Miroslav Zvoníček, prom.mat.
I. Brněnské závody KG

SYSTÉM NORMOVANÉHO PROGRAMOVÁNÍ V PL/1

1 Úvodem základní pojmy, princip normovaného programování

Většinu úloh v oblasti zpracování hromadných dat (ZHD) lze charakterisovat následujícím způsobem:

existuje jeden nebo více vstupních souborů, které jsou setříděny podle jistých třídících hledisek, které nazýváme třídící klíče. Z těchto souborů vytváříme

- nové soubory (setříděním, výběrem, změnovým řízením)
- tiskové sestavy (položkové, součtové, výběrové)

Jednotlivé vstupní soubory mohou být značně rozsáhlé, zatímco operace prováděné nad jednotlivými větmi jsou malého rozsahu. Zpracování vstupních souborů spočívá v cyklickém zpracování jejich jednotlivých vět.

Uveďme některé základní pojmy z oblasti ZHD, které budeme dále potřebovat.

Každý setříděný soubor je tvořen skupinami vět, které mají stejnou hodnotu některého třídícího klíče. Velikost skupiny závisí na prioritě třídícího klíče. Klíče s nejmenší prioritou budou vytvářet nejmenší skupiny, klíče s největší prioritou skupiny největší. Skupina vět se stejnou hodnotou klíče největší priority se skládá ze skupin vět určených klíčem nižší priority atd.

Klíčům budeme přiřazovat prioritu v závislosti na pořadí v zadání, tzn., že nejvyšší prioritu bude mít první klíč.

Skupinu určenou i-tým klíčem budeme nazývat skupinou i-té úrovně. Počet úrovní je roven počtu klíčů.

Mějme soubor setříděný dle čísla závodu, střediska a dílny. Největší skupinu budou tvořit věty týkající se vždy jednoho závodu. Tato skupina bude rozdělena pomocí čísla střediska na skupiny vět pro jednotlivá střediska a každá tato skupina bude dělena na nejmenší skupiny týkající se jedné dílny.

Je zřejmé, že změna střediska (změna 2. klíče) je zároveň změnou dílny, protože jde o dílnu v jiném středisku. Obdobně změna závodu je zároveň změnou střediska i dílny. Obecně můžeme říci:

Změna i-tého třídícího klíče znamená změnu skupiny (skupinovou změnu) určené i-tým třídícím klíčem, tedy skupinovou změnu i-té úrovně. Znamená konec skupiny i-té úrovně a začátek další skupiny i-té úrovně. Současně však znamená konec a začátek skupin nižší úrovně: i+1, i+2, ... , k kde k je počet klíčů.

V závislosti na úrovni skupinové změny můžeme provádět jisté činnosti, např. součtování. Při skupinové změně i-té úrovně provádíme činnosti odpovídající úrovním: k, k-1, ... i, takže činnosti provádíme postupně od nejnižší úrovně.

Pokud máme zpracovávat zhotovením zpracováním věty několika vstupních (setříděných) souborů, pak je možné zpracovávat v pořadí, které by vzniklo jejich sloučením (merge). V případě, že máme rozhodnout o pořadí vět se stejnými třídícími klíči, řídíme se prioritou vstupních souborů. Priorita je určena pořadím deklarací: nejvyšší prioritu má první deklarovaný soubor, tzv. řídící soubor.

Zpracování jednotlivých vět se provádí v podstatě shodným způsobem, což znamená, že existuje jistý pracovní cyklus, který se aplikuje na všechny věty vstupních souborů. Budeme jej nazývat pracovním krokem.

1.1 Co je to normované programování

Rozsáhlá analýza úloh v oblasti ZHD ukázala, že existuje obecné logické schéma programu, které je nezávislé na typu úlohy. Na základě tohoto logického schématu se program dělí na několik základních funkčních částí s pevně definovaným obsahem a vazbami mezi jednotlivými částmi.

Uvedené zobecňující funkční schéma programu z oblasti ZHD bylo publikováno firmou UNIVAC jako základ normovaného programování. Zároveň byla vytvořena standardizace tvorby a používání jmen paměťových oblastí (konstant, polí, pomocných pracovních polí, vět, indikátorů) a programových oblastí. Rovněž byla vytvořena pravidla pro organizaci paměti. Tím byla vytvořena norma pro zpracování úloh ZHD, kterou nazýváme normovaným programováním.

