The Trainer's Friend

9355 E Center Ave., #8C
Denver, Colorado 80247
U.S.A.

Telephone: (303) 355-2752

E-mail steve@trainersfriend.com
Internet www.trainersfriend.com

Advanced Topics in COBOL

Student Materials

The following terms that may appear in these course materials are trademarks or registered trademarks:

Trademark s and registered trademarks of International Business Machines, Inc.:

BookMaster, CICS, DB2, Enterprise Systems Architecture/370, Enterprise Systems Architecture/390, Enterprise Systems Connection Architecture, ESA/370, ESA/390, FICON, IBM, IBM Z, IBM Z family, IBM Z systems, IBM z13, IBM z13s, IBM z14, IBM z15, IBM z16, IBM zSystems, IBMLink, ibm.com, IMS, Language Environment, MQSeries, MVS, NetView, AIX, AS/400, BookManager, BookMaster, CICS, BladeCenter, DB2, DS8000, developerWorks, IMS, Language Environment, MQSeries, MVS, PR/SM, Processor Resource / Systems Manager, OS/390, Parallel Sysplex, RACF, Redbooks, S/390, System/360, System/370, System/390, System z, System z9, System z10, z/OS, z/Architecture, zEnterprise, zSeries, OS/390, Parallel Sysplex, RACF, Redbooks, S/390, System/390, System z, System z9, System z10, z/OS, z/Architecture, zEnterprise, zSeries, z9, z10, z13, z14, z15, z16,

Trademarks and registered trademarks of Microsoft Corp.: Excel, Internet Explorer, JScript, Microsoft, SQL Server, Windows

Trademark of American National Standards Institute: ANSI

Trademark of Phoenix Software International: (E)JES

Trademark of Fischer International Systems: IOF

Trademark of TONE Software: OMC-FLASH

Trademarl of Syncsort Corp.: SyncSort

Registered Trademark of The Open Group: UNIX

Registered Trademark of Linus Torvalds: LINUX

Registered Trademarks of Institute of Electrical and Electronic Engineers: IEEE, POSIX

Registered Trademark of Unicode, Inc.: Unicode

Trademarks held on behalf of World Wide Web Consortium: W3C, HTML, HTTP, DOM, CSS, XML, XHTML, XSL, WebFonts, Amaya

Trademarks of Adobe Systems, Inc.: Macromedia, PDF, Shockwave, Flash

The content of this publication is copyright © 2024 by Steven H. Comstock.

Advanced Topics in COBOL (Enterprise COBOL, z/OS) - Course Objectives

On successful completion of this class, the student, with the aid of the appropriate reference materials, should be able to:

1. Code COBOL mainline programs that CALL subroutines, passing arguments as necessary and COBOL subroutine programs that are CALLed, receiving any passed parameters

2. Explain the tradeoffs in deciding whether to use static or dynamic linkages when using external subroutines

3. Define and process tables, including:

   Initializing tables using VALUE or REDEFINES clauses, the INITIALIZE statement, loops, and extracting data from records in a file

   Working with tables of more than one dimension

4. Process tables using subscripting, indexing, or mixing subscripting and indexing; also using relative subscripting and relative indexing

5. Use the SET, SEARCH, and SEARCH ALL verbs as they apply to indexed tables

6. Use the ALL subscript capability of intrinsic functions

7. Work with variable length records in sequential files

8. Use the string handling features of COBOL, including:
   a. Reference modification (sub-stringing)
   b. Hexadecimal notation
   c. LENGTH OF special register and the LENGTH intrinsic function
   d. INSPECT, STRING, and UNSTRING verbs
   e. Null-terminated strings

9. Use some advanced features of the newest IBM COBOL compilers, including

   a. Local-storage section
   b. Recursive programs
   c. Pointers, procedure-pointers, function-pointers, Address Of special register
   d. Dynamic file allocation

10. (Optionally) use the COBOL SORT and MERGE verbs.

These Materials Copyright 2024 by Steven H. Comstock

Advanced Topics in COBOL (Enterprise COBOL, z/OS) - Topical Outline

Day One

These Materials Copyright 2024 by Steven H. Comstock

Day Two

 These Materials Copyright 2024 by Steven H. Comstock

Day Three

 These Materials Copyright 2024 by Steven H. Comstock

Section Preview

- **Introduction to Subroutines**
  - **What Are Subroutines?**
  - **Invoking Subroutines - CALL**
  - **Leaving a CALLed Program**
  - **Passing Arguments and Receiving Parameters**
  - **A Mainline and Subroutines (Machine Exercise)**

What Are Subroutines?

❒ **A subroutine is a self-contained program designed to be used by another program**

Each subroutine is designed to accomplish a particular function

✗ Often this function is used by many different programs, so a subroutine is typically used by all of these programs

✗ A good reason to have a subroutine, then, is to write the code to accomplish the necessary function once, so each programmer that needs the function does not need to reinvent the wheel

✗ All users of the subroutine always get the same results, and maintenance of the function can be isolated to the single, common subroutine

❒ **COBOL supports two kinds of subroutines**

External subroutines

Internal subroutines (nested programs)

❒ **Typically, each installation will have a library of subroutines that perform functions needed in the kind of work that installation does**

Invoking Subroutines

❒ **A program uses a subroutine by the CALL verb**

Usually passing one or more arguments (values or variables used as inputs to the subroutine)

The subroutine uses the passed values to accomplish its particular function

✗ One or more of the passed variables might be modified, if appropriate to the function of the subroutine

✗ Usually a value is returned to the invoking program, either in one of the variables passed to the subroutine, or in a special register, RETURN-CODE

➢ This returned value generally indicates if the subroutine's function was performed without error or if some actual or potential problem was encountered

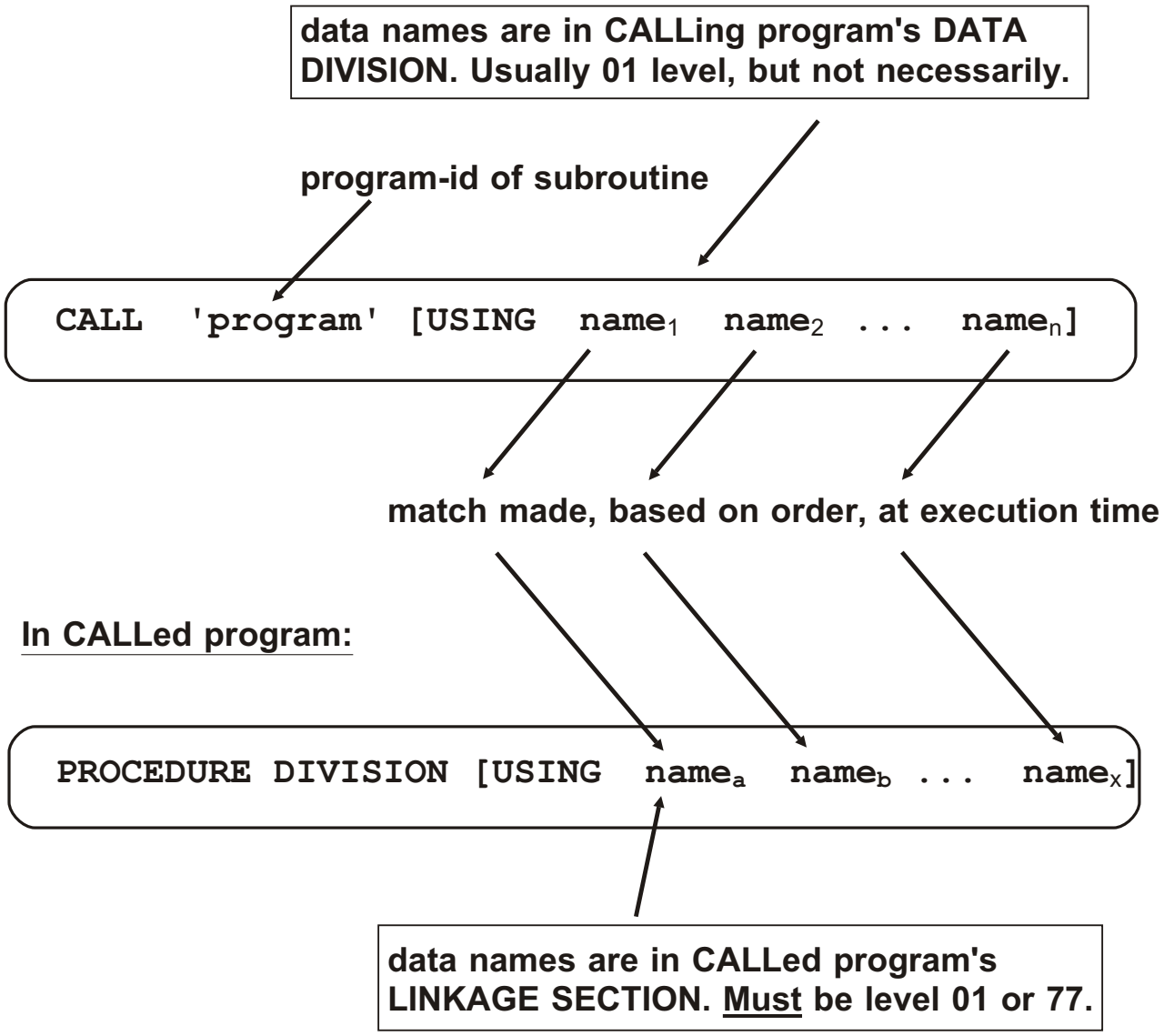
➢ The expected values and their interpretation are decided on by the person coding the subroutine, there are no universal values and meanings, although a returned value of zero almost always means a successful completion

❒ **Subroutines may in turn call other subroutines**

❒ **A mainline program is the top of the calling chain: it is not called, but it calls one or more subroutines which in turn may call additional subroutines**

Subroutine Coding

In CALLing program:



Code a LINKAGE SECTION after a WORKING-STORAGE SECTION, if any

Leaving a CALLed Program

❒ **Return to CALLing Program via:**

```
GOBACK
```

Returns control out of this program to the calling program

If issued from a main program, control returns to host operating system

```
STOP RUN
```

Returns control out of this program directly to host operating system

If issued from a subroutine, you will not get back to the program that called the subroutine

```
EXIT PROGRAM
```

Control returns to the program that CALLed this program

If this program was not CALLed, this statement is ignored (in other words, this statement has no effect in a main program)

Notes on Leaving Any Program

❒ **EXIT PROGRAM may be used for a subroutine**

❒ **STOP RUN may be used for a main program**

Note if you issue STOP RUN in a subroutine, the whole run unit is terminated without returning to the calling program

✗ Only appropriate if an error is detected or the application is designed to come to an abrupt halt for some other reason

❒ **GOBACK may be used for either a subroutine or a main program**

❒ **GOBACK was not part of the ISO/ANSI standards until the 2002 standard; prior to that, it was an IBM extension to the ANSI standard**

❒ **If the last physical instruction is executed and there is no STOP RUN, EXIT PROGRAM, or GOBACK, the program issues an implicit EXIT PROGRAM**

This is OK in a subroutine, but can be a problem in a main program (use STOP RUN or GOBACK)

CALL “BY CONTENT”

```
CALL 'SUBX' USING TRANS-REC
    BY CONTENT 'UPDATE'  'FILENAME'  FLAG-DATA
    BY REFERENCE  MASTER-REC  EDIT-NOTES
    BY CONTENT  'PGM005'  XYZ-AR
```

```
CALL 'CBLTDLI' USING BY CONTENT 'GU  '
    BY REFERENCE PCB-AREA IO-AREA SSA-AREA
```

- “BY CONTENT” and “BY REFERENCE” apply to subsequent arguments until another “BY” phrase occurs

- “BY REFERENCE” is the default, and reflects the way arguments have been passed in COBOL historically:

```
CALL 'CALCWIT' USING PERSON, RESULT
```

