Lab 11

Objective: Builds on Lab 10 and allows the report title some flexibility by passing the type of shift being reported on to the program. The report title will be passed as a parameter and will indicate Day Shift, Night Shift, Afternoon Shift or All Shifts.

Changes:

1. PAYROLLPG2
2. PAYRPT2 (TITLE record include a named 30-character field)
3. CONFIRMPY2 (add the 30 character output only field of SHIFTTYPE)
4. Add code to support passing a parameter

To prototype a dynamic call to a program, you code the Extpgm keyword on the prototype (PR) definition, specifying the name of the program to call in uppercase letters within apostrophes (').

When coding the called program, you must specify both a prototype (PR) and a procedure interface (PI) in the definition specifications

In the calling program (e.g. PGM1), RPG IV uses the prototype definition to describe the list of parameters that it will pass on to the called program (e.g., PGM2). The called RPG IV program also uses a prototype definition to describe the corresponding list of parameters that it will receive from the caller. (Languages other than RPG IV may use mechanisms other than prototypes for this purpose.)

The prototypes in each program must match each other. That is, they must have the same number of parameters, and the corresponding parameters between the two programs must have the same data type and length. RPG IV passes parameter arguments by passing the address of the storage location represented by the field, so in fact these corresponding parameters reference the same storage location within the computer. The data names of the parameters used in the caller and callee need not be the same.

To access those storage locations passed by the calling program, the called program must include a **procedure interface** definition (in addition to a prototype). Like the prototype, the procedure interface describes the parameters that the callee will share with the caller. Unlike the prototype, however, the procedure interface defines variable names to hold the parameter values that the called program will receive, letting you refer to and manipulate those parameter values.

> **Procedure interface**: Program lines within a subprocedure that define the data type and size of the procedure's return value and the parameters the procedure will use.

ILE RPG for AS/400 Reference

The following is an example of a prototype for a program call interface:

```
*.. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8
DName++++++++++ETDsFrom+++To/L+++IDc.Functions++++++++++++++++++++++++++++++
D UpdCust         PR                  EXTPGM('AR003')
D                                5
D                                7  0
```

PR, coded in positions 24–25 (Ds) signals the beginning of a prototyped call interface. You must name the prototype in positions 7–21 (Name++++++++++). Code the external name of the program associated with this prototype, using the EXTPGM keyword.
In the example above, when you perform a call to the UpdCust prototype, program AR003 will actually be called.
If the call interface involves passing parameters, describe the parameters on subsequent lines, following the PR header. Indicate a parameter's length in positions 33–39 (To/L+++); for numeric fields, code a decimal positions entry in positions 41–42 (Dc). Code the parameter's data type in position 40 (I); if the type entry is blank, the default is packed decimal for numeric fields (fields with a decimal positions entry) and character for fields with blanks in positions 41–42. In the sample prototype above, there are two parameters: a five-character field and a seven-digit packed decimal field with zero decimals. Note that you need not name the parameters. You may find it convenient to document the parameter usage by naming the parameters (in positions 7–21), but the compiler will treat these names as comments.

To access those storage locations passed by the calling program, the called program must include a **procedure interface** definition (in addition to a prototype). Like the prototype, the procedure interface describes the parameters that the callee will share with the caller. Unlike the prototype, however, the procedure interface defines variable names to hold the parameter values that the called program will receive, letting you refer to and manipulate those parameter values. The following is an example of a prototype and a matching procedure interface:

```
*.. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8
DName++++++++++ETDsFrom+++To/L+++IDc.Functions++++++++++++++++++++++++++++++
D Main            PR                  EXTPGM('AR003')
D                                5
D                                7  0
D Main            PI
D  Company                       5
D  Customer                      7  0
```

PI, coded in positions 24–25 (Ds) signals the beginning of a prototyped call interface. You must name the prototype in positions 7–21 (Name++++++++++). This name must match the name in the callee's prototype (but not necessarily the name for the caller's prototype).