1.2 Funkční schéma

Funkční schéma vychází z této základní myšlenky:

1) Existuje několik setříděných vstupních souborů (alespoň jeden), jejichž věty jsou dle daných klíčů setřiďovány do jednoho proudu vět
2) Z proudu vět je postupně vždy další věta předána na zpracování:
   - testování změny třídících klíčů, vzhledem k předchozí větě. Změna třídících klíčů se rovněž nazývá skupinová změna, totál.

- činnost při změně třídících klíčů - totálová činnost
- zpracování věty - detailní činnost

Základní funkční schéma viz obr. na následující straně.

Uvedené schéma je hrubým funkčním schématem programu ZHD v normovaném programování. Ze schématu jsou patrné základní funkční části, které budeme nazývat oblastí. Jsou to:

INITIAL	oblast deklarací a počátečních stavů
INPUT	oblast vstupu vět ze vstupních souborů
SORTING	oblast řazení vět do proudu vět (nalezení věty s minimálními klíči)
TOTAL	testování změny třídících klíčů a případné provedení totálových činností
DETAIL	oblast zpracování věty
OUTPUT	výstup vět do výstupních souborů
FINAL	závěrečné činnosti před ukončením programu

Tyto oblasti v podstatě odpovídají oblastem A, B, C, D, E a F v popisu normovaného programování systémem UNIVAC.

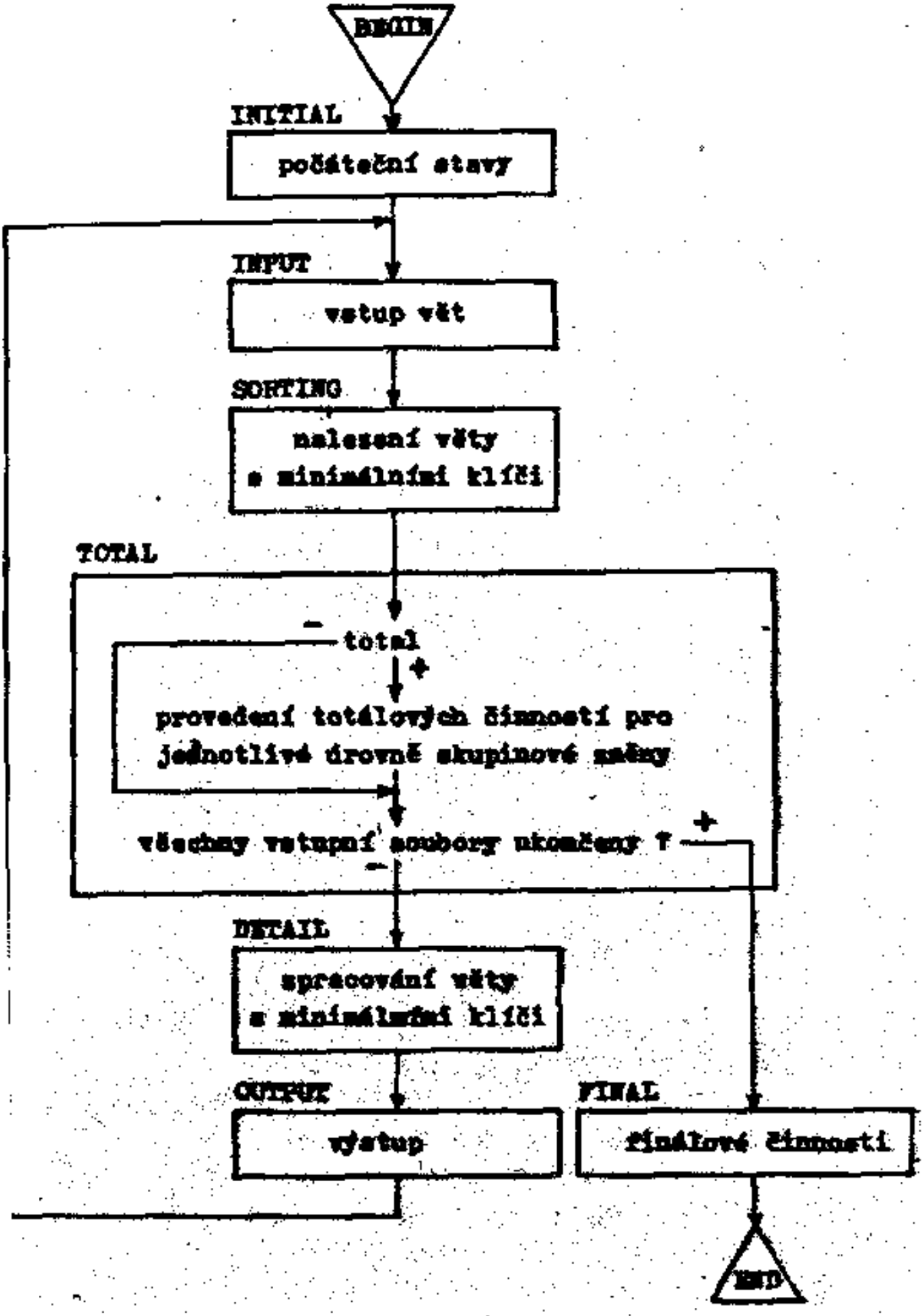
Oblasti TOTAL a DETAIL lze ještě dále dělit:

TOTAL:

TOTAL(1) činnosti při změně skupiny nejvyšší úrovně
.
.
.
TOTAL(počet-klíčů) činnosti při změně skupiny nejnižší úrovně

DETAIL:

DETAIL(jméno-souboru) zpracování vět ze souboru "jméno-souboru"


BEGIN
INITIAL
počáteční stavy
INPUT
vstup vět
SORTING
nalezení věty
s minimálním klíčí
TOTAL
total
provedení totálových činností pro
jednotlivé úrovně skupinové změny
všechny vstupní soubory ukončeny ?
DETAIL
zpracování věty
s minimálním klíčí
OUTPUT
výstup
FINAL
finálové činnosti
END

2 Možnosti aplikace normovaného programování

Normované programování (NP) je u nás dnes již všeobecně známé a používané. Je však důležité, jakým způsobem je aplikováno. Jsou možné následující aplikace:

1) Pouze jako metodika programování úloh ZHD
2) Jako parametrický systém
3) Pomocí makroprocesoru
4) Pomocí účelově orientovaného programovacího jazyka

ad 1)

Nejjednodušší, ale také nejlepší aplikací NP je jeho použití pouze jako metodiky programování úloh ZHD bez jakýchkoliv účelově orientovaných systémových prostředků. Pro programátory je tento způsob značně nesympatický, neboť jim nedává k dispozici žádný nový programovací systém, ale pouze jim přikazuje co "musí a nesmí dělat". Přitom neexistuje jiná než lidská kontrola toho, zda dané předpisy jsou dodržovány. Myslím, že není v lidských možnostech zajistit, aby tyto předpisy byly za všech okolností skutečně dodržovány. Přitom má programátor stále značnou volnost při tvorbě programu a závisí na jeho schopnostech a znalostech jak metodiku NP aplikuje. I přes tyto nevýhody je použití NP jako metodiky značně výhodnější než jeho nepoužití vůbec.

Všechny ostatní možnosti aplikace představují různé způsoby realizace základní myšlenky: existuje obecné funkční schéma úloh ZHD, které je využíváno modifikovaného jakéhokoliv programátorem úlohy pro následující dané úlohy. Můžeme také hovořit o systému modulů (modulechnice) s obecně definovanými funkcemi a vazbami, které jsou upřesňovány dle požadavků dané úlohy. Normovaný program vzniká více nebo méně jako samostatný výsledek použití účelově orientovaného programovacího systému. Existují zhruba tři způsoby jak uvedený systém realizovat. Všechny dávají daleko efektivnější výsledky než

způsob první. Uvědomme si jejich základní vlastnost:

principy normovaného programování jsou nyní implicitní vlastností programovacího systému. Nemusí existovat soustava předpisů jak normovaně programovat, ale pouze jediná zásada: používat daného programovacího prostředku. A to je důležitý psychologický moment: programátor není omezován ve své tvorbě soustavou předpisů, ale dostává k dispozici výkonný programovací prostředek, jehož vlastnosti musí respektovat, a který ho nutí uvažovat dle zásad NP.

ad 2)

Parametrický systém je realizován systémovým programem, který na základě parametrů, zadaných obvykle ve formě tabulek, provádí:

- buď přímo výpočet dané úlohy
- nebo generuje výkonný program (optimálnější výpočet)

Nevýhody parametrického systému jsou:

a) Složité zadávání parametrů specifikujících danou úlohu
b) Špatná čitelnost funkčních vlastností dané úlohy
c) Omezené možnosti oproti procedurálním jazykům
d) Napojení na jiné programovací jazyky: buď neexistuje nebo má omezené možnosti. V rozsáhlých úlohách ZHD potřebujeme někdy programovat dílčí problémy, které parametrický systém nezvládne - např. optimalizační výpočty.

ad 3)

Třetí způsob je realizován makroprocesorem generujícím řídící činnosti oblastí INPUT, SORTING, TOTAL, DETAIL. Generaci je vhodné provádět do programovacího jazyka s výkonnými vstupy a výstupy (FORTRAN, COBOL, