Ing. Karel M E T Ž L

ZVT - OKR Ostrava

Generátor zdrojových programů

1. Úvod.

Se zaváděním nových a stále výkonějších počítačů narůstají nároky na programovací kapacity. Situaci nelze řešit dalším zvyšováním počtu programátorů, ale důslednou racionalizací programátorské práce. Jednou z forem racionalizace je právě vytváření nových programů generováním jejich zdrojové formy.

Myšlenka generování vychází ze skutečnosti, že v řadě zdrojových programů se různé sekvence výroků opakují a nabízí se možnost je z jednoho programu do druhého beze změny okopírovat. Typickým příkladem této skupiny výroků jsou rozpisy dat, ať již pro jazyk Cobol nebo PL/I. Další oblastí, ve které může generátor zdrojových programů racionalizovat i skvalitňovat práci programátora, je generování nových výroků ve zdrojové formě příslušného jazyka pomocí vyvolávání různých funkcí s parametry nebo i bez nich. Sem patří na příklad generátory normalizovaného programování pro ANS Cobol nebo PL/I.

V další části svého příspěvku se budu zabývat generátorem zdrojových programů vyvinutým v Závodě výpočetní techniky OKR v Ostravě [1]. Generátor byl vypracován pro počítač IBM 370 pracující pod systémem OS/VS1, ale je použitelný i na počíta-

čích JSEP vyšších řad pracujících pod systémem OS (na př. na počítačích EC 1040 fy Robotron). Je budován modulárně a každý uživatel si generuje verzi podle svých vlastních požadavků. Podle používaného jazyka si vybírá požadované funkce ev. si své vlastní doprogramuje, volí si jazyk pro diagnostické zprávy a určuje default hodnoty některých operandů.

2. Popis funkce generátoru.

Generátor vytváří zdrojové programy a ukládá je jako členy do knihovny s členěnou organizací. Jednotlivé výroky generovaného programu pocházejí z následujících zdrojů:

a. běžné výroky zdrojového jazyka kódované programátorem,
b. sekvence výroků vyjmuté z libovolných členů zdrojové knihovny,
c. sekvence výroků generované některými funkcemi na př. generátory normalizovaného programování pro ANS Cobol nebo PL/I.

Zdroje jsou vzájemně nezávislé, lze je libovolně kombinovat a kterýkoliv z nich může i chybět. Vzhledem k tomu, že generátor připouští i vyjímání sekvencí výroků ze staré verze stejnojmenného programu, lze pomocí něj provádět i opravy programů na zdrojové knihovně.

Výstupní výroky jsou automaticky číslovány. Poloha a délka pořadového čísla je volitelná, stejně tak jako počáteční hodnota a přírůstek číslování. Při jednom chodu je možno vygenerovat i několik zdrojových programů.

V programu je zabudována dostatečná chybová diagnostika se samovysvětlujícími zprávami. Dojde-li k jakékoliv chybě ve vstupních datech, právě generovaný člen se neuloží a po skončení chodu je vydán návratový kód (Condition Code) různý od nuly. Návratový kód může být v dalších krocích testován, a tak se lze vyhnout zbytečnému kompilování neopravené verze programu.

3. Řídící výroky.

Řídící výroky jsou formálně podobné řídícím výrokům opravného programu IEBUPDTE fy IBM [2] nebo fy Robotron [3] t. zn., že se řídící výrok skládá z operace a operandů zapsaných ve volném formátu od sloupců 4 až do 71, přičemž ve sloupcích 1 a 2 musí být uloženo ./. Každý štítek, který nemá ve sloupcích 1 a 2 ./ je považován za výrok zdrojového jazyka a ukládá se přímo do výstupního souboru.

3.1. Řídící výrok ADD nebo REPL.

Jeden z výše uvedených řídících výroků musí být vždy uveden před generováním každého členu. Udává zda jde o přidání nového členu na knihovnu nebo výměnu verze. V operandu je pak jméno ukládaného programu případně následované požadavky na speciální číslování nebo na současný výpis generovaného členu.

