Ing. Pavel Juřina
Ing. Oldřich Helman
NHKG Ostrava

FORMÁTOVÁNÍ VÝSTUPU (TISKU) KOMPILÁTORU PROGRAMOVACÍHO JAZYKA PL/I

1. Úvod

V NHKG Ostrava - Kunčice se začal v roce 1973 současně s instalací počítače IBM 370/145 používat programovací jazyk PL/1. Kompilátor tohoto programovacího jazyka provádí tisk zdrojového programu ve stejné formě, v jaké ho napsal programátor, což bylo pro nás zhoršení situace oproti původnímu programovacímu jazyku CLEO počítače LEO 360, jehož kompilátor prováděl formální úpravy zdrojového programu.

S rostoucím počtem vyprodukovaných programů v jazyce PL/1, s nutností předávat je mezi jednotlivými programátory a zejména s rostoucím počtem oprav klesala jejich čitelnost, přehlednost a přenositelnost mezi jednotlivými programátory. Proto byla v NHKG vypracována interní norma, týkající se formální stránky zápisu zdrojového programu. Bylo například stanoveno, že instrukce se budou psát od desátého sloupce, že instrukce v DO skupinách a BEGIN blocích budou odsazeny o dva sloupce doprava, že příkaz END bude "zarovnán" s IHEN a podobně.

Příklad:

```
A = C * D;
IF A> 10
THEN DO;
  X = Y + 20;
  C = C + 1;
END;
```

Tyto zásady je možno stanovit pro všechny případy zápisu programovacího jazyka PL 1 a pak po programátorech důsledně vyžadovat jejich dodržování.
V praxi je však realizace těchto opatření velice neekonomická a velmi výrazně snižuje produktivitu práce programátorů i jejich vedoucích, kteří by měli dodržování těchto zásad kontrolovat. Tyto zásady je totiž možné dodržet snad při prvotním psaní programu. Dodržet tyto zásady při provádění oprav do programů je však velmi nesnadné. Při provádění oprav do programů je nutno provést následující činnosti:

a) Nejprve je nutno odměřit na kompilačním výtisku, do kterého sloupce je nutno příkaz napsat.

b) Je nutno příkaz napsat do odpovídajícího sloupce na programovacím formuláři.

Toto je možné pouze v případech, že se opravují jednotlivé příkazy. Při vsunování bloků typu DO nebo BEGIN je však situace poněkud jiná.

Příklad:

```
IF A > 10
THEN DO;
  C = 12;
  C1 = C2 + C3;
  IF X > Y
  THEN DO;
    Z = 14;
    Z1 = Z2 + Z3;
    Z4 = Z5 * Z6;
     .
     .
    WRITE FILE (OUT) FROM (ZAKAZ);
  END;
  ELSE DO;
    X = Y;
     .
     .
    WRITE FILE (OUT) FROM (PZAKAZ);
  END;
END;
```

Ve stávajícím programu je třeba provést opravu tak, že skupinu instrukcí, označených jako ** budeme provádět pouze v případě, že hodnota proměnné A1 bude větší jak hodnota proměnné A2.

Opravu je možno provést takto:

```
                IF A> 10
                THEN DO;
                  C = 12;
                  C1 = C2 + C3;
                  IF A1> A2          /* OPRAVA */
                  THEN DO;           /* OPRAVA */
                  IF X > Y
                  THEN DO;
                    Z = 14;
                    Z1 = Z2 + Z3;
                    Z4 = Z5 * Z6;
                        .
                        .
                        .
                        .
                    WRITE FILE (OUT) FROM (ZAKAZ);
**              END;
                  ELSE DO;
                    X = Y;
                       .
                       .
                       .
                       .
                    WRITE FILE (OUT) FROM (PZAKAZ);
                  END;
                  END;               /* OPRAVA */
                END;
```

Je nepravděpodobné, že by programátor opravil (posunul o dva sloupce doprava) i celou množinu instrukcí, označených xx.
Stejné problémy nastávají i při vypouštění bloků.

