Today we’ll discuss . . .

1. Scope
2. Data
3. Structured programming
4. Subroutines
5. Built-in functions
6. Files and database
7. Miscellaneous enhancements
8. Final thoughts
Scope

Q: If you were designing a new computer language, would you include support for arithmetic?

A: It depends. First you have to tell me the purpose of the language.
Scope

• CL falls into the category of job control languages.
• (It is fair to say that CL is more than a JCL, since CL not only controls jobs, but all the operations on the system.)
• Job control languages do not need many of the capabilities of RPG and other high-level application languages.
• CL was introduced in 1980 with the first shipment of the System/38 and remained largely unchanged until 2002 (V5R3).
• Somebody with clout persuaded IBM that CL was not robust enough.

Data
New Data Types

- Signed integer (*INT)
- Unsigned integer (*UINT)
- Pointer (*PTR)

- First new data types since 1980.
- Signed integer type is more straightforward than the %BINARY (%BIN) function (but the %BIN function is not obsolete).
- Signed integer type is used by many APIs.

```
dcl &BytesAvail *int 4
dcl &BytesRtn   *int 4
Chgvar   &BytesAvail  %size(&RcvVar)
```
Defined-on variables

- Allow one variable to overlay another in memory
- Similar in function (but not in syntax) to data structures in RPG
- Syntax:
  
  `STG(DEFINED) DEFWVAR(defined-on-variable beginning-position)`
- Default beginning position is 1

Defined-on variable example

```
dcl &WorkDate  *char 7  stg(*defined) defvar(&WorkDate 1)
dcl &Century  *char 1  stg(*defined) defvar(&Century 1)
dcl &DateYMD  *char 6  stg(*defined) defvar(&DateYMD 1)
dcl &Year  *char 2  stg(*defined) defvar(&Year 1)
dcl &Month  *char 2  stg(*defined) defvar(&Month 1)
dcl &Day  *char 2  stg(*defined) defvar(&Day 1)
```
Pointers

- Store memory addresses
- The %ADDRESS (or %ADDR) function lets you set a pointer.
- The %OFFSET (or %OFS) function lets you do pointer arithmetic.

Structured programming
Looping – DOWHILE

- Top-tested loop
- Body of loop may not be executed at all
- Terminated by ENDDO

DOWHILE example

dcl  &Libs   *char  80
dcl  &Lib    *char  10

dowhile (&Libs *ne ' ')
   /* process the first library in the list */
   chgvar &Lib &Libs
   
   ... do something with &Lib

   /* remove the library that was just processed */
   chgvar &Libs %sst(&Libs 11 70)
endo
1. &LIBS is a list of library names, such as:

   MYLIB       YOURLIB     HISLIB     HERLIB     BIGLIBNAME

2. &LIB is one library name.

3. The first CHGVAR command copies the first 10 bytes of &LIBS to &LIB. (We don’t need %SST (substring) because CHGVAR truncates, but we could use %SST if we wanted to clarify our intentions.)

   MYLIB

4. After the system has done something with that library name, the second CHGVAR removes the first library name from the list. After the first iteration, &LIBS looks like this:

   YOURLIB       HISLIB     HERLIB     BIGLIBNAME

5. The loop continues until all libraries have been processed.

---

**Looping – DO UNTIL**

- Bottom-tested loop
- Body is always executed at least once
- Terminated by ENDDO
DOUNTIL example

duntil (&Status *eq '0')
    call qrcvdataq (&DtaQ &DtaQLib &DtaLen &Dta &Wait)
    select
        when (%sst(&Dta 11) *lt '1') then( chgvar &Status '0')
        when (%sst(&Dta 11) *eq '1') then( call Pgm1)
        when (%sst(&Dta 11) *eq '2') then( call Pgm2)
        when (%sst(&Dta 11) *eq '3') then( call Pgm3)
    endselect
enddo

Looping – DOFOR

• Top-tested loop
• Counted loop
• Terminating condition evaluated on each iteration (a “moving target”)
• Terminated by ENDDO
• Control variable must be an integer
• Loop increment can be positive or negative
DOFOR example

dcl &inOptions  *char 80  /* list of options */
dcl &Ndx       *uint 2

dcl &Option    *char 8

dcl &OptOffset *uint 2  /* offset into the list */
dcl &ToReport  *lgl

chgvar &OptOffset 3

dofor var(&Ndx) from(1) to(\%bin(&inOptions 1 2))
  chgvar &Option \%sst(&inOptions &OptOffset 8)
  select
    when (&Option *eq REPORT) then(chgvar &ToReport '1')
    when (&Option *eq NOREPORT) then(chgvar &ToReport '0')
  endselect
  chgvar &OptOffset (&OptOffset + 8)
endo

(See https://www.itjungle.com/2014/07/09/hg070914-story02/ for the complete example.)

LEAVE and ITERATE

- LEAVE jumps to the statement immediately following the end of the loop
- ITERATE jumps to the loop-control test
- Optional CMDLBL parameter allows you to specify which loop you wish to LEAVE or ITERATE
Looping – Middle-tested loops

• Implemented with DOWHILE

```
DOWHILE '1'
  . . . code that must run at least once
  IF COND(. . .) THEN(LEAVE)
  . . . code that might not run at all
ENDDO
```

• Must include a conditioned LEAVE to stop the loop

---

Middle-tested loop example

dclf Schedule opnid(s)

dowhile '1'
  rcvf opnid(s)
  monmsg cpf0864 exec(leave)
  if (&s_active *ne A) then(iterate)
  . . . do something to active record
  enddo
Case construct

- Implemented with SELECT/WHEN/OTHER/ENDSELECT keywords
- THEN may be null to ignore a case
- May be nested up to 25 levels deep

SELECT example (with keywords)

```sql
select
  when (&Option *eq 0) +
    then(chgvar &DevOption ('OUTQ ('cat &OptValue *tcat ')'))
  when (&Option *eq 0) +
    then(chgvar &DevOption ('DEV(' *cat &OptValue *tcat ')'))
  otherwise +
    cmd(chgvar &DevOption ('OUTQ(*DEV')])
endselect
```
SELECT example (without keywords)

```c
select
    when (&Option *eq 0) +
        chgvar &DevOption ('OUTQ(' *cat &OptValue *tcat ')')
    when (&Option *eq D) +
        chgvar &DevOption ('DEV(' *cat &OptValue *tcat ')')
    otherwise +
        chgvar &DevOption ('OUTQ(*DEV)')
endselect
```

Limitations

- Maximum of 25 levels of DOxxx commands
- Maximum of 25 levels of SELECT commands
Subroutines

- Follow the main routine and precede ENDPGM
- Bounded by SUBR and ENDSUBR commands
- Invoked by CALLSUBR command
- RTNSUBR and ENDSUBR leave a subroutine
- RTNSUBR and ENDSUBR may return a signed integer value
- Labels are local
- Recursive subroutine calls are permitted
- GOTO is not allowed into or out of subroutines
- Stack size is 99 levels, and may be changed to 20 thru 9999 in the SUBRSTACK parameter of the DCLPRCOPT command
- DMPCLPGM includes the subroutine stack
Subroutine with RTNVAL

dcl &SubrStatus *int 4
dcl &Reccount *dec 10

callsbr Prepare rtnval(&SubrStatus)
select
if (&SubrStatus *eq 2) do
  sndpmsg msg1d(usr1001) msgf(myusrmsgf) +
  msgdta('Prepare routine requires file POST40.') +
  msgtype('escape')

  subr Prepare /* Return code */
  /* 0 = all OK */
  /* 1 = completed with warnings */
  /* 2 = not completed due to a fatal error */

  chgvar &SubrStatus 0

  chkobj post40 *file aut(*objexist)
  monmsg cpf9801 exec(rtnsubr rtnval(2))

  rtvmbrd file(post40) nbcurrccd(&Reccount)
  if (&Reccount *eq 0) chgvar &SubrStatus '1'

  ... do something else
endsubr rtnval(&SubrStatus)

A more realistic example

Use subroutines to divide a program into logically-related pieces.

callsbr PrcParms /* process parameters */
callsbr CrtWrkF /* create the work files */
callsbr Updt /* update the database */
return

subr PrcParms
  ... code to process the parameters
endsubr

subr CrtWrkF
  ... code to create work files
endsubr

subr Updt
  ... code to update the database
endsubr
Built-in Functions

Trim Functions

- %TRIM, %TRIML, %TRIMR
- First parameter is the character value to trim
- Second parameter is the group of characters to trim
- If no second parameter, these functions trim blanks
%TRIM example

dcl &File *char 10
dcl &Lib *char 10
dcl &Stmf *char 128

chgvar var(&Stmf) +
  value('/QSYS.LIB/' *cat +
    %trim(&Lib) *cat '.LIB/' *cat +
    %trim(&File) *cat '.FILE/' *cat +
    %trim(&File) *cat '.MBR')

&Stmf has a value like this:

/QSYS.LIB/QTEMP.LIB/TEMP01.FILE/TEMP01.MBR

Case-conversion Functions

• %UPPER, %LOWER
• First parameter is the character value to convert
• Second parameter is the CCSID in which the data is stored. The default is zero, which means the job’s CCSID.
Case-conversion example

Allow the user to enter a value in uppercase, lowercase, or mixed case.

dcl &Response *char 8

chgvar var(&Response) value(%upper(&Reponse))
if (&Response *eq RETRY) do

Scan Functions

• %SCAN find the position of one string within another one
• %CHECK search a string for a character that is not part of a set
• %CHECKR like %CHECK but search backwards from the end

• You may specify *LDA as the string to be searched/checked.
Example

Does the list of options contain the value *REPORT?

dcl &Options *char 82
dcl &CrtRpt *lgl
chgvar &CrtRpt (%scan('*REPORT ' &Options) gt 0)

Type-conversion functions

- %CHAR
- %DEC
- %INT
- %UINT
Example

dcl &DeleteCt *dec 7
SndPgmMsg ('Deleted' *bcat %char(&DeleteCt) *bcat 'orders."

Sample output:
Deleted 10$ orders.

%CHAR discards leading zeros

---

Memory-allocation functions

- %SIZE returns the number of bytes occupied by a variable
- %LEN returns the number of digits (if numeric) or characters in a variable
Example

dcl &Cmd  *char  96
dcl &CmdLen  *dec  (15 5)
chgvar  &Cmd  ...whatever...
chgvar  &CmdLen  %size(&Cmd)
Call  qcmdexc  (&Cmd &CmdLen)

If the size of &Cmd changes, no mod is required here.

Files and database
File support - Multiple Files

- Up to five declared files in one CL procedure
- OPNID parameter assigns an identifier to each open file
- One file may have an OPNID of *NONE
- CL variables are prefixed with the OPNID value and an underscore
- OPNID is used for all file-related commands: DCLF, CLOSE, RCVF, SNDF, SNDRCVF, WAIT, ENDRCV

Example

```cl
  dclf QAFDMBRL opnid(MbrList)
  dspfd file(&srlib/srcfile) type(*mblist) +
    output(*outfile) outfile(qtemp/fdmbrl)
  ovrdbf qafmdbrl tofile(qtemp/fdmbrl)
  dowhile '1'
    rcvf opnid(MbrList)
    monmsg cpf0864 exec(leave)
      ... do something with one or more &MBLIST_ fields...
  enddo
```
File Support – CLOSE

- Allows you to close a file so it can be re-read
- A RCVF against a closed file opens the file

File Support – RUNSQL

- Allows you to execute a single SQL statement in a CL procedure
- Work on those concatenation skills!
dcl &FromDate *dec 7
dcl &ThruDate *dec 7
dcl &Comma *char 1 value(',,')
dcl &SQL *char 256

chgvar &SQL ('declare global temporary table SltData +
(FromDate dec(7), ThruDate dec(7)) +
with replace')
runsql sql(&Sql) commit(*none)

chgvar &SQL ('insert into session.SltData +
values (' *cat %char(&FromDate) *cat +
&Comma *cat %char(&FromDate) *cat '))'
runsql sql(&Sql) commit(*none)
INCLUDE

- Tells the compiler to copy in source code
- Like /COPY and /INCLUDE in RPG
- SRCMBR is the source member to copy
- SRCFILE is the file containing the copy member
- SRCFILE(*INCFILE) refers to the INCFILE parameter in CRTBNDCM, CRTCLPGM, CRTCLMOD

```
pgm

include srcmbr(ErrorCode) srcfile(qclsrc)
... more code

include srcmbr(ErrorCode) srcfile(qclsrc)

endpgm
```
Pass Parameters by Value

- Allowed on CALLPRC (specify *BYREF or *BYVAL)
- Useful for binding to C and MI routines
- Also works for calling RPG routines!

RPG module

ctl-opt nomain;
dcl-s goption packed(3);
dcl-proc SetOption export;
dcl-pi *n;
  inoption packed(3) value;
end-pi;
goption = inoption;
end-proc;

CL module

dcl &option *dec 3
callprc prc(SetOption) parm((&option *byval))
Increased maximums

- Maximum number of parameters on PGM command increased from 40 to 255
- Maximum number of parameters on CALL and TFRCTL commands increased to 255
- Maximum number of PARM commands in command source increased from 75 to 99
- Maximum size of *CHAR variables increased from 9999 to 32,767

Final thoughts
Final thoughts

• CL is probably more or less what it will be when it comes to end of life, so you may as well master what’s there.
• We could already do what these many of these enhancements do by calling HLL programs.
• Don’t try to replace RPG.

ENDPGM