Na př.:

```
          ./    ADD   NAME=ZC062
```

nebo

```
          ./    REPL NAME=ZROO,LIST=ALL,NUM=10,SEQID=016
```

3.2. Řídící výrok INSERT.

Výrok udává, z kterého členu a která rozmezí výroků se mají vybrat a uložit do generovaného členu. Rozmezí se zapisuje ve tvaru číslo1-číslo2 a udává, že se mají vybrat výroky, jejichž čísla vyhovují podmínce

$$\text{číslo1} \leq \text{číslo.výroku} \leq \text{číslo2}$$

Je-li výrok INSERT ukončen za posledním výrokem rozmezí čárkou, očekává se, že bude vybírání z uvedeného členu pokračovat. Pokračovací štítek pro vybírání ze stále stejného členu nemá uvedenou žádnou operaci a za povinnými ./ v 1. a 2. sloupci následují rovnou další požadované rozmezí. Počet

pokračovacích štítků není omezen a navíc je možno mezi ně vkládat i štítky s vlastním kódováním nebo řídící štítky s operací FUNC.

Na př.:

```
./  INSERT  NAME=UP27,0-284,286-314,
              vlastní kódování
              (libovolný počet štítků)
./    827-853,
./     FUNC  NAME=GNC,P12
./      854-925,927-1025,1027-1230
```

Při zápisu operace INSERT s uvedením několika rozmezí, je třeba dbát na to, aby rozmezí postupovala vzestupně. Je-li nutné vybírat rozmezí, která naleží v pořadí nebo části opakovat, musí se znovu udat kompletní operace INSERT.

3.3. Řídící výrok FUNC.

Výrokem FUNC se vyvolává některá ze zabudovaných funkcí. Je-li třeba, předávají se jí formou znakového řetězce parametry. Uživatel si může vybrat některou z funkcí dodávaných s generátorem nebo si napsat i vlastní funkce a začlenit je do své verze generátoru.

Vyvolaná funkce pak na základě zadaných parametrů, nebo i bez nich, vygeneruje do výstupního členu řadu zdrojových výroků příslušného programovacího jazyka.

Na př. výrokem:

```
./    FUNC    NAME=GNC,W
```

se vyvolává generátor normalizovaného programování pro ANS Cobol a parametr W udává, že má generovat výroky rezervací pro klíčová pole a výhybky do WORKING-STORAGE SECTION.

4. Zabudované funkce.

Výběr zabudovaných funkcí je závislý na používaném zdrojovém jazyku nebo jazycích. Uživatel si při vytváření své verze generátoru volí, které funkce chce zařadit. Jednoduchým způsobem lze začlenit i speciální funkce napsané ve středisku a rozšířit tak rejstřík funkcí.

4.1. Funkce vhodné pro použití v jazyku ANS Cobol.

IDC — Generuje začátek IDENTIFICATION DIVISION až po hlavičku paragrafu REMARKS včetně. Potřebné údaje se dosadí z parametru, nebo jsou uloženy již při vytváření verze generátoru (údaj do paragrafu INSTALLATION).

EDC — Generuje začátek ENVIRONMENT DIVISION až po hlav. paragrafu FILE-CONTROL včetně.

DDC — Generuje začátek DATA DIVISION až po hlavičku FILE SECTION včetně.

PODC — Používá se k podtrhávání nebo optickému členění programu. Generuje sled hvězdiček počínaje sloupcem 7 až po pozici ve které se v předcházejícím příkaze vyskytla první tečka. Nenarazí-li nikde na tečku, skončí vkládání v 72. sloupci.

GNC — Generátor normalizovaného programování pro ANS Cobol. Podrobnější popis je uveden dále.

4.2. Funkce vhodné pro použití v jazyku PL/I.

PODP — Používá se k podtrhávání, tvorbě rámečků nebo optickému členění programu. Generuje sled hvězdiček (s případnými lomítky) s ohledem na náležitosti komentářů v jazyku PL/I.

GNP Generátor normalizovaného programování pro jazyk PL/I. Podrobnější popis je uveden dále.

4.3. Funkce napsané uživatelem.

Uživatel si může do generátoru začlenit libovolné vlastní moduly napsané v jazyku assembler. Vazba s řídícím modulem se děje přes společnou oblast (common area) a zápis vytvořených řádků, případný zápis na soubor diagnostických zpráv se provádí jednoduchými makroinstrukcemi. V Závodě výpočetní techniky OKR je na př. zabudována funkce pro generování standardních záhlaví tiskových sestav pro použití ve zdrojovém jazyku ANS Cobol.

5. Generátory normalizovaného programování.

Použití generátorů předpokládá u programátora znalost základních principů normalizovaného programování, tak jak byly popsány v [4] nebo [5]. V jednom programu, vyžaduje-li to jeho logika, může pracovat až 5 generátorů současně. Každý z nich může obhospodařovat až 9 vstupních souborů setříděných až podle 9 klíčů.

K tomu, aby mohl generátor pracovat, musí si vybudovat vnitřní tabulky obsahující údaje o zpracovávaných souborech, větách, třídících klíčích a jejich prioritách. Budování vnitřních tabulek se děje pomocí funkce typu P. Funkční štítky s typem P se vkládají bezprostředně za zdrojové štítky (nebo záznamy vtažené operací INSERT) se jmény souborů, klíčových položek atd.

Jakmile jsou vybudovány vnitřní tabulky, lze vyvoláním funkce typu W zkontrolovat jejich úplnost a logickou správnost a generovat výroky pro rezervování paměti pro řídící oblasti a výhybky. V generátoru GNP (pro jazyk PL/I) se současně generují BEGIN bloky pro ON ENDFILE všech vstupních souborů. Úspěšný průchod typem funkce W je podmínkou pro generování ostatních bloků.

Generování ostatních bloků normalizovaného programování se děje vyvoláváním funkce GNC (pro ANS Cobol) nebo GNP (pro jazyk PL/I) s uvedením typu A až I.

Při uvedení typu A se generuje úvodní blok. Ve verzi pro Cobol se zde otevírají vstupní soubory. Ostatní náplň závisí na programu a programátor ji musí sám ve zdrojovém jazyku zakódovat. Patří sem na př. zpracování parametrů, nastavení počátečních hodnot, mulování atd.

Při vyvolání generátoru s uvedením typu B se generuje blok vstupů. Zahrnuje podmíněné čtení všech souborů, přesuny klíčových položek z přečtené věty do řídící oblasti v paměti a nastavení stavového bytu v řídící oblasti. Ve verzi pro Cobol se zde řeší i situace AT END. Generátor pro PL/I si podle svých vnitřních tabulek volí pro jednotlivé vstupní soubory buď čtení v modu LOCATE nebo MOVE.

Má-li generátor více než jeden vstupní soubor, lze vyvoláním funkce s typem C vygenerovat blok výběru věty ke zpracování. Vygenerovaný kód vyhledává nejmenší z řídících oblastí, přenáší ji do řídící oblasti nových klíčů a muluje v ní stavový byte.

Blok D testuje změny na jednotlivých úrovních třídících klíčů a podle nich vyvolává příslušné subrutiny z bloku G. Podle způsobu vyvolání generátoru se generuje buď plný blok D generující vyvolávání součtových subrutin počínaje nejnižší úrovní až po změnu s dalším vyvolávením hlavičkových subrutin od změny až po nejnižší úroveň nebo zkrácený blok D, který podle volby generuje vyvolávání pouze buď součtových nebo hlavičkových subrutin.

Vyvolání generátoru s typem E generuje podmíněné vyvolávání subrutin bloku H, nastavení výhybky prvního průchodu Rg-E a skok zpět na začátek bloku B.

Pro ostatní bloky normalizovaného programování vytváří

generátor pouze úvodní rámečky s textem názvu bloku. V blocích C a H pak musí programátor použít standardní návěští na která se předává řízení z bloků D a E.

V příloze je uveden příklad jednoduchého programu v jazyku PL/I při jehož vytváření bylo použito generátoru GFP. S ohledem na formát sborníku musely být výroky zprava upraveny, což by však nemělo program příliš zkreslit. Výroky označené svislou čarou byly generovány generátorem. V místech označených šipkou byly ve vstupních štítcích štítky s operací FUNC.

6. Závěr.

Generátor zdrojových programů je významným prostředkem modulárního programování na úrovni zdrojového jazyka. Operace INSERT umožňující vybírání již dříve zakódovaných sekvencí z jiných programů a FUNC generující nové sekvence zdrojových výroků racionalizují práci programátora. Použití generátorů normalizovaných programů přispívá k zavádění uvedené metody a je automatickým strážcem všech jejích základních principů.

Literatura.

[1] Informace č. 93: Generátor zdrojových programů ZGENSRC Informace OPTV Závodu výpočetní techniky OKR

[2] GC35-0005 OS/VS Utilities fy IBM

[3] Dienstprogramme fy Robotron

[4] Normované programování. Mechanizace a automatizace administrativy, 11, 1971, č.6, s.180-184, č,7-8, s.215-220

[5] METZL, K: Metoda normalizovaného programování ve zpracování hromadných dat. Sborník Metody programování počítačů III. generace. Havířov 1975.

```
ZKUSCHP: PROC OPTIONS(MAIN);
/*****************************************************************/
*                                                                 *
*       ZKUSCHP.                                                  *
*       NOVAK,ZVT OKR - OSTRAVA JE     3.12.1975                  *
*                                                                 *
*       UKAZKA POUZITI GENERATORU NORMALIZOVANEHO PROGRAMOVANI.   *
*       AKTUALIZACE KMENOVEHO SOUBORU JEDNIM ZMENOVYM SOUBOREM.   *
*       SOUBORY JSOU SETRIDENY PODLE  1. PODNIKU,                 *
*                                     2. ZAVODU,                  *
*                                     3. CISLA ZNAMKY.            *
*       ZE ZMENOVEHO SOUBORU SE VYBIRAJI POUZE VETY S TYPEM 323.  *
*       K JEDNOMU KMENOVEMU UDAJI MUZE EXISTOVAT 1 NEKOLIK ZMEN.  *
*       NENAJDE-LI SE K NEKTERE ZMENOVE VETE ODPOVIDAJICI KMENOV  *
*       VETA, TISKNE SE ZPRAVA O CHYBE.                           *
*       VLASTNI AKTUALIZACE SPOCIVA V OPRAVE HODNOTY=323 O HODNO  *
*       ZMEN.                                                     *
*       O PROVEDENYCH ZMENACH SE TISKNE PREHLEDNA SESTAVA S MEZI:  *
*       - ZA ZAVOD A PODNIK.                                      *
*       PRI ZMENE PODNIKU SE NASTAVUJE NOVA STRANKA.              *
*                                                                 *
*****************************************************************/

DECLARE /*SOUBORY*/
P1S         IN1 FILE RECORD INPUT,          /*ZMENY*/
P2S         IN2 FILE RECORD INPUT,          /*KMEN1*/
            OUT FILE RECORD OUTPUT,         /*KMEN2*/
            TISK FILE RECORD OUTPUT ENVIRONMENT(PRINT);

P1Q DECLARE 1 E1 BASED(P_E1),               /*ZMENOVA VETA*/
            2 FILLER1 CHAR(16),
            2 TYP     CHAR( 3),
P12         2 ZAV     CHAR( 2),
P11         2 POD     CHAR( 2),
P1S         2 CIS_ZN  CHAR( 6),
            2 ZMENA DECIMAL FIXED(7),
            ;
P2Q DECLARE 1 F1 BASED(P_F1),               /*VSTUPNI KMENOVA VETA*/
            2 FILLER1 CHAR(36),
P21         2 POD     CHAR( 2),
P22         2 ZAV     CHAR( 2),
P2S         2 CIS_ZN  CHAR( 6),
            2 FILLER2 CHAR(20),
            2 JMENO   CHAR(18),
            2 ADRESA  CHAR(24),
            2 FILLER3 CHAR(30),
            2 HODNOTA_323 DECIMAL FIXED(7),
            2 FILLER4 CHAR(18),
            2 DAT_NAR,
              5 DD CHAR(2),
              5 MM CHAR(2),
              5 RR CHAR(2),
          1 F1_DEF BASED(P_F1) CHAR(160),
            ;
```

```
DECLARE 1 G1 STATIC CHAR(160),          /*VYSTUPNI KMENOVA VETA*/
          ;

DECLARE /*TISKOVE RADKY*/
        1 TZ_RADEK BASED(P_TISK) CHAR(121),
        1 T1_RADEK BASED(P_TISK),
          2 CTLASA CHAR( 1),
          2 MEZAV  CHAR(11),
          2 TLIST  CHAR( 4),
          2 LIST   PIC'ZZZZ9',
        1 T1RADEK BASED(P_T1SK),
          2 FILLER1 CHAR(39),
          2 T1_NADPIS CHAR(46),
        1 T2_RADEK BASED(P_T2SK),
          2 CTLASA CHAR( 1),
          2 INDIK_CHYBY CHAR(10),
          2 FILLERX1    CHAR( 2),
          2 POD         CHAR( 2),
          2 FILLERX2    CHAR( 2),
          2 ZAV         CHAR( 2),
          2 FILLERX3    CHAR( 2),
          2 CIS_ZN      CHAR( 6),
          2 FILLERX4    CHAR( 2),
          2 JMENO       CHAR(20),
          2 ADRESA      CHAR(24),
          2 HODNOTA_323 PIC'---8---8---9',
          2 T2_NOVA_HOD PIC'---8---8---9',
          2 T2_ROZDIL   PIC'---8---8---9',
          ;

DECLARE
        POCRAD      STATIC DEC FIXED (3) INIT(100),
        MAXRAD      STATIC DEC FIXED (3) INIT(68),
        POCLIST     STATIC DEC FIXED (5) INIT(0),
        ROZDIL_CZ   STATIC DEC FIXED (7) INIT(0),
        ROZDIL_ZAV  STATIC DEC FIXED (7),
        ROZDIL_POD  STATIC DEC FIXED (7),
          ;

/*****************************************************
*                                                    *
*     KLICE A VYHYBKY GENERATORU I                   *
*                                                    *
*  IN1                    IN2                        *
*  E1                     F1                         *
*  P_E1                   P_F1                       *
*  POD                    2 POD          2           *
*  ZAV                    2 ZAV          2           *
*  CIS_ZN                 6 CIS_ZN       6           *
*                                                    *
*****************************************************
```

```
DCL 1 WIN_K0 STATIC,
      3 WIN_S  CHAR(1) INIT('0'),
      3 WIN_P1 CHAR(2),
      3 WIN_P2 CHAR(2),
      3 WIN_P3 CHAR(6),
      3 WIN_C  CHAR(1) INIT('1'),
    WIN_K  CHAR(12)  DEF WIN_K0,
    WIN_K1 CHAR(1)   DEF WIN_K0,
    WIN_K2 CHAR(5)   DEF WIN_K0,
    WIN_K3 CHAR(3)   DEF WIN_K0,
        ;

DCL 1 W1S_K0 STATIC,
      3 W1S_S  CHAR(1) INIT('9'),
      3 W1S_P1 CHAR(2),
      3 W1S_P2 CHAR(2),
      3 W1S_P3 CHAR(6),
      3 W1S_C  CHAR(1) INIT('1'),
    W1S_K  CHAR(12)  DEF W1S_K0,
    W1S_K1 CHAR(1)   DEF W1S_K0,
    W1S_K2 CHAR(5)   DEF W1S_K0,
    W1S_K3 CHAR(3)   DEF W1S_K0,
        ;

DCL 1 R1_SW0 STATIC,
      3 (R1_D1,R1_D2,R1_D3) CHAR(1),
      R1_SW DEF R1_SW0 CHAR(3),
    R1_E STATIC CHAR(1) INIT('0');

    ON ENDFILE (IN1)                    BEGIN;
                                          W11_S = '2';
                                          CLOSE FILE (IN1);
                                          GOTO B11_900;
                                          END;

DCL 1 W11_K0 STATIC,
      3 W11_S  CHAR(1) INIT('0'),
      3 W11_P1 CHAR(2),
      3 W11_P2 CHAR(2),
      3 W11_P3 CHAR(6),
      3 W11_C  CHAR(1) INIT('1'),
    W11_K  CHAR(12)  DEF W11_K0,
        ;

    ON ENDFILE (IN2)                    BEGIN;
                                          W12_S = '2';
                                          CLOSE FILE (IN2);
                                          GOTO B12_900;
                                          END;

DCL 1 W12_K0 STATIC,
      3 W12_S  CHAR(1) INIT('0'),
      3 W12_P1 CHAR(2),
      3 W12_P2 CHAR(2),
      3 W12_P3 CHAR(6),
```

```
        3 W12_C  CHAR(1) INIT('2'),
        W12_K  CHAR(12)  DEF W12_KO,
          ;
A

      /*****************************************************************
      *
      *     A1  U V O D N I   B L O K
      *
      *****************************************************************
        CALL TTISK;                          /* UVODNI LOCATE A NEZEROV
BF

      /*****************************************************************
      *
      *     B1  B L O K   V S T U P U
      *
B1B   *****************************************************************
B11_001:
      IF W11_S = '0' THEN DO;
         READ FILE (IN1          ) SET  (P_E1          );
B1E      IF E1.TYP ^= '323' THEN GOTO B11_001;
         W11_P1 = E1.POD;
         W11_P2 = E1.ZAV;
         W11_P3 = E1.CIS_ZN;
         W11_S = '1';
         END;
B2    B11_900:

B12_001:
      IF W12_S = '0' THEN DO;
         READ FILE (IN2          ) SET  (P_F1          );
         W12_P1 = F1.POD;
         W12_P2 = F1.ZAV;
         W12_P3 = F1.CIS_ZN;
         W12_S = '1';
         END;
C     B12_900:

      /*****************************************************************
      *
      *     C1  V Y B E R   V E T Y   K E   Z P R A C O V A N I
      *
      *****************************************************************

C10_001: ;
         IF W11_K < W12_K THEN DO;
            WIN_K = W11_K;
            W11_S = '0';
            END;
         ELSE DO ;
```

```
                    WIN_K = W12_K;
                    W12_S = '0';
                    END;
DF

          /******************************************************
          *
          *    D1  T E S T O V A N I   Z M E N   K L I C U
          *
DS        *******************************************************

D10_001:
          R1_SW = '000';
          IF WIN_K1 ¬= W1S_K1 THEN DO;
             R1_D1 = '1';
             CALL G13_001;
             IF WIN_K2 ¬= W1S_K2 THEN DO;
                R1_D2 = '1';
                CALL G12_001;
                IF WIN_K3 ¬= W1S_K3 THEN DO;
                   R1_D3 = '1';
                   CALL G11_001;
                   IF WIN_S = '2' THEN GOTO F10_001;
                   END;
                END;
             END;
          W1S_K = WIN_K;
E

          /******************************************************
          *
          *    E1  Z P R A C O V A N I   V E T Y
          *
          *******************************************************

E10_001: ;
          IF WIN_C = '1' THEN CALL H11_001;
          IF WIN_C = '2' THEN CALL H12_001;
          R1_E = '1';
          GOTO B11_001;
F

          /******************************************************
          *
          *    F1  Z A V E R E C N Y   B L O K
          *
          *******************************************************
F10_001:
             RETURN;
```

```
