RPG

Dopo la MOVE c'è ancora vita?

Marco Riva

www.markonetools.it

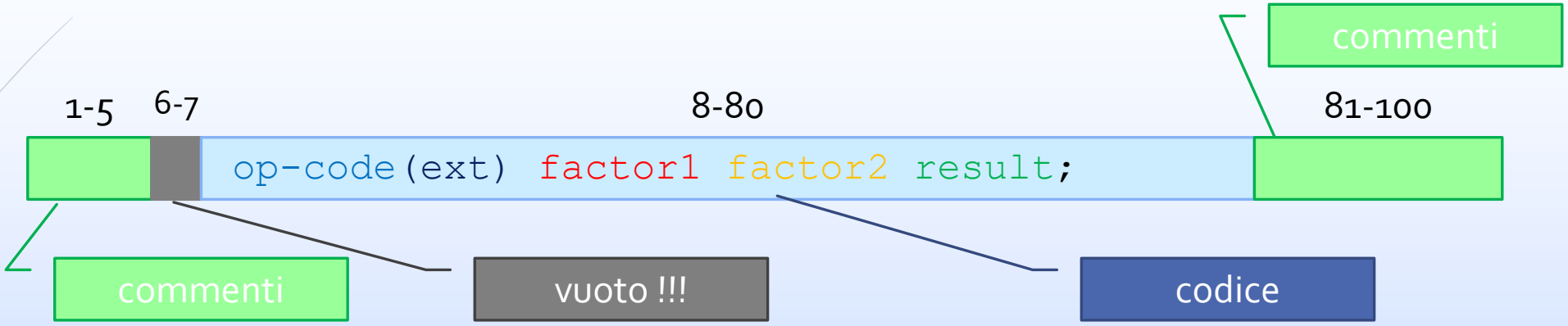
Ultimo aggiornamento: 15/12/2020

La fine della MOVE

nel formato libero non esistono i codici operativi MOVE, MOVEL, MOVEA!

- come faremo?
- ci sarà ancora vita per i programmatori?
- dovremo rimpiangere i bei tempi quando c'era la MOVE?

Regole base



- ogni istruzione deve terminare con il carattere **;**
- `/free` … `/end-free` **non** sono più necessari da 7.1 TR7

Codici operativi "obsoleti"

- ATTENZIONE: non tutti i codici operativi possono essere scritti in formato libero: ADD, ANDxx, CABxx, CALL, CASxx, CAT, COMP, DEFINE, DIV, DOUxx, DOWxx, GOTO, IFxx, **MOVE**, **MOVEA**, **MOVEL**, MULT, MVR, ORxx, PARM, PLIST, SCAN, SETOFF, SETON, SQRT, SUB, SUBST, TAG, TESTB, TESTN, TIME, WHENxx, Z-ADD, Z-SUB
- I codici operativi non gestiti dal formato libero possono essere definiti "obsoleti" e "deprecabili"

vediamo come sostituire il codice operativo più usato: MOVE

eval

- codice operativo che consente di assegnare ad una variabile il risultato di un'espressione
- in formato libero **può essere omesso** a meno di dover specificare un operatore di estensione
- eval **sostituisce l'intero contenuto** della variabile di destinazione

https://www.ibm.com/support/knowledgecenter/ssw_ibm_i_74/rzasd/zzeval.htm#zzeval

esempi

```
// variabile in un'altra variabile
eval MioCampo = wsCampo;
```

```
// sostituzione porzione di variabile
eval %subst(MioCampo:1:2) = 'AA';
```

```
// assegnazione di un'espressione
eval MioCampo = 'select * from ' + %trim(NomeTabella);
```

```
dcl-s Omaggio ind;
// assegnazione di un'espressione booleana
eval Omaggio = (ImportoRiga = *zeros);
```

MOVE: muovere a destra

- in alternativa si può usare **`evalr`**
- attenzione che **sovrascrive interamente il contenuto** della variabile risultato
- si può usare `evalr` con funzione `%subst`

```
evalr %subst(TO10:6) = From5;
```

- `evalr` non può essere utilizzato per modificare campi a lunghezza variabile (`varchar`)

MOVE: spostare per convertire

- con MOVE e MOVEL si poteva «liberamente» spostare il contenuto di una variabile numerica in alfanumerica o viceversa
- di fatto si trattava di una **conversione di tipo dati** e quindi come tale va trattata nel formato libero
- conversione da numero positivo a alfanumerico

```
MyNbr       = 12345.67;
To10        = *all'x';
      //to10 contains '   1234567'
Evalr to10 = %editc(mynbr:'X');
      //to10 contains '0001234567'
Evalr to10 = '0000000000' + %trim(%editc(myNbr:'X'));
      //to10 contains '0012345,67'
Evalr to10 = '0000000000' + %trim(%editc(myNbr:'3'));
```

MOVE: spostare per convertire

- conversione da numero positivo a alfanumerico con carattere di riempimento :

```
MyNbr1  = 1234567.89;
        // '$1234567,8'
to10 = %editc(myNbr1:'3':'$');
        // '01234567,8'
to10 = %editc(myNbr1:'3':'0');
MyNbr= 123.89;
        // '**123,89'
to10 = %editc(myNbr:'3':*astfill);
```

MOVE: spostare per convertire

- «sottostringare» un campo numerico

```
MyNbr = 12345.67;
  // a destra: MyNbr2V0 = 67. !!! Non funziona con i negativi
MyNbr2V0 = %int(%subst(%editc(MyNbr:'X'):6:2));
MyNbr = -12345.67;
  // a destra:MyNbr2V0 = -67
MyNbr2v0 = %rem(%int(MyNbr*%int(10**%decPos(MyNbr))):100);
  // a sinistra: MyNbr2V0 = -12
MyNbr2V0 = %int(%subst(%editc(MyNbr:'L'):1:2));//12
EvalR Sign1 = %editc(MyNbr:'L');              // '-'
If Sign1='-';
  MyNbr2V0 *= -1;
Endif;
```

MOVEA e array

- non esiste equivalente di `MOVEA` in formato libero
- si definisce una DS che ha come sottocampo l'array
- quindi invece di usare l'array con MOVEA si utilizza la DS con EVAL

```
dcl-ds dsArray;
    Array char(2) dim(10);
end-ds;
dcl-s tmp char(20);

c                   movea     Array          tmp
   // EQUIVALE A
    tmp = dsArray;
```

- oppure si usa la funzione **`%subarr`**

```
%subarr(Array:3:2) = 'AaBb';
```

Composizione di un codice numerico

- Talvolta vengono utilizzati campi numerici per definire codici nelle tabelle
- P.es. il codice cliente può essere un campo numerico 10 interi e 0 decimali
- Sarebbe più corretto definire i codici sempre come campi alfanumerici, ma la storia insegna che possono esistere codici numerici che sono da interpretare come composizione di più sottocampi
- Abitualmente si gestiscono con una MOVEL (porzione a sinistra) e una MOVE (porzione a destra)

Composizione di un codice numerico

```
d CodCliente                      10s 0
d Serie                            2s 0
d Codice                           8s 0

c                   z-add     10              Serie
c                   z-add     456             Codice
c                   movel     Serie           CodCliente
c                   move      Codice          CodCliente
 * CodCliente = 1000000456
```

```
dcl-ds MiaDS;
  CodCliente zoned(10);
  Serie zoned(2) overlay(CodCliente:1);
  Codice zoned(8) overlay(CodCliente:*next);
end-ds;
dcl-s Msg char(52);

Serie = 10;
Codice = 456;
  // CodCliente = 1000000456
```

Composizione di un codice numerico

```
dcl-s CodCli zoned(10);
dcl-s Serie zoned(2);
dcl-s Codice zoned(8);
dcl-s Coefficiente like(Codice:+1);
dcl-s Msg char(52);

Coefficiente = 10**%size(Codice);              // 100000000

Serie = 10;
Codice = 456;
CodCli = Serie*Coefficiente + Codice;          // 1000000456

CodCli = 1000000456;
Serie = %div(CodCli:Coefficiente);             // 10
Codice = CodCli  - (Serie * Coefficiente);     // 456
```

Riferimenti

- E-mail aziendale: mriva@sirio-is.it
- Blog: www.markonetools.it
- Facebook: https://www.facebook.com/markonetools/
- YouTube: https://www.youtube.com/channel/UCb47YJQJCzU-5x4nnGzDu-w
- E-mail blog: info@markonetools.it
- Linkedin: www.linkedin.com/in/marcoriva-mk1