More on Passing Arguments

- **For each argument passed “BY REFERENCE”, COBOL passes a pointer to the actual data item to the subroutine ("by reference indirect")**

  Literal values may not be passed BY REFERENCE

  Changes to passed arguments by the subroutine are reflected in the CALLing program's corresponding arguments: they are the same location in memory

- **For each argument passed “BY CONTENT” COBOL passes a pointer to a work area containing the value of an alphanumeric literal or the value in a named data item ("by value indirect")**

  Any changes by the CALLed program do not affect the value in the CALLing program

BY REFERENCE passes the item

BY CONTENT passes a copy of the item

- **The pointers are gathered together in a list and the address of this list of pointers to arguments is passed to the CALLed program**

- **Vocabularly police alert: CALL passes arguments; the subroutine receives parameters - same data, just different names**

A Subroutine - Example

- **This simple subroutine expects to be passed a payroll record (which contains the current gross earnings and deductions for a person), and a field to place the result of the calculation:**
  amount-due = gross - deductions

Notice the calculated value is placed into `'result'` - the second argument passed to the program

```
Identification Division.
Program-id. CALCWIT.
Environment Division.
Data Division.
Linkage Section.
01  Person.
    02            Pic x(30).
    02  Gross     Pic S9(7)V99.
    02  Deducts   Pic S9(5)V99.
    02            Pic x(241).
01  Result        Pic S9(7)V99.
Procedure Division using Person, Result.
    Compute result = Gross - Deducts
    Exit program.
```

Notice, too, the Procedure Division header has a USING clause as described a few pages ago

A Mainline Program That Calls a Subroutine

❒ **This program uses CALCWIT to calculate the amount due for an employee's pay check ...**

```
Identification Division.
Program-id. PAYROLL.
Environment Division.
.
.
Input-Output section.
File-Control.
    Select People assign to --------.
    Select Checks assign to --------.
Data Division.
File Section.
FD  People
    Block Contains 0 records.
01  Personnel-Record.
    02  Employee-id  Pic x(7).
.
.
    02  Gross-income Pic S9(7)V99.
    02  Deductions   Pic S9(5)V99.
.
.
FD  Checks.
.
.
```

A Mainline Program That Calls a Subroutine, p.2.

Notice on the CALL, this program calls the second argument "Amount-due", while in CALCWIT the second parameter is called "Result"

The CALL process makes the connection positionally

```
Working-Storage Section.
77  Amount-due     Pic S9(7)V99.
.
.
Procedure Division.
Mainline.
    Open input People, output Checks
    Perform read-people
    Perform Write-checks
                   until no-more-records
    Stop run.
.
.
Write-checks.
.
.
    Call 'CALCWIT' Using Personnel-record,
                                    Amount-due
.
.
```

Subroutine Notes

❒ **Notice how important it is for the arguments being passed to match the parameters being caught**

The pictures, size, and data types should be defined the same in both the CALLing routine and the CALLed routine

The order of the arguments passed must match the order of the parameters received

❒ **Notice that you may pass an element in a structure instead of the entire structure, if that is appropriate**

Current thinking is that it is better to pass just the data elements being worked on

✗ This can simplify maintenance if structures change

❒ **COBOL allows a program to pass a file**

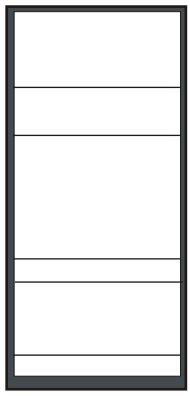
However, there is no way for a COBOL subroutine to receive a file

Files may be shared among COBOL programs through the use of the EXTERNAL attribute, discussed later

Internal vs. External Subroutines

❒ **If a subroutine is internal, it is nested in the program that calls it,**

The calling program and its nested subroutines are compiled and bound all together at the same time - they are in the same source code