G
      /*****************************************************************
      *                                                               *
      *     G1  S U B R U T I N Y  Z M E N  K L I C U                 *
      *                                                               *
      *****************************************************************/

G11_001: PROC; /*** ZPRACOVANI SOUCTU ZA PODNIK ***/
      IF K1_C <> '0' THEN DO;
         T2_RADEK.JMENO = 'CELKEM ZA PODNIK';
         T2_RADEK.T2_ROZDIL = ROZDIL_POD;
         T2_RADEK.CTLASA = '0';
         CALL TISK;
         END;
      POCRED = 100;
      ROZDIL_POD = 0;
      END;

G12_001: PROC; /*** ZPRACOVANI SOUCTU ZA ZAVOD ***/
      IF K1_C <> '0' THEN DO;
         T2_RADEK.JMENO = 'CELKEM ZA ZAVOD';
         T2_RADEK.T2_ROZDIL = ROZDIL_ZAV;
         T2_RADEK.CTLASA = '0';
         CALL TISK;
         CALL TISK;
         ROZDIL_CO = ROZDIL_POD + ROZDIL_ZAV;
         END;
      ROZDIL_ZAV = 0;
      END;

G13_001: PROC; /*** ZPRACOVANI SOUCTU ZA CISLO ZNAMKY ***/
      IF K1_C <> '0' THEN DO;
         ROZDIL_ZAV = ROZDIL_ZAV + ROZDIL_CI;
         ROZDIL_CI = 0;
         END;
      END;
H
      /*****************************************************************
      *                                                               *
      *     H1  S U B R U T I N Y  Z P R A C O V A N I  V E T         *
      *                                                               *
      *****************************************************************/

H11_001: PROC; /*** ZPRACOVANI ZMENOVE VETY ***/
      IF (W11_P1=W12_P1 & W11_P2=W12_P2 & W11_P3=W12_P3) THEN
         /*ROVNOST KLICU Tj. NALEZENA K NI KMENOVA VETA*/
         ROZDIL_CI = ROZDIL_CI + EL.ZMENA;
      ELSE /*NENALEZENA PRISLUSNA KMENOVA VETA*/
         DO;
```

```
            IF POCRAD > MAXRAD THEN CALL IHLAVICKA;
            T2_RADEK.INDIK_CHYBY = '**********';
            T2_RADEK.POC = W11_P1;
            T2_RADEK.ZAV = W11_P2;
            T2_RADEK.CIS_ZN = W11_P3;
            T2_RADEK.JMENO = 'NENALEZEN KM.UCAJ';
            CALL ITISK;
            END;
      END;

H12_001: PROC; /***   ZPRACOVANI KMENOVE VETY   ***/
      IF ROZDIL_CZ ¬= 0 THEN DO;
            IF POCRAD > MAXRAD THEN CALL IHLAVICKA;
            T2_RADEK = F1; BY NAME;
            F1.HODNOTA_323 = F1.HODNOTA_323 + ROZDIL_CZ;
            T2_NOVA_HOD = F1.HODNOTA_323;
            T2_ROZDIL = ROZDIL_CZ;
            CALL ITISK;
            END;
      G1 = F1_DEF;
      WRITE FILE(OUT) FROM(G1);
      END;

   /*****************************************************************
   *
   *    I I   P O M O C N E   S U B R U T I N Y
   *
   *****************************************************************/

IHLAVICKA: PROC;
      POCRAD = 0;
      POCLIST = POCLIST + 1;
      T1_RADEK.CTLASA = '1';
      CALL ITISK;
      T1_NADPIS='Z M E N Y  P R I P L A T K U  T Y P  3 2 3'
      T1_RADEK.TLIST = 'LIST';
            T1_RADEK.LIST = POCLIST;
      T1_RADEK.CTLASA = '0';
      CALL ITISK;
      T1_RADEK.HLAV =
            '                    POC ZAV CIS.ZN. JMENO                A
                  PUV.HODN.  NOVA HODN.    ROZDIL';
      CALL ITISK;
      CALL ITISK;
      END;

ITISK: PROC; /***   LOCATE A VYPEZEROVANI TISKOVEHO RADKU   ***/
      LOCATE TZ_RADEK FILE(TISK) SET(P_TISK);
      TZ_RADEK = ' ';
      END;
END;
```