Moderní RPG

Vladimír Župka, 2024

Vývoj tvaru jazyka RPG

Formulářový zápis programu

1959	RPG	děrné štítky
1969	RPG II	děrné štítky
1975	RPG II	diskety
1978	RPG III	terminály, disky

Výrazy v některých rubrikách formuláře C

1994	RPG IV	výrazy v příkazech (EVAL, IF, DOW, DOU)

Volný zápis ve formuláři C

2001	ILE RPG	úseky výpočtů ve volném tvaru (/FREE, /END-FREE) - verze 5.1
2010	ILE RPG	smíšené výpočty (ve volném nebo formulářovém tvaru) - verze 7.1

Alternativa k formulářům H, F, D, P - smíšený zápis

2013	ILE RPG	(CTL-OPT, DCL-F, DCL-xx, END-xx) - verze 7.2

Zcela volný zápis programu

2016	ILE RPG	úvodní příkaz **FREE - verze 7.3

Program ve smíšeném zápisu (od verze 7.2)

```
1....678.1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8
      // komentář volného tvaru až od sloupce 7
       * komentářový řádek

     FCENIK     IF   E           K DISK

     D  pocet          C                   const(10)
           DCL-DS struktura;   // příkaz až od sl. 8 nebo dále
              znaky char(10);
              cislo int(10);
           END-DS struktura;

        // popisy souborů a dat lze zapsat v libovolném pořadí
        DCL-F STAVYP  // příkaz zapsaný ve více řádcích
              keyed
              usage(*update);

        if *in03;
     C                   GOTO      KONEC
        endif;
        // ... výpočty
     C     KONEC         TAG
```

Program ve zcela volném zápisu (od verze 7.3)

```
1...                                                                    ...
**FREE
CTL-OPT dftactgrp(*no);

DCL-F CENIK keyed;

    DCL-C pocet const(10);

DCL-F TISK printer(*ext)  // komentář může být na každém řádku příkazu
           usage(*output)     // komentář 2
           oflind(*in55);     // komentář 3

   // příkazy mohou začínat kdekoliv v řádku.

     /INCLUDE QRPGLESRC,PROTOTYPY

    DCL-S soucet packed(9: 2) inz(0); // jednoduchá proměnná
```

Nahrazení formulářů (od verze 7.2)

Formulář		***Volný zápis***	
6	24		
H		CTL-OPT	Řídicí volby pro kompilátor
F		DCL-F	Popis souboru
D	C	DCL-C	Pojmenovaná konstanta
	S	DCL-S	Samostatná proměnná
	DS	DCL-DS	Datová struktura
		DCL-SUBF	Podpole (subfield)
		END-DS	Konec datové struktury
	PI	DCL-**PI**	Rozhraní procedury (procedure interface)
		DCL-PARM	Parametr
		END-PI	Konec rozhraní procedury
	PR	DCL-**PR**	Prototyp procedury
		DCL-PARM	Parametr
		END-PR	Konec prototypu
C		příkaz ;	Výpočty
P	B	DCL-PROC	Začátek procedury
	E	END-PROC	Konec procedury
I		/INCLUDE nebo /COPY	
O		/INCLUDE nebo /COPY	

Použití souboru

Formulář		*Volný zápis*			
17	20				
I		DCL-F	filename	DISK	USAGE(*INPUT)
I	A			**DISK**	USAGE(*INPUT: *OUTPUT)
U				**DISK**	USAGE(*INPUT: *UPDATE: *DELETE)
U	A			**DISK**	USAGE(*INPUT: *UPDATE: *DELETE: *OUTPUT)
O				**DISK**	USAGE(*OUTPUT)
O	A			**DISK**	USAGE(*OUTPUT)
O		DCL-F	filename	**PRINTER**	USAGE(*OUTPUT)
C		DCL-F	filename	**WORKSTN**	USAGE(*INPUT: *OUTPUT)