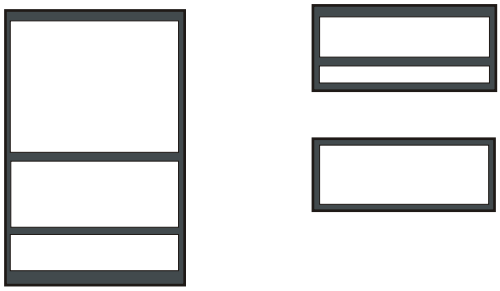


❒ **If a subroutine is external, the calling program and its subroutines must be compiled separately**

The subroutines may be bound together with the calling program (static linkage)

Or they may be bound separately, to be brought together as needed at run time (dynamic linkage)

Or you may mix and match, some static calls and some dynamic calls



Computer Exercise: A Mainline and Subroutines

Code a program that reads records from two similar files and produces a report that contains lines from records from both files.

The two files, informally called INPUTA and INPUTX, each contain inventory information from different warehouses. The record layouts are the same, and the files contain the same number of records. Each record represents an item maintained in inventory.

Basically, your program should read one record from each file, and format and write the two records to your report file, one after the other.

To do the actual formatting, we'll use a subroutine called FORMATIT. This subroutine expects to receive four parameters:

* A record from INPUTA or INPUTX
* A one-character code, "A" or "X", indicating which file the record is from (use a literal value)
* A place to put the formatted output (use "reprt-record")
* A two-field structure containing accumulator fields that will keep track of values for totals (use "totallers")

The subroutine FORMATIT will, in turn, call a subroutine called VALU to calculate the value of the current inventory item. The value is calculated as the item unit-price times the item quantity-on-hand, rounded. VALU should be passed the unit-price, the quantity-on-hand, the accumulator field structure ("totallers"), and the report line layout ("reprt-record"). VALU should calculate the item-value, move it to reprt-item-value, and add it to page-value.

The mainline, which is called SC4CMPR, is supplied as a skeleton, as are the subroutines FORMATIT and VALU . The skeletons are shown on the following pages. You need to:

* Finish coding VALU then compile and link it
* Finish coding FORMATIT, then compile and link it
* Finish coding SC4CMPR, then compile, link, and run it.

Computer Exercise: A Mainline and Subroutines, 2

We have provided a setup procedure to simplify running labs for this course. You need to run the rexx exec we have provided called "D725STRT". To run this dialog, from ISPF option 6, key in:

```
===> ex   '________.train.library(d725strt)'   exec
```

and press <Enter>. This rexx exec will ask you for the high level qualifier to use for your data set names, with the default being your TSO id (which is probably fine).

As a result of running D725STRT, three data sets are created for the labs:

<hlq>.TR.CNTL - JCL for running jobs
<hlq>.TR.COBOL - COBOL skeleton code, and your programs
<hlq>.TR.LOAD - load module library used to hold your executable programs.

where "<hlq>" represents your data set name high level qualifier

The setup will have copied the skeleton and starter programs needed for all the labs, including SC4CMPR, FORMATIT, and VALU, into your TR.COBOL library, so this is where you code your solutions.

In your TR.CNTL library, you will find a member called COBSUB, which is a job for compiling and linking subroutines [you need to change the **SET O=** line to the name of the program you are compiling]. You will also find a member called D725RUN1, which compiles, links, and runs your mainline, automatically linking in your subroutines.

Computer Exercise: A Mainline and Subroutines, 3

To summarize your steps:

One time only:

1. Run the setup exec

After you have coded VALU, and each time you make a change to your copy of VALU (in TR.COBOL):

1. Edit COBSUB and change the SET O= JCL statement to be SET O=VALU

2. Submit this job; check that it had a clean compile and link

Each time you change your copy of FORMATIT (in TR.COBOL)

1. Edit COBSUB and change the SET O= JCL statement to be SET O=FORMATIT

2. Submit this job; check that it had a clean compile and link

Each time you change your copy of SC4CMPR (or if you need to test SC4CMPR after changing FORMATIT or VALU) (in TR.COBOL)

1. In TR.CNTL, submit member D725RUN1

2. Check you had a successful compile, link, and test

After the listings of the provided programs is a sample of the expected output.