Meyers, Bryan, and Judy Yaeger. "Chapter 10 - Modular Programming Concepts". *Programming in RPG IV, Third Edition*. 29th Street Press. © 2002. Books24x7. <*http://lcweb.senecac.on.ca:2121/toc.aspx?bookid=5706*>

```
FMT FX FFilename++IPEASF.....L.....A.Device+.Keywords++++++++++++++++++++
0001.00 FSHIFTRATESIF   E             DISK    RENAME(SHIFTRATES:SHIFTRATER)
0002.00 FALLSHIFT  IF   E           K DISK    RENAME(ALLSHIFT:ALLSHIFTR )
0003.00 FCONFIRMPY2CF   E             WORKSTN
0004.00 FPAYRPT2   O    E             PRINTER OFLIND(*IN01)
0005.00 D Main            PR                  EXTPGM('PAYROLLPG2')
0006.00 D  ShiftType                    30
0007.00 D Main            PI
0008.00 D  ShiftType                    30
0009.00 D  HOURSOVER      S              3  0
0010.00  /Free
0011.00                     READ ShiftRates;
0012.00                     Write Title;
0013.00                     Write ColHdg;
0014.00                     READ ALLSHIFT;
0015.00                     DOW  NOT %EOF;
0016.00                     EXSR PAYSR;
0017.00                     IF *IN01;
0018.00                         Write Title;
0019.00                         Write ColHdg;
0020.00                         *IN01 = *Off;
0021.00                     ENDIF;
0022.00                     Write  EmpDetail;
0023.00                     READ ALLSHIFT;
0024.00                     ENDDO;
0025.00                     TOTEMPPAY  = TOTREGPAY  + TOTOVTPAY;
0026.00                     WRITE TOTALS;
0027.00                     EXFMT RECORD1;
0028.00                     *INLR = *ON;
0029.00                     RETURN;
0030.00     BEGSR  PAYSR;
0031.00                     SELECT;
0032.00                     WHEN WORKSHIFT = 'D';
0033.00                          HOURLYRATE = DAYRATE;
0034.00                     WHEN WORKSHIFT = 'N';
0035.00                          HOURLYRATE = NIGHTRATE;
0036.00                     WHEN WORKSHIFT = 'A';
0037.00                          HOURLYRATE = AFTNRATE;
0038.00                     ENDSL;
0039.00                     SELECT;
0040.00                     WHEN PAYGRADE = '1';
0041.00                          EVAL(H) HOURLYRATE = HOURLYRATE * 1.075;
0042.00                     WHEN PAYGRADE = '2';
0043.00                          EVAL(H) HOURLYRATE = HOURLYRATE * 1.035;
0044.00                     WHEN PAYGRADE = '3';
0045.00                          EVAL(H) HOURLYRATE = HOURLYRATE * .96;
0046.00                     ENDSL;
0047.00                     IF   HRSWORKED <= 40;
0048.00                          OVERPAY = 0;
0049.00                          HOURSOVER = 0;
0050.00                          EVAL    REGULARPAY = HRSWORKED * HOURLYRATE;
0051.00                     ELSE;
0052.00                          HOURSOVER = HRSWORKED - 40;
0053.00                          EVAL(H) OVERPAY = HOURSOVER * (1.5 * HOURLYRATE);
0054.00                          EVAL    REGULARPAY = 40 * HOURLYRATE;
0055.00                     ENDIF;
0056.00                     TotalPay = RegularPay + OverPay;
0057.00                     TOTREGPAY  = TOTREGPAY  + REGULARPAY;
0058.00                     TOTOVTPAY  = TOTOVTPAY  + OVERPAY;
0059.00                     ENDSR;
       ****************** End of data ************************************
```

```
       ************** Beginning of data *********************************
0000.01      PGM
0001.00      DCL        &Shift      *CHAR 1
0002.00      DCL        &ShiftType  *CHAR 30
0003.00      DCL        &OutQ       *CHAR 10
0004.00      DCL        &OutQLib    *CHAR 10
0005.00
0006.00      RTVUSRPRF  OUTQ(&OUTQ) OUTQLIB(&OUTQLIB)
0007.00 /*   CLROUTQ &OutQLib/&OutQ   destroys user's output & causes +
0008.00          CPF2207     Not authorized to use object */
0009.00      ADDLIBLE SENECAPAY
0010.00      MONMSG     MSGID(CPF0000)
0011.00      SNDUSRMSG  MSG('1 - Day Shift, 2 - Night Shift, 3 - +
0011.01                          Afternoon Shift, 4 - All Shifts 5 - EXIT  +
0011.02                          Note this program clears output queues - +
0011.03                          don''t do this in industry') +
0011.04                          MSGRPY(&SHIFT)
0013.00      DOWHILE    (&Shift *NE '5')
0014.00           Select
0015.00           When (&Shift = '1') Do
0016.00              CHGVAR     &ShiftType 'D A Y   S H I F T'
0017.00              OVRPRTF    FILE(PAYRPT2) SPLFNAME(DAYSHIFT)
0018.00              OVRDBF     ALLSHIFT  DAYS
0019.00              CALL       PGM(PAYROLLPG2) PARM(&SHIFTTYPE)
0020.00              DSPSPLF    FILE(DAYSHIFT) SPLNBR(*LAST)
0021.00           ENDDO
0022.00           When (&Shift = '2') Do
0023.00              CHGVAR     &ShiftType 'N I G H T   S H I F T'
0024.00              OVRPRTF    FILE(PAYRPT2) SPLFNAME(NIGHTSHIFT)
0025.00              OVRDBF     ALLSHIFT   NIGHTS
0026.00              CALL       PAYROLLPG2 Parm(&ShiftType)
0027.00              DSPSPLF    FILE(NIGHTSHIFT) SPLNBR(*LAST)
0028.00           ENDDO
0029.00           When (&Shift = '3') Do
0030.00              CHGVAR     &ShiftType  'A F T E R N O O N   S H I F T'
0031.00              OVRPRTF    FILE(PAYRPT2) SPLFNAME(AFTRNSHIFT)
0032.00              OVRDBF     ALLSHIFT   AFTERNOONS
0033.00              CALL       PAYROLLPG2 Parm(&ShiftType)
0034.00              DSPSPLF    FILE(AFTRNSHIFT) SPLNBR(*LAST)
0035.00           ENDDO
0036.00           When (&Shift = '4') Do
0037.00              CHGVAR     &ShiftType  'A L L   S H I F T S'
0038.00              OVRPRTF    FILE(PAYRPT2) SPLFNAME(ALLSHIFTS)
0039.00              OVRDBF     ALLSHIFT   ALLSHIFT
0040.00              CALL       PAYROLLPG2 Parm(&ShiftType)
0041.00              DSPSPLF    FILE(ALLSHIFTS) SPLNBR(*LAST)
0042.00           ENDDO
0043.00           When ( &Shift *NE '5')    DO
0044.00             SndUsrMsg MSG(&Shift *CAT ' is an Invalid selection. Press the Enter Key')
0045.00           ENDDO
0046.00           ENDSELECT
0047.00           SNDUSRMSG  MSG('1 - Day Shift, 2 - Night Shift, 3 - Afternoon Shift, +
0048.00                         4 - All Shifts 5 - EXIT') MSGRPY(&SHIFT)
0049.00      ENDDO
0049.01              SNDPGMMSG  MSG('Okay to clear out an output queue in +
0049.02                           school - Never at work in production')
0049.03
0050.00      WRKOUTQ &OutQLib/&OutQ
0051.00   ENDPGM
       ***************** End of data **************************
```

```
                                                                  DAYS
SEU==>
 FMT LF .....A.........T.Name++++++.Len++TDpB......Functions++++++++++++++++
        *************** Beginning of data ***********************************
0001.00      A          R ALLSHIFT                  PFILE(SENECAPAY/ALLSHIFT)
0002.00      A          K HRSWORKED
0003.00      A          S WORKSHIFT                 COMP(EQ 'D')
        ****************** End of data ************************************
```

```
SEU==>                                                            AFTERNOONS
 FMT LF .....A.........T.Name++++++.Len++TDpB......Functions++++++++++++++++
        *************** Beginning of data ***********************************
0001.00      A          R ALLSHIFT                  PFILE(SENECAPAY/ALLSHIFT)
0002.00      A          K HRSWORKED
0003.00      A          S WORKSHIFT                 COMP(EQ 'A')
        ****************** End of data ************************************
```