Typy dat

Formulář		*Volný zápis*	*Hodnoty*
40	44		
A		CHAR (n)	1 - 16773104
A	VARYING	VARCHAR (n {:m})	1 - 16773102 : 2|4
C		UCS2 (n)	1 - 8386552 (počet dvoubajtů)
C	VARYING	VARUCS2 (n {:m})	1 - 8386550 : 2|4
P		PACKED (n {:m})	1 - 63 : 0 - 63
S		ZONED (n {:m})	1 - 63 : 0 - 63
I		INT (n)	3, 5, 10, 20
U		UNS (n)	3, 5, 10, 20
B		BINDEC (n {:m})	1 - 9 : 0 - 9
F		FLOAT (n)	4, 8
N		IND	'0' = *ON, '1' = *OFF
D		DATE {(*ISO)}	*DMY-, ...
T		TIME {(*ISO)}	*HMS., ...
Z		TIMESTAMP {(n)}	0 - 12 des. místa u sekund
*		POINTER	

Externí jméno souboru

Soubor `CENIKP`

```
     A            R CENIKPF0
     A              MATER          5
     A              CENA          10  2
     A              NAZEV         50
     A            K MATER
```

Program `EXTF`

```
dcl-PI *N;  // rozhraní volaného programu EXTF
   filename char(21);   // vstupní parametr (plné jméno souboru)
end-PI;
dcl-F ceny                          // jméno pro program
      EXTFILE(filename)             // jméno pro výpočet (aktuální soubor)
      EXTDESC('*LIBL/CENIKP');      // jméno pro kompilaci (vzorový soubor)
read ceny;
...

*inlr = *on;  // uzavře soubor a vyčistí proměnné
```

Volání z RPG

```
dcl-PR EXTF EXTPGM;  // prototyp
   *N char(21);  // vstupní parametr (plné jméno souboru)
end-PR;

dcl-S qualifName char(21) inz('VZRPG_FREE/CENIKP');

CALLP EXTF(qualifName);  // CALLP nepovinné

return;  // povinný příkaz kvůli RPG cyklu
```

Volání z CL

```
CALL   PGM(EXTF) PARM('VZRPG_FREE/CENIKP')
```

nebo

```
OVRDBF FILE(CENIKP) TOFILE(VZRPG_FREE/CENIKP)  /* OVRDBF je silnější */
CALL   PGM(EXTF) PARM('VZRPGIV/CENIKP')
```

Program bez RPG cyklu ("lineární program")

```
ctl-OPT MAIN (LINPGM)    // jmenuje hlavní proceduru modulu

dcl-F  CENIKP;           // globální soubor generuje popisy I a O

dcl-PROC LINPGM;         // hlavní procedura (program) vypadá jako podprocedura
    dow not %eof;
        read  CENIKP;
        dsply MATER;
    enddo;
end-PROC;
```

Volání z RPG s prototypem bez parametrů

```
dcl-PR LINPGM EXTPGM end-PR;  // povinný prototyp bez parametrů a návr. hodnoty
CALLP LINPGM();       // volání programu (CALLP nepovinné, závorky povinné)
```

Volání z CL

```
CALL PGM(LINPGM)
```

Datové struktury

Nepojmenovaná podpole

```
DCL-DS status;    // API QDCLCFGD (List Configuration Descriptions)
   *n   char(10) inz('*EQ');
   *n   char(10) inz('*ACTIVE');
END-DS;
```

Datová struktura bez podpolí

```
dcl-DS dstr LEN(100) END-DS;    // END-DS je součástí příkazu
```

Program status data structure

```
dcl-DS pgmStat PSDS qualified;
   status *STATUS;
   routine *ROUTINE;
   library CHAR(10) POS(81);           // pozice čítaná od 1
   internal_job_id CHAR(16) POS(380); // od verze 7.4
   system_name CHAR(6) POS(396);       // od verze 7.4
end-DS;
```