Code Supplied as SC4CMPR

```
 process dynam
 Identification division.
 program-id.  sc4cmpr.
*  Copyright (C) 2024 by Steven H. Comstock                    Ver2

 environment division.
 input-output section.
 file-control.
     select inputa assign to inputa.
     select inputx assign to inputx.
     select reprt assign to reprt.

 data division.
 file section.
 fd  inputa
     block contains 0 records.
 01  inputa-record                      pic x(100).

 fd  inputx
     block contains 0 records.
 01  inputx-record                      pic x(100).

 fd  reprt.
 01  reprt-rec                          pic x(106).

 working-storage section.
 01  a-record                pic x(100).

 01  x-record                pic x(100).
```

Code Supplied as SC4CMPR, p.2.

```
*
*  headers for reprt
*
 01  inq-list-head-1.
     05                        pic x      value spaces.
     05                        pic x(57)
            value 'Inventory - Data List'.
     05              pic x(5)              value 'Date '.
     05 rpt-date.
        10 mo         pic 99.
        10            pic x                 value '/'.
        10 da         pic 99.
        10            pic x                 value '/'.
        10 yr         pic 9999.
     05              pic x(6)              value spaces.
     05              pic x(6)              value 'Page: '.
     05 rpt-page-no   pic 99.
     05                        pic x(19)   value spaces.

 01  inq-list-head-2.
     05              pic x(5)              value spaces.
     05              pic x(41)
             value  'Part       Product                          '.
     05              pic x(30)
             value  '              Number    Number'.
     05              pic x(25)
             value  '  Reorder    Inventory'.
     05              pic x(05)             value spaces.

 01  inq-list-head-3.
     05              pic x                 value spaces.
     05              pic x(4)              value ' Id '.
     05              pic x(41)
             value  'Number     Description                    '.
     05              pic x(34)
             value  '  Unit price  On hand   On order'.
     05              pic x(26)
             value  'Level        value   Flags'.
     05              pic x(04)             value spaces.
```

Code Supplied as SC4CMPR, p.3.

```
*
 01  reprt-record.
     05                          pic xx.
     05 reprt-file-id            pic x.
     05                          pic xx.
     05 reprt-part-number        pic x(9).
     05                          pic x(2).
     05 reprt-description        pic x(30).
     05                          pic x(3).
     05 reprt-unit-price         pic z,zz9.999.
     05                          pic x(3).
     05 reprt-quantity-on-hand   pic zz,zz9.
     05                          pic x(5).
     05 reprt-quantity-on-ord    pic zz9.
     05                          pic x(5).
     05 reprt-reorder-level      pic zz9.
     05                          pic x.
     05 reprt-item-value         pic zzz,zzz,z99.99.
     05                          pic x(5).
     05 reprt-flag               pic x.
     05                          pic x.

 01  end-of-report.
     02              pic x(001)          value all ' '.
     02              pic x(044)          value all '*'.
     02              pic x(015)          value ' End of Report '.
     02              pic x(046)          value all '*'.

 01  totallers.
     05  page-no-flags   pic s9999        binary.
     05  page-value      pic s9(7)v99     packed-decimal.
```

Code Supplied as SC4CMPR, p.4.

```
*
*  variables associated with reprt listing
*
 01  inq-list-misc.
     05  inq-list-line-count pic s99       binary  value  +0.
     05  inq-list-max-lines  pic s99       binary  value +55.
     05  inq-list-space      pic 9                 value   2.
     05  pge-ctr             pic s99       comp-3  value  +1.
     05  rpt-no-flags        pic s9999             value  +0.
     05  rpt-value           pic s9(7)v99  comp-3  value   0.
     05  grand-title     pic x(30)   value 'Report totals:'.
     05  in-date.
         10 yr        pic 9999.
         10 mo        pic 99.
         10 da        pic 99.

 77  more-records               pic x     value 'Y'.
     88  no-more-records                  value 'N'.

 01  total-line.
     02              pic x(41)             value spaces.
     02 total-title  pic x(41)  value 'Page totals:'.
     02 total-value  pic z,zzz,z99.99.
     02              pic xx                value spaces.
     02 total-flags  pic zzz9.
     02              pic x(6)              value spaces.
```