RUNPAYPGM2 and PAYROLLPG2 were created with CRTBND? commands and a CALL statement is used to invoke the RPGLE program. That means at run time extra work is required to link the two programs and provide support for the parameter passing. This must occur every time the RUNPAYPGM2 program is run. IBC233 Lab 11

This is referred to as a dynamic call.

RUNPAYPGM3 and PAYROLLPG3 were created with CRT???MOD commands and a CALLPRC statement is used to invoke the RPGLE program module. The linking of the two programs and resolving addresses etc can now take place at a CRTPGM step done from the command line a single time. It does not occur every time the RUNPAYPGM3 program is run.

This is referred to as a static call.

The program you want to demonstrate will be the one that combines two modules.
The steps to produce the two modules are selecting the following compile commands in RDp
CRTCLMOD – for the CLLE program
CRTRPGMOD – for the RPGLE program

Combining the modules would involve the following one time command:
===> CRTPGM PGM(RUNPAYPGM3) MODULE(RUNPAYPGM3 PAYROLLPG3)

Extra Notes:

PI Procedure interface
PR Prototype
S Standalone field

EXTPGM(name)
The EXTPGM keyword indicates the external name of the program whose prototype is being defined. The name can be a character constant or a character variable.
When EXTPGM is specified, then a dynamic call will be done.
If neither EXTPGM or EXTPROC is specified, then the compiler assumes that you are defining a prototype for a procedure, and assigns it the external name found in positions 7-21.
Any parameters defined by a prototype with EXTPGM must be passed by reference.
In addition, you cannot define a return value.

EXTPROC(name)
The EXTPROC keyword indicates the external name of the procedure whose prototype is being defined. The name can be a character constant or a procedure pointer. When EXTPROC is specified, then a bound call will be done.
If neither EXTPGM or EXTPROC is specified, then the compiler assumes that you are defining a procedure, and assigns it the external name found in positions 7-21.

But you can also use PR/PI definitions to replace the *ENTRY PLIST for an RPG IV program When coding a procedure interface for a main procedure (i.e., a program), you must remember a few additional requirements:

The prototype for the main procedure must include the Extpgm keyword.

The procedure interface must be named (the same as the prototype).

The prototype must precede the procedure interface.

You have a choice when naming the prototype and the procedure interface. You can name the PR and PI descriptions to match the name of the module, or you can assign a name of your choosing -- perhaps a shop-standard name such as Main. If the name on the PR/PI descriptions doesn't match the module name, you must specify the module name with the Extpgm keyword (e.g., EXTPGM('MYPGM')).