorrect value for Annual Interest';

          EndMon;
          ExFmt PayFmt;
          ErrMsg = *blank;
      EndDo;

      *InLR = *On;
      Return;
 /end-free
```

Exercise 14: Using dates

RPG IV: DATERPG

```
FDateDSPF  CF   E             Workstn IndDS(WKIND)
D WkInd           DS
D  Exit                    3      3N
D  BadDate                40     40N
D  Future                 50     50N
 /Free
  Today = %date(*date);       // Determine value of today
  Write     Header;
  Write     Footer;
  Exfmt     Prompt;

  Dow NOT Exit;
  // Reset date indicator
      Future = *off;
  // Test for valid date value
      Test(DE) CharDate;
      If %error;
         BadDate = *On;
      Else;
  // Display details
  // extract year, month, day from date netered
         Year  = %subdt(%date(CharDate):*years);
         Month = %subdt(%date(CharDate):*months);
         Day   = %subdt(%date(CharDate):*days);
  // Determine number of days between job date and date entered
         Days = %Diff(today :%date(CharDate):*days);
  // Is Days in the past or the future?
         If Days < 0;
            Future = *on;
         Else;
            Future = *off;
         EndIf;

         Write Detail;
      EndIf;
  // Display prompt
      Exfmt Prompt;
  Enddo;

  *InLR = *On;

 /End-free
```

Exercise 15: Prototyping

RPG IV Program VnrSchSPR:

```
     FVndNam_LF IF   E           K Disk
     FVndSchS5  CF   E             Workstn Sfile(VSearchDta:Rrn)
     F                                     IndDS(WkStnIndics)

     D WkStnIndics     DS
     D  Exit                   3      3N
     D  PageDown              30     30N
     D  PageUp                31     31N
     D  SflEnd                40     40N
     D  SflBegin              41     41N
     D  SflDspCtl             85     85N
     D  SflDsp                95     95N
     D  SflClr                75     75N

     D Rrn             S              4  0 INZ
     D RrnCount        S                   Like(Rrn)
     D EmptySfl        S               N
     D ErrorMsg        S             30A

>>   D VnrDelete       PR                  ExtPgm('VNRDLT')
>>   D  VndNbr                        5  0
      /FREE
       DoW not Exit;
         Select;
         When PageDown;
           Exsr NextPage;
         When PageUp;
           Exsr PrevPage;

         Other;
           ExSR SearchRtn;
         Endsl;

       EndDo;

       *InLr = *on;

        // subroutines
       BegSR *InzSR;     // Initialization subroutine

         SflRrn = 1;
         SflSize = 14;
         SflEnd = *On;
         SflBegin = *ON;
         SflDspCtl = *On;
         Write FKey_Fmt;
         ExFmt VSearchCtl;
       Endsr;
```

```
BegSR SearchRtn;
  ExSr SFLClear;          // Clear subfile for new search

  SetLL Search VndNam_LF; // Position file cursor
                         // using search key
  Exsr CheckBOF;
  // Read VndNam_LF;        // Read first record after cursor

  Rrn = 1;               // Rrn set to 1 even if we are at EOF
  Exsr Fill;             // Fill Subfile
  Exsr Prompt;           // Prompt for new search
EndSR;

Begsr CheckBOF;

// Check if this is the first record in the file
  ReadP VndNam_LF;
  SflBegin = %EOF(VndNam_LF);

  If %EOF(VndNam_LF);
    Setll *Start VndNam_LF;
  Endif;

  Read VndNam_LF;
Endsr;

Begsr Fill;

 // Load entire subfile
RrnCount = 1;
 // If already at EOF, will not enter loop
  Dow (NOT %EOF(VndNam_LF))
      AND (RrnCount <= (SflSize));
    Write VSearchDta;
    Read VndNam_LF;
    Rrn = Rrn + 1;
    RrnCount = RrnCount + 1;
  Enddo;

  EmptySfl = (Rrn <= 1);  // No records to display?
  SflEnd = %EOF(VndNam_LF); // Have reached EOF of VndNam_LF?