Code Supplied as SC4CMPR, p.5.

```
 procedure division.
 mainline.
     open input inputa inputx output reprt
     move function current-date(1:8) to in-date
     move corr in-date to rpt-date
     move spaces to reprt-record
     move zeros to page-no-flags, page-value
     perform list-new-page
     perform read-2-records
     perform until no-more-records
*************************************************************
*
*  insert a call to 'formatit', using a-record, 'A',
*                                     reprt-record, and
*                                     totallers
*
*************************************************************
        write reprt-rec from reprt-record
*************************************************************
*
*  insert a call to 'formatit', using x-record, 'X',
*                                     reprt-record, and
*                                     totallers
*
*************************************************************
        write reprt-rec from reprt-record
        add 2 to inq-list-line-count
        if inq-list-line-count > inq-list-max-lines
           perform page-totals-print
           perform list-new-page
        end-if
        perform read-2-records
     end-perform

     perform grand-totals-print
     close inputa inputx reprt
     stop run.
```

Code Supplied as SC4CMPR, p.6.

```
read-2-records.
    read inputa into a-record
        at end set no-more-records to True
    end-read
   read inputx into x-record
        at end set no-more-records to True
    end-read.

list-new-page.
    move pge-ctr to rpt-page-no
    write  reprt-rec from inq-list-head-1 after advancing page
    write  reprt-rec from inq-list-head-2    after advancing 2
    write  reprt-rec from inq-list-head-3    after advancing 1
    add 1 to pge-ctr
    move 1 to inq-list-line-count
    move 2 to inq-list-space.

page-totals-print.
    move page-no-flags to total-flags
    move page-value to total-value
    write reprt-rec from total-line after advancing 2
    add page-no-flags to rpt-no-flags
    add page-value to rpt-value
    move zeros to page-no-flags, page-value.

grand-totals-print.
    if inq-list-line-count > 1
          perform page-totals-print
    end-if
    move grand-title to total-title
    move rpt-no-flags to total-flags
    move rpt-value to total-value
    write reprt-rec from total-line after advancing 2.
```

Code Supplied as FORMATIT

```
 process dynam
 Identification division.
 program-id.  formatit.
*  Copyright (C) 2024 by Steven H. Comstock                    Ver2
 data division.
*
 linkage section.
 01  f-record.
     05 f-part-number        pic x(9).
     05 f-description        pic x(30).
     05 f-mixed-string       pic x(5).
     05 f-unit-price         pic s9999v999 packed-decimal.
     05 f-quantity-on-hand   pic s99999    packed-decimal.
     05                      pic x.
     05 f-quantity-on-ord    pic s999      binary.
     05 f-reorder-level      pic s999      binary.
     05 f-switch             pic x.
     05 f-old-part-number    pic x(9).
     05 f-switch             pic x.
     05 f-category           pic x(10).
     05                      pic x(23).

 01  f-letter               pic x.
```

Code Supplied as FORMATIT, p.2.

```
 01  reprt-record.
     05                          pic xx.
     05 reprt-file-id            pic x.
     05                          pic xx.
     05 reprt-part-number        pic x(9).
     05                          pic x(2).
     05 reprt-description        pic x(30).
     05                          pic x(3).
     05 reprt-unit-price         pic z,zz9.999.
     05                          pic x(3).
     05 reprt-quantity-on-hand   pic zz,zz9.
     05                          pic x(5).
     05 reprt-quantity-on-ord    pic zz9.
     05                          pic x(5).
     05 reprt-reorder-level      pic zz9.
     05                          pic x.
     05 reprt-item-value         pic zzz,zzz,z99.99.
     05                          pic x(5).
     05 reprt-flag               pic x.
     05                          pic x.
*
*************************************************************
*
 01  totallers.
     05  page-no-flags    pic s9999       binary.
     05  page-value       pic s9(7)v99    packed-decimal.
```