Aby tyto problémy byly odstraněny, byl vypracován programový modul, který provádí formátování tisku kompilace bez ohledu na formální zápis zdrojového programu. Je tím dosaženo to, že programátor se nemusí starat o formální strukturu programu, ani při jeho vytváření, ani při provádění oprav. Při kompilaci s použitím tohoto modulu se takto upraví i již dříve vytvořené programy. Vyvolání formátovacího modulu nevyžaduje žádné další specifikace od programátora při zadání kompilace. Formátovací modul je napsán v jazyce ASSEMBLER a je velký asi 2,5 KB.

2. POPIS FUNKCÍ MODULU

Modul provádí úpravy výstupu kompilátoru ve 3 částech.

V 1. části se upravuje tisk před zdrojovým programem. V této části se vypouštějí hlavičky a prázdné řádky.

Ve 2. části se provádějí úpravy zdrojového programu s využitím 132 pozic na řádku. Víceřádkové hlavičky jsou nahraženy jednořádkovou hlavičkou a využívá se 70 řádků na stránce.

Ve 3. části se vytiskne pouze první hlavička, ostatní se vypouštějí, stejně tak se vypouštějí i prázdné řádky.

Těmito opatřeními se nesníží přehlednost kompilačního tisku, dojde přitom k úspoře papíru cca o 30 %.

3. ÚPRAVY PROVÁDĚNÉ VE ZDROJOVÉM PROGRAMU

Modul pracuje systémem úpravy řádku programu a neprovádí přesuny z jednoho řádku programu do druhého řádku. Rovněž neprovádí syntaktickou analýzu příkazů jazyka PL/1, ale využívá již kompilátorem zjištěných

skutečností, zejména hodnot "LEVEL" a "NESTED". Proto je nutno tyto volby zahrnout do voleb kompilátoru při provádění kompilace, jinak budou všechny příkazy začínat ve sloupci 10.

3.1 ÚPRAVA DEKLARACÍ

Posouvají se údaje začínající DCL nebo DECLARE do 10. sloupce (zdrojového zápisu), ostatní štítky patřící k DCL bez čísla úrovně, do 14. sloupce. Pokud štítek začíná číslem úrovně, posouvá se o číslo úrovně a 2 sloupce doprava.
Pokud je na štítku za deklarací poznámka, posouvá se vpravo a ukládá se:

a) zleva od tiskové pozice 87 v případě, že je poznámka kratší než 45 znaků
b) zprava, když je poznámka delší jak 45 znaků, takže končí ve sloupci 132.

Tyto poznámky musí být na štítku ukončeny, jinak modul neidentifikuje.

Příklad:

```
DCL    1   A   STATIC,        /* STRUKTURA */
           2 B CHAR(3),        /* VAHA */
           2 C FIXED(5);       /* POCET */
```

Poznámky uvnitř DCL na samostatných štítcích však nemusí být ukončeny na témže štítku.

Příklad:

```
DCL    1 STR  STATIC,          /* STRUKTURA */
         2 S1 CHAR(1),         /* ZNAK ZAVEDENI */
         2 S2 CHAR(3),         /* HODNOTA */
                               /* DALSI POPIS
                                * DEKLARACI
                                * SOUBORU      */
         2 S3,
           3 S4(4),
             4 S5 CHAR(1),
             4 S6 FIXED(3);
```

C pozuámek na samostatných štítcích se provádí rozlišení mezi tzv. "dělicími" poznámkami, které oddělují části programu a poznámkami, které jsou pokračováním předcházející poznámky u instrukce.
Pokud tyto poznámky začínají před sloupcem 21 (programovacího formuláře), pak jsou zarovnány vlevo od úrovně (sloupce) předcházející instrukce.
Začínají-li za sloupcem 20, upravují se stejně jako v kap. 3.1 body a), b).

3.2 ÚPRAVA NÁVĚŠTÍ

Návěští jsou posunuta od sloupce 2 (zdrojového zápisu) pokud za ním následuje příkaz a návěští je delší a tím jeho konec zasahuje do dané pozice příkazu, je příkaz odsunut na první volný sloupec.