Endsr;

Begsr Prompt;

  If NOT EmptySfl;

    SflDspCtl = *ON; // Display Subfile
    SflDsp = *ON;
    SflRrn = Rrn - 1;

  Else;

    SflDspCtl = *ON;  // No records to view - display message
    SflDsp = *OFF;
```

```
      SflBegin = *On;
      Message = 'No vendors found';
      Write Msg;
    Endif;

    // Write Header_Fmt;
    ExFmt VSearchCtl;
     // ExFmt Prompt_Fmt;
    If Rrn > 1; // Process changes only if subfile has records
       Exsr Changes;
    EndIf;
  Endsr;

  Begsr NextPage;

  // Load next block of records
    Exsr SflClear;
    Rrn = 1;
    Exsr Fill;
    Exsr Prompt;
  Endsr;

  Begsr PrevPage;
    // Find out where I am now in the DB
    Chain 1 VSearchDta; // Chain to first record in subfile
    Exsr SflClear;

    Setll VndName VndNam_LF;// Load previous block of records
    ReadP VndNam_LF;
    RrnCount  = 1;

    // Read back through the file one page
    DOW (NOT %EOF(VndNam_LF)) AND
        (RrnCount <= (SflSize));
      ReadP VndNam_LF;
      RrnCount = RrnCount + 1;
    Enddo;

     // Prevent PAGEUP if first record
    Exsr CheckBOF;
    Rrn = 1;
    Exsr Fill;
    Exsr Prompt;
  Endsr;

  Begsr SFLCLear;   // Subfile clear subroutine
    SflClr = *on;
    SflDsp = *OFF;
    SflDspCtl = *OFF;
    Write VSearchCtl;  // New search - clear subfile
    SflClr = *off;
    SflBegin = *OFF; // Set BOF of subfile
  EndSR;

  Begsr Changes;
    ReadC VSearchDta;
```

```
       // Process subfile changes
         Dow NOT %EOF(VndSchs5);
           Select;

       // Add new Vendor
           When Option = '1';
             SflDspCtl = *ON;  // display message
             SflDsp = *OFF;
             SflBegin = *ON;
             Message = 'Calling Add Program';
             Write Msg;
             Exfmt VSearchCtl;
      /End-free
     C                   Call(E)   'ADDPROGRAM'
      /Free
             If %ERROR;
       // Check if Add function completed successfully
             Endif;

       // Change Vendor details
           When Option = '2';
           Monitor;
            SflDspCtl = *ON;  // display message
            SflDsp = *OFF;
            SflBegin = *ON;
            Message = 'Calling Change Program';
            Write Msg;
            Exfmt VSearchCtl;
      /End-free
    C                   Call      'CHGPROGRAM'
    C                   Parm                    VndNbr
     /Free
            On-error 00211;
      // Check Change function completed successfully
              ErrorMsg = 'Change Program not found '
                         + %editw(%status:'0    ');
             Dsply ErrorMsg '*REQUESTER';
           EndMon;

       // Delete existing Vendor
           When Option = '4';
             SflDspCtl = *ON;  // display message
             SflDsp = *OFF;
             SflBegin = *ON;
             Message = 'Calling Delete Program';
             Write Msg;
             Exfmt VSearchCtl;

>>           CallP(E) VnrDelete (VndNbr);
             If %ERROR;
       // Check Delete function completed successfully
>>           SflDspCtl = *ON;  // display message
>>       SflDsp = *OFF;                          >>       SflBegin = *ON;
>>           Message = 'Delete Program not Found - Delete Fa
>>           Write Msg;
```

```
>>            Exfmt VSearchCtl;
              Endif;
            Endsl;

            ReadC VSearchDta;

          Enddo;
          Write FKey_Fmt;
        Endsr;
       /END-FREE
```

RPG IV program: VNRDLT

```
     FVendor_PF UF   E           K Disk
     FVndDlt    CF   E             Workstn

     D VnrDelete       PR                  ExtPgm('VNRDLT')
     D  VndNbr                        5  0

     D VnrDelete       PI
     D  VndNbr                        5  0

      /FREE
       Setll(E) VndNbr Vendor_PF;

       If not %error;
          If %equal(Vendor_PF);
             Delete VndNbr Vendor_PF;
             Message = 'Vendor number ' + %char(VndNbr) +
                       ' deleted successfully';
             ExFmt MsgWin;
          EndIF;
       Else;
          Message = 'Attempted to delete Vendor ' + %char(VndNbr) +
                    ' that does not exist on file';
          ExFmt MsgWin;
       EndIf;

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

Exercise 16: Subprocedures

RPG IV: LOANPAYSP

```
FLoanPayD  CF   E             WorkStn IndDS(LoanPDS)
D LoanPDS         DS
D  Exit                   3      3N
 /Copy RatPer_PR
 /Copy Paymnt_PR

 /free
      ExFmt PayFmt;

      DoW NOT Exit;
          RatePeriod = Ratper(RatePCAnn:NbrPayYr);
          PaymentAmt = Paymnt(Principal:RatePeriod:NbrPayTot);

          ExFmt PayFmt;
      EndDo;

      *InLR = *On;
      Return;
 /end-free
 /Copy RatPer
 /Copy Paymnt
```

RPG IV: Paymnt

```
PPaymnt           B

** PAYMNT - Calc loan payment SUBPROCEDURE

DPaymnt           PI             9  2
DPrincipal                       9  2
DRatePeriod                     13 11
DNbrPayTot                       4  0

 /Free

  Return (Principal*RatePeriod) /
         (1-(1/((1+RatePeriod)**NbrPayTot)));

 /End-free
PPaymnt           E
```

RPG IV: Paymnt_PR

```
DPaymnt           PR             9  2

 ** Calc loan payment PROTOTYPE

DPrincipal                       9  2
```

```
DRatePeriod                     13 11
DNbrPayTot                       4  0
```

RPG IV: Ratper

```
PRatPer           B

** RATPER - Calc dec periodic interest rate SUBPROCEDURE

D                 PI            13 11
DRatePCAnn                       5  3
DNbrPayYr                        2  0

 /Free

  Return ( RatePCAnn * 0.01 ) / NbrPayYr;

 /End-free

PRatPer           E
```

RPG IV: RatPer_PR

```
DRatPer           PR            13 11

** RATPER - Calc dec periodic interest rate PROTOTYPE

DRatePCAnn                       5  3
DNbrPayYr                        2  0
```

Compilation Listing:

```
1 FLoanPayD  CF   E             WorkStn IndDS(LoanPDS)
    *------------------------------------------------------------------------
    *                             RPG name         External name
    * File name. . . . . . . . . :  LOANPAYD         AS07V2LIB/LOANPAYD
    * Record format(s) . . . . . :  PAYFMT           PAYFMT
    *------------------------------------------------------------------------
 2 D LoanPDS         DS
 3 D  Exit                    3      3N
 4  /Copy RatPer_PR
    *------------------------------------------------------------------------
    * RPG member name  . . . . . :  RATPER_PR
    * External name  . . . . . . :  AS07V2LIB/QRPGLESRC(RATPER_PR)
    * Last change  . . . . . . . :  07/31/03  14:58:09
    * Text 'description' . . . . :  Ex 15 - Calc periodic interest rate PR
    *------------------------------------------------------------------------
 5+
 6+DRatPer           PR            13 11
 7+
 8+** RATPER - Calc dec periodic interest rate PROTOTYPE
 9+
10+DRatePCAnn                      5  3
11+DNbrPayYr                       2  0
12+
```

```
13  /Copy Paymnt_PR
    *----------------------------------------------------------------
    * RPG member name  . . . . . :  PAYMNT_PR
    * External name  . . . . . :  AS07V2LIB/QRPGLESRC(PAYMNT_PR)
    * Last change  . . . . . :  07/31/03  14:58:09
    * Text 'description' . . . . :  Ex 16 - Calc loan payment PROTOTYPE
    *----------------------------------------------------------------
14+
15+DPaymnt          PR             9  2
16+
17+ ** Calc loan payment PROTOTYPE
18+
19+DPrincipal                      9  2
20+DRatePeriod                    13 11
21+DNbrPayTot                     4  0
22+
23
24  /free
25=IPAYFMT
    *----------------------------------------------------------------
    * RPG record format  . . . . :  PAYFMT
    * External format  . . . . . :  PAYFMT : AS07V2LIB/LOANPAYD
    *----------------------------------------------------------------
26=I                            S    1    9 2PRINCIPAL
27=I                            S   10   14 3RATEPCANN
28=I                            S   15   16 0NBRPAYYR
29=I                            S   17   20 0NBRPAYTOT
30      ExFmt PayFmt;
31
32      DoW NOT Exit;
33          RatePeriod = Ratper(RatePCAnn:NbrPayYr);
34          PaymentAmt = Paymnt(Principal:RatePeriod:NbrPayTot);
35
36          ExFmt PayFmt;
37      EndDo;
38
39      *InLR = *On;
40      Return;
41  /end-free
42  /Copy RatPer
    *----------------------------------------------------------------
    * RPG member name  . . . . . :  RATPER
    * External name  . . . . . :  AS07V2LIB/QRPGLESRC(RATPER)
    * Last change  . . . . . :  07/31/03  14:58:09
    * Text 'description' . . . . :  Ex 15 - Calc periodic interest rate SU
    *----------------------------------------------------------------
43=OPAYFMT
    *----------------------------------------------------------------
    * RPG record format  . . . . :  PAYFMT
    * External format  . . . . . :  PAYFMT : AS07V2LIB/LOANPAYD
    *----------------------------------------------------------------
44=O                        PRINCIPAL           9S ZONE      9,2
45=O                        RATEPCANN          14S ZONE      5,3
46=O                        NBRPAYYR           16S ZONE      2,0
47=O                        NBRPAYTOT          20S ZONE      4,0
48=O                        RATEPERIOD         33S ZONE     13,11
```

```
49=O                       PAYMENTAMT          46S ZONE     13,2
50=O                       ERRMSG              86A CHAR       40
51+PRatPer           B
52+
53+** RATPER - Calc dec periodic interest rate SUBPROCEDURE
54+
55+D                  PI            13 11
56+DRatePCAnn                       5  3
57+DNbrPayYr                        2  0
58+
59+ /Free
60+
61+  Return ( RatePCAnn * 0.01 ) / NbrPayYr;
62+
63+ /End-free
64+
65+PRatPer           E
66+
67   /Copy Paymnt
     *----------------------------------------------------------------
     * RPG member name  . . . . . :  PAYMNT
     * External name  . . . . . . :  AS07V2LIB/QRPGLESRC(PAYMNT)
     * Last change  . . . . . . . :  07/31/03  14:58:09
     * Text 'description' . . . . :  Ex 16 - Calc loan payment SUBPROCEDURE
     *----------------------------------------------------------------
68+PPaymnt           B
69+
70+** PAYMNT - Calc loan payment SUBPROCEDURE
71+
72+DPaymnt           PI             9  2
73+DPrincipal                      9  2
74+DRatePeriod                    13 11
75+DNbrPayTot                      4  0
76+
77+ /Free
78+
79+  Return (Principal*RatePeriod) /
80+         (1-(1/((1+RatePeriod)**NbrPayTot)));
81+
82+ /End-free
83+PPaymnt           E
```

Exercise 17: Creating ILE objects

No solution necessary.

Exercise 18: Bind by copy

Step 1 -3 The Extpgm keyword on the prototype.

VNRDLTPROC - in the prototype, change the ExtPGM keyword to ExtPROC and the name of the procedure to VNRDLTPROC rather than VNRDLT

VNRSCHMAIN - in the prototype, change the ExtPGM keyword to ExtPROC and the name of the procedure to VNRDLTPROC rather than VNRDLT

Step 3-3 The VNRSCHMAIN procedure lists the VNRDLTPROC Module in the Imported (unresolved) symbols display. The import request would be resolved when we run CRTPGM.

Step 4-5 PEP is in module VNRSUBMAIN.

Step 6-3 VNRSCHMAIN.

Step 6-4 RPGLE

Step 6-5 ILE

Step 6-6 two modules (VNRSCHMAIN and VNRDLTPROC)

Step 6-7 four; all system modules in QSYS

Prototype is modified in both VNRDLTPROC and VNRSCHMAIN:

```
D VnrDelete       PR                  ExtProc('VNRDLTPROC')
D  VndNbr                        5  0
```

CRTPGM:

```
                          Create Program (CRTPGM)

Type choices, press Enter.

Program  . . . . . . . . . . . . > VNRSCHMAIN    Name
  Library  . . . . . . . . . . . >   AS07nnLIB   Name, *CURLIB
Module . . . . . . . . . . . . . > VNRSCHMAIN    Name, generic*, *PGM, *ALL
  Library  . . . . . . . . . . . >   AS07nnLIB   Name, *LIBL, *CURLIB...
            + for more values > VNRDLTPROC
                              >   AS07nnLIB
```

```
Text 'description' . . . . . .   *ENTMODTXT

                        Additional Parameters

Program entry procedure module     *FIRST       Name, *FIRST, *ONLY, *PGM
  Library  . . . . . . . . . . .                  Name, *LIBL, *CURLIB...
```

Exercise 19: Bind by reference

Step 1 - 2 *SRVPGM

Step 1 - 3 Error message; you cannot call a service program.

Step 2 - 1

```
CRTPGM PGM(AS07nnLIB/VNRSCHREF)
       MODULE(AS07nnLIB/VNRSCHMAIN)
       BNDSRVPGM(AS07nnLIB/MYSRVPGM)
```

Step 3 - 1 VNRSCHMAIN is the PEP; it is the first (and only) module referenced.