Code Supplied as FORMATIT, p.3.

```
 procedure division using f-record, f-letter,
                          reprt-record, totallers.
 do-the-work.
     move f-letter               to reprt-file-id
     move f-part-number          to reprt-part-number
     move f-description          to reprt-description
     move f-quantity-on-hand     to reprt-quantity-on-hand
     move f-quantity-on-ord      to reprt-quantity-on-ord
     move f-unit-price           to reprt-unit-price
     move f-reorder-level        to reprt-reorder-level

*************************************************************
*
* Insert a call to 'valu' here, passing the unit price,
*     and quantity on hand fields, as well as the
*     'totallers' and 'reprt-record' structures
*
*************************************************************
     if f-quantity-on-hand + f-quantity-on-ord
          < f-reorder-level
        move '*' to reprt-flag
        add  1 to page-no-flags
     else
        move ' ' to reprt-flag
     end-if
     exit program.
```

Code Supplied as VALU

```
 Identification division.
 program-id.   valu.
*   Copyright (C) 2024 by Steven H. Comstock

 data division.
 working-storage section.
 01   wk-value        pic s9(9)v99 packed-decimal.

 linkage section.

*********************************************************
* Good place to define your four linkage section items:
*    the unit price, the quantity on hand, the
*    'totallers' structure and the 'reprt-record' structure
*********************************************************

 01

 01

 01   totallers.
      05   page-no-flags    pic s9999      binary.
      05   page-value       pic s9(7)v99   packed-decimal.

 01   reprt-record.
      05                              pic xx.
      05 reprt-file-id                pic x.
      05                              pic xx.
      05 reprt-part-number            pic x(9).
      05                              pic x(2).
      05 reprt-description            pic x(30).
      05                              pic x(3).
      05 reprt-unit-price             pic z,zz9.999.
      05                              pic x(3).
```

Code Supplied as VALU, p.2.

```
     05 reprt-quantity-on-hand    pic zz,zz9.
     05                           pic x(5).
     05 reprt-quantity-on-ord     pic zz9.
     05                           pic x(5).
     05 reprt-reorder-level       pic zz9.
     05                           pic x.
     05 reprt-item-value          pic zzz,zzz,z99.99.
     05                           pic x(5).
     05 reprt-flag                pic x.
     05                           pic x.

*********************************************************
* Good place to define your procedure division, specifying
*      the four parameters defined in the linkage section
*********************************************************
 procedure division using

*********************************************************
* Good place to calculate wk-value, add wk-value to
*      page-value, and move wk-value to reprt-item-value
*********************************************************
      exit program.
```

Copyright © 2024 by Steven H. Comstock

Expected Output

❒ **The generated report from this program should look like this:**

```
Inventory - Data List                                  Date mm/dd/yyyy      Page: 01

    Part       Product                                      Number    Number  Reorder    Inventory
 Id Number     Description                        Unit price On hand   On order Level        value   Flags
 A  PART00105  MCKINNEY MARVEL                        10.750      35      115      78       376.25
 X  PART03105  Final Flatulence                       10.750      35      115      78       376.25
 A  PART00108  FIBERGLASS PUNCHCARD  (20/BOX)        10.900      35      216     107       381.50
 X  PART03108  Giggling Gigolos                       10.900      35      216     107       381.50
 A  PART00111  FIBERGLASS CHALKBOARD                  11.050      35      111     113       386.75
 .
 .
 .
 A  PART00735  NEIGHBORHOOD QUARKS                    42.250     245      254     173      10,351.25
 X  PART03735  Founding Bearers                       42.250     245      254     173      10,351.25
 A  PART00738  SYMPATHETIC VANDALS                    42.400     245      166     201      10,388.00
 X  PART03738  Noisy Smells                          42.400     245      166     201      10,388.00

                                        Page totals:                          312,556.00     0

                                        Report totals:                      1,798,916.00    44
```