Příklad:

```
LABEL:   C=0;
         IF X>Y
         THEN DO;
           A=B;
NAVESTI_ZMENY:X=Y;
           A1=B1;
         END;
```

3.3 ÚPRAVA PŘÍKAZŮ

Všeobecně: provádí se určení pozice počátku příkazu dle algoritmu: daná pozice = 2 * (LEVEL + NESTED - 1) + 10.
To znamená, že příkazy na LEVEL = 1 a NESTED = 0 jsou umístěny od sloupce 10, všechny ostatní jsou odsunovány v případě, že vzroste LEVEL nebo NESTED o 1, o 2 sloupce doprava. To znamená, že se tato

úprava týká bloků DŌ, BEGIN a PRŌC. Příkaz END; který končí uvedené bloky, je zarovnán na úroveň počátku bloku. V případě, že je na řádku více příkazů, neprovádí se jejich rozepsání do více řádků, ale posouvají se jako jeden příkaz.

Příklad:

```
          ŌN ERRŌR
          BEGIN;
            CALL HPRINT(LN,A);
            STŌP;
          END;

          IF A > 10
          THEN DŌ;
            A1=B1;   X1=Y1;
            C1=D1;
            IF A2=A3
            THEN DŌ;
              A4=A5+A6;
              A7=A8+A4;
            END;
            CALL TEST;
          END;
TEST:     PRŌC;
            PUT SKIP EDIT(A,B,C)(A);
            IF C=D
            THEN X=Y;
          END;
```

3.4 ÚPRAVA POZNÁMEK ZA INSTRUKCÍ

Provádí se úprava poznámek za instrukcí, stejně jako poznámek za deklaracemi (bod 3.1).
Využitím celé šíře papíru se vytváří souvislý pás komentářů, což ulehčuje orientaci v programu.

3.5 ÚPRAVA POZNÁMEK PŘED INSTRUKCÍ

Pro tyto poznámky platí stejná pravidla pro úpravu jako pro návěští.

3.6 ÚPRAVA PŘÍKAZU, NAPSANÝCH NA VÍCE ŘÁDCÍCH

Pokračovací řádky příkazu začínají o 2 sloupce doprava od pozice příkazu, ke kterému patří.

Příklad:

```
A=POCET||DRUH||JAKOST
  ||ATYP||BTYP
  ||CTYP||DTYP;
```

3.7 ÚPRAVA INFORMACÍ KOMPILÁTORU A ČÍSLOVÁNÍ ŠTÍTKU

Jsou zhuštěny informace STMT LEVEL NEST a k ním je přesunuto číslování štítku. Tiskne se od sloupce 1.

Příklad:

```
STMT L NT ZDROJDA
   2 1  0 0003 11
   3 1  0 0004 08
```

Při použití PREPROCESSORU se provádí doplnění číslování štítku nevýznamnými nulami, protože při opravách je nutno uvádět číslování štítku i s těmito nevýznamnými nulami.

4. ZÁVĚR

Formátování lze provozovat na počítačích IBM 360, IBM 370 a počítačích řady JSEP s operačním systémem OS/MFT, OS/MVT, OS/VS1, OS/VS2 nebo OS/VS1 (OS/VS2) pod řízením VM 370 za předpokladu, že je na dané instalaci zaveden ZN NIKG číslo 1937/74 a 2650/76 Programový systém pro opravy programů ve zdrojové formě na zařízeni DASD. Modul lze použít pro PL/1(F) nebo PL/1 OPTIMIZING kompilátor.

Zavedením této formátovací služby se:

- zvýší produktivita práce programátorů a jejich vedoucích
- sníží se spotřeba papíru eliminací prázdných řádku a hlaviček
- sníží se počet kompilací a zkoušek v důsledku větší přehlednosti programů
- zvýší se přenositelnost programů mezi jednotlivými programátory.

Zatím byla uvedena do provozu prvni verze formátovacího modulu. V příští verzi se uvažuje s rozšířením poskytovaných služeb, zejména o:

- rozdělování více příkazů na jednom řádku
- vypouštění nadbytečných mezer
- úplnou identifikaci příkazu PL/1.

Zavedení formátování tisku kompilátoru PL/1 se v NIKG stalo významným přínosem k racionalizaci programátorské práce.