Přektytí podpole daného jménem a pozicí

```
dcl-ds *n;           // nepojmenovaná struktura
   field1 CHAR(10) inz('1234567890');
   array1 CHAR(1) dim(5) OVERLAY(field1);
   array2 CHAR(1) dim(5) OVERLAY(field1: 6); // pozice v podpoli
end-ds;

dcl-ds DATUM;
   RRRRMMDD zoned(8);
   RRRR zoned(4) POS(1);                 // pozice od začátku struktury
   MM   zoned(2) OVERLAY(RRRR: *NEXT); // následující pozice
   DD   zoned(2) OVERLAY(MM: *NEXT);
end-ds;
```

Poznámka: V OVERLAY nesmí být jméno struktury (DATUM). Místo toho použijeme POS(1).

Překrytí oblasti od podpole daného jménem (od verze 7.4)

```
dcl-ds struct;
   string  CHAR(100);
   P1      CHAR(5);
   P2      CHAR(5);
   P3      CHAR(5);
   array   CHAR(5)  dim(3)  SAMEPOS(P1); // array překrývá P1, P2, P3
end-ds;
```

Externě popsaná datová struktura

```
DCL-DS CENIKP  EXT  END-DS;  // v programu nesmí být DCL-F CENIKP

DCL-DS STA_DS  EXTNAME('STAVYP': *ALL) END-DS;  // END-DS je součástí příkazu

DCL-DS CENY_DS LIKEREC(CENIKPF0); // bez END-DS, automaticky qualified
DCL-F  CENIKP; // pro LIKEREC musí být definován soubor!
```

```
// Změna a doplnění podpolí kvalifikované externí struktury
DCL-DS CENY_DS EXTNAME('CENIKP': *INPUT) qualified;
    material EXTFLD('MATER');       // přejmenování podpole
    CENA     EXTFLD  INZ(2.00);     // změna hodnoty (EXTFLD bez parametru)
    druh_mater char(2) OVERLAY (material);  // první dva znaky materiálu
    dodatek  char(10) inz('ABC');  // nové podpole až po existujících polích
END-DS;

// reference kvalifikovaným jménem podpole
CENY_DS.dodatek = 'DEF';  // kvalifikace jménem struktury a tečkou
CENY_DS . druh_mater '04';  // přepíšu první dva znaky materiálu
```

Poznámka: Konstantní jména objektů v parametrech je nutné psát s apostrofy!

Vektor datové struktury, třídění, hledání

```
dcl-f CENIKP;
dcl-s  idx   int(5);
dcl-c  pocet const(5);
dcl-ds STR LIKEREC(CENIKPF0) DIM(pocet) inz;

dow not %eof and idx < pocet;  // naplní položky struktury
   idx += 1;
   read CENIKP STR(idx);  // čte do položky struktury STR
enddo;

SORTA(D) STR(*).CENA;     // setřídí STR sestupně podle ceny zboží

idx = %LOOKUP(10.99: STR(*).CENA);  // zjistí index podle zadané ceny
if idx > 0;
   dsply STR(idx).NAZEV;  // zobrazí odpovídající název zboží
endif;

SORTA STR %fields(CENA: MATER);  // setřídí STR vzestupně podle ceny a materiálu

dump(a);
```

Maximum a minimum ve vektoru datových struktur

```
dcl-f CENIKP;

dcl-s  idx   int(5);
dcl-ds str LIKEREC(CENIKPF0) DIM(4) inz;  // vektor dat. struktur

dow not %eof and idx < %elem(str);  // naplním položky struktury
   idx += 1;
   read CENIKP str(idx);  // čtu do idx-té položky struktury
enddo;

idx = %MAXARR(str(*).CENA) ;   // 3      index
dsply %char( str(idx).CENA );  // 10.99 maximum

dsply %char( str(%MINARR(str(*).CENA)).CENA );  // 1.23 minimum
dump(a);
```

Obsah souboru

```
MATER            CENA   NAZEV
00003             3.33   To je materiál 00009
00001             1.23   Materiál pro kompletaci automobilu Škoda Yeti
00002            10.99   cvičky od Bati
00004             4.44   Zboží 00004
```

Vnořená struktura, šablona, ukazatel

```
dcl-ds info_t TEMPLATE qualified; // šablona pro kompilaci
   jmeno  varchar(25);
   dcl-ds addresa DIM(2);  // vektor vnořené datové struktury
      ulice  varchar(25);
      mesto  varchar(25);
   end-ds;
end-ds;
// proměnná podle šablony bázovaná ukazatelem
dcl-ds informace LIKEDS(info_t) DIM(100) BASED(basPtr);
// ukazatel na datovou strukturu
dcl-s basPtr POINTER;

// získám paměť ze systému a její adresu do ukazatele
basPtr = %ALLOC(%size(informace: *all));

informace(2).jmeno = 'Štěpán Řihošek';
informace(2)  .addresa(1).mesto = 'Praha 9';
informace(2)  .addresa(1).ulice = 'Křižíkova 45';
informace(2)  .addresa(2).mesto = 'Beroun';
informace(2)  .addresa(2).ulice = 'Plzeňská 81';
```

Maximum a minimum ve vektoru

```
dcl-s  idx   int(5);
dcl-s  arr int(10) dim(5);

arr = %LIST (43: 78: 12: 6: 59);  // naplnění vektoru

idx =  %MAXARR (arr);      // 2   index
dsply arr(idx);            // 78  maximální hodnota

dsply arr(%MINARR(arr));  // 6   minimální hodnota

dump(a);
```

Proměnlivá dimenze vektoru

```
dcl-s AUTO_ARR char(3) DIM(*AUTO: 1000);   // 0 prvků

auto_arr(100) = 'abc';          // 100 prvků
auto_arr(*NEXT) = 'abc';        // 101 prvků
%ELEM(auto_arr: *ALLOC) = 120; // 101 prvků, přiděleno 120 dostupných prvků
%ELEM(auto_arr: *KEEP) = 120;  // 120 prvků (zachováno 101 hodnot, ost. nedef.)


dcl-s VAR_ARR  char(1) INZ('*') DIM(*VAR: 1000);   // 0 prvků

%ELEM(var_arr) = 3;             //  |*|*|*|     3 prvky
var_arr(1) = 'a';               //  |a|*|*|
var_arr(2) = 'b';               //  |a|b|*|
%ELEM(var_arr) = 1;             //  |a|         1 prvek
%ELEM(var_arr: *ALLOC) = 4;     //  |a|         1 prvek
%ELEM(var_arr: *KEEP) = 4;      //  |a|b|*|_|   3 zachované hodnoty, 4. nedef.
var_arr(4) = 'd';               //  |a|b|*|d|   4. hodnota doplněná
```

Kódování dat v programu

CTL-OPT **CCSID** (parametr { : *ccsid* }) – předvolba CCSID pro modul

***CHAR :** **JOBRUN* | **UTF8* | **HEX* | *0 – 65535*
***UCS2 :** **UTF16* | *1200* | *13488*
***EXACT** správné zacházení s literály, proměnnými a buffery souborů

CTL-OPT **CCSIDCVT** (parametr1 { : parametr2 }) – akce při konverzi dat

***EXCP** zpráva 452 při chybě v konverzi mezi různými CCSID
***LIST** kompilátor vypíše seznam implicitních a explicitních konverzí

/SET **CCSID** (*CHAR | *UCS2 : *ccsid*) – dočasné nastavení nové předvolené hodnoty

/RESTORE **CCSID** (*CHAR | *UCS2) – vrácení dosavadní předvolené hodnoty

Poznámka: V příkladech znamená *JOBRUN kódování CCSID(870) jako ve zdrojovém souboru.

DCL-S | DCL-DS **CCSID** (*ccsid*) – nastavení u proměnné

0 – 65535	platná číselná hodnota CCSID
*DFT	použije se platná předvolená hodnota CCSID pro modul. Užitečný u LIKE.
*HEX | 65535	data nemají CCSID a nelze je použít u CCSID konverzí
*JOBRUN	CCSID převzaté z úlohy (job) při výpočtu
*UTF8	CCSID 1208 (Unicode UTF-8)
*UTF16	platí jen pro data typu UCS-2 a CCSID je 1200 (Unicode UTF-16)
*EXACT	externí podpole DS má stejné CCSID jako externí pole souboru (v DDS)
*NOEXACT	externí podpole DS má platnou předvolenou hodnotu CCSID

Převod textu na malá a velká písmena – UTF-8

```
ctl-opt  ccsid(*char: *utf8);

dcl-s sr varchar(10) inz('ď');  // hledací argument jako proměnná!

dcl-s text  varchar(30) inz('Přijeď do žďáru.');
dcl-s text_low varchar(30);
dcl-s text_up  varchar(30);

dsply %CHARCOUNT(text);   // 16 znaků
dsply %len(text);         // 21 bajtů

text_low = %LOWER(text);  // přijeď do žďáru.
text_up  = %UPPER(text);  // PŘIJEĎ DO ŽĎÁRU.

dsply %SCAN(sr: %LOWER(text): *NATURAL);  // 6  (šestý znak)
dsply %SCAN(sr: %LOWER(text));            // 7  (sedmý bajt)

dump(a);
```

Spojení prvků vektoru do řetězce s oddělovačem – UTF-8

Kódování znaků v UTF-8 v celém modulu podle zadání CCSID v CTL-OPT

```
CTL-OPT CCSID(*CHAR: *UTF8);  // předvolená hodnota CCSID v modulu

dcl-s text_array varchar(20) dim(4);
dcl-s text       varchar(100);

text_array = %LIST('ďas': 'ťal': 'doň': 'řach');  // vytvoření vektoru slov

text = %CONCATARR(', ' : text_array);  // spojení prvků do řetězce s oddělovačem

dump(a);
```

```
NAME                ATTRIBUTES           VALUE
TEXT                CHAR(100) VARYING(2) CCSID(1208)
                                         'ďas, ťal, doň, řach'
                    VALUE IN HEX         '0017C48F61732C20C5A5616C2C20646FC588...
                    ...
```

Rozčlenění textu do vektoru podle oddělovače – UTF-8

Čítání znaků kódovaných v UTF-8 podle zadání CCSID u definice dat

```
dcl-s  text_array  varchar(20) ccsid(*UTF8) dim(4);
dcl-s  text   varchar(100) ccsid(*UTF8);
dcl-s  sep    varchar(10) ccsid(*UTF8) inz(', ');  // oddělovač mezera a čárka
dcl-s  prvek  char(20);

text = %CONCAT(sep: 'ďas': 'ťal': 'doň': 'řach');  // spojení slov s oddělovačem

text_array = %SPLIT(text: sep);  // rozčlenění textu do vektoru slov

FOR-EACH prvek IN text_array;
   dsply prvek;
endfor;
dump(a);
```

```
NAME                 ATTRIBUTES           VALUE
TEXT_ARRAY           CHAR(20) VARYING(2)  CCSID(1208) DIM(4)
                       (1)                'ďas'
                       VALUE IN HEX       '0004C48F617300000000000000000000000000000000'X
                       (2)                'ťal'
                       VALUE IN HEX       '0004C5A5616C00000000000000000000000000000000'X
                       ...
```

Kódování UCS-2

```
dcl-s ucs2_AB VARUCS2(5) inz(U'00410042');  // literál AB v UCS-2
dcl-s ucs2_dia VARUCS2(5) inz('žš');        // U'017E01610'

dcl-s cat_char varchar(10);
dcl-s cat_ucs2 VARUCS2(10);

cat_char = ucs2_AB + 'ŽŠ' + ucs2_dia;  // konverze do *CHAR(*JOBRUN)
cat_ucs2 = ucs2_AB + 'ŽŠ' + ucs2_dia;  // konverze do *UCS2(13488)

dump(a);
```

```
NAME                ATTRIBUTES            VALUE
CAT_CHAR            CHAR(10) VARYING(2)   'ABŽŠžš'
                    VALUE IN HEX          '0006C1C2B8BCB69C00000000'X
CAT_UCS2            UCS2(10) VARYING(2)   CCSID(13488)
                                          'ABŽŠžš'
                    VALUE IN HEX          '0006004100420017D0160017E016100000000000000000'X
```

Kódování dat v souboru a programu

CTL-OPT **OPENOPT** (parametr) – předvolba pro konverzi dat souborů z / do *JOBRUN

***CVTDATA** - předvolba pro DATA(*CVT) u souborů (probíhá konverze)
***NOCVTDATA** - předvolba pro DATA(*NOCVT) u souborů (neprobíhá konverze)

DCL-F **DATA** (parametr) – u definice souboru: konverze dat z / do *JOBRUN

***CVT** - probíhá konverze
***NOCVT** - neprobíhá konverze

Databázový soubor CCSID_F

```
     A          R CCSID_FR
     A            CS_37          10A         CCSID(37)
     A            CS_870         10A         CCSID(870)
     A            CS_UTF8        10A         CCSID(1208)
     A            CS_UTF16       10G         CCSID(1200)
```

Program – zápis z literálů do polí různých CCSID v databázi

```
// předvolené kódování modulu je *JOBRUN
dcl-F  CCSID_F  disk usage(*OUTPUT);

CS_37 = 'ž š č';
CS_870 = 'ž š č';
CS_UTF8 = 'ž š č';
CS_UTF16 = 'ž š č';

write  CCSID_FR;  // proběhne konverze z *JOBRUN do jednotlivých CCSID
```

Data souboru – výpis DSPPFM

```
CS_37     CS_870    CS_UTF8   CS_UTF16
*...+....1....+....2....+....3....+....4....+....5
Ţ đ č     ž š č     E´ E~ Dý   =   /
B484444444B494444444CB2CA2C822070206020002020202 02
30C07000006 0C07000005E05104D001E0011001D0000000000
```

Program – čtení dat do datové struktury s konverzí do UTF-8

```
CTL-OPT  ccsid (*char: *utf8)    // předvolené kódování modulu je UTF-8
         ccsidcvt (*list) ;      // výpis konverzí v protokolu o překladu

dcl-F  CCSID_F data(*cvt);       // s konverzí

dcl-DS STRUCT likerec(CCSID_FR);

read  CCSID_F  STRUCT;  // čte přes *JOBRUN do UTF-8 (ale UTF-16 je vždy bez konverze)

dump(a);
```

```
                    C C S I D   C o n v e r s i o n s
          From CCSID        To CCSID          References
          870               1208                  15
          37                1208                  15
          *JOBRUN           1208                  15      15      15
```

```
STRUCT                DS
  CS_UTF16            UCS2(10) CCSID(1200) 'ž š č     '
                      VALUE IN HEX         '017E00200161002001 0D00200020002000200020'X
  CS_UTF8             CHAR(10) CCSID(1208) 'ž š č  '        'C5BE20C5A120C48D2020'X
  CS_37               CHAR(10) CCSID(1208) 'ž š č  '        'C5BE20C5A120C48D2020'X
  CS_870              CHAR(10) CCSID(1208) 'ž š č  '        'C5BE20C5A120C48D2020'X
```

Čtení do datové struktury přesně podle souboru

```
ctl-OPT  ccsid (*char: *utf8)   // předvolené kódování modulu je UTF-8
         ccsidcvt (*list) ;     // výpis konverzí v protokolu o překladu

dcl-F  CCSID_F;  // s konverzí

dcl-DS STRUCT likerec(CCSID_FR) CCSID(*EXACT);

read  CCSID_F  STRUCT;  // čte přes *JOBRUN do stejných CCSID

dump(a);
```

```
                    C C S I D   C o n v e r s i o n s
         From CCSID        To CCSID          References
         *JOBRUN           1208                  15
         *JOBRUN           870                   15
         *JOBRUN           37                    15


STRUCT                DS
  CS_UTF16            UCS2(10) CCSID(1200) 'ž š č     '
                      VALUE IN HEX         '017E00200161002001 0D00200020002000200020'X
  CS_UTF8             CHAR(10) CCSID(1208) 'ž š č  '        'C5BE20C5A120C48D2020'X
  CS_37               CHAR(10) CCSID(37)   '          '     'B3408C40474040404040'X
  CS_870              CHAR(10) CCSID(870)  'ž š č     '     'B6409C40474040404040'X
```

Čtení do datové struktury bez konverze

```
CTL-OPT  ccsid (*char: *utf8)   // předvolené kódování modulu je UTF-8
         ccsidcvt (*list) ;     // výpis konverzí v protokolu o překladu

dcl-F  CCSID_F data(*nocvt);  // bez konverze

dcl-DS STRUCT likerec(CCSID_FR);

read  CCSID_F  STRUCT;  // čte rovnou do UTF-8

dump(a);
```

```
                    C C S I D   C o n v e r s i o n s
         From CCSID        To CCSID          References
         870               1208                  15
         37                1208                  15


STRUCT                DS
  CS_UTF16            UCS2(10) CCSID(1200) 'ž š č     '
                      VALUE IN HEX         '017E00200161002001 0D00200020002000200020'X
  CS_UTF8             CHAR(10) CCSID(1208) 'ž š č  '        'C5BE20C5A120C48D2020'X
  CS_37               CHAR(10) CCSID(1208) '       '        'C2B720C3B020C3A52020'X 슷 Hangul
  CS_870              CHAR(10) CCSID(1208) 'ž š č  '        'C5BE20C5A120C48D2020'X
```

Převod některých příkazů do volného formátu

Příkaz KLIST

Pevný formát

```
FSTAVYP    UF   E             KDISK
C                    MOVEL     '01'           ZAVOD
C                    MOVEL     '01'           SKLAD
C                    MOVEL     '00001'        MATER

C      KLIC          CHAIN     STAVYP

C      KLIC          KLIST
C                    KFLD                     ZAVOD
C                    KFLD                     SKLAD
C                    KFLD                     MATER
```

Volný formát

```
dcl-f  STAVYP keyed usage(*update);

dcl-ds klicStavu LIKEREC(STAVYPF0: *KEY);

klicStavu.ZAVOD = '01';
klicStavu.SKLAD = '01';
klicStavu.MATER = '00001';

CHAIN   %KDS(klicStavu : 3) STAVYP;
```

```
ZAVOD = '01';
SKLAD = '01';
MATER = '00001';

CHAIN   (ZAVOD: SKLAD: MATER) STAVYP;
```

CALL, PLIST

Pevný formát

```
c                   CALL      'PGM1'
c                   PARM      'XXX'          A                 10
c                   PARM      'YYY'          B                 10
```

```
c                   CALL      'PGM1'         PARAMS

c     PARAMS        PLIST
c                   PARM      'XXX'          A                 10
c                   PARM      'YYY'          B                 10
```

Volný formát

```
DCL-PR PGM1 EXTPGM; // prototyp volání
    *N char(10);
    *N char(10);
END-PR;

DCL-S A char(10) inz('XXX');  // definice skutečných parametrů
DCL-S B char(10) inz('YYY');

CALLP PGM1 ( A : B );  // volání programu
```

*ENTRY PLIST

Pevný formát

```
c      *ENTRY         PLIST
c                     PARM                    A             10
c                     PARM                    B             10

c      A              dsply
c      B              dsply
c                     return
```

Volný formát

```
DCL-PI *N ;          // program interface
    A char(10);
    B char(10);
END-PI;

dsply A;
dsply B;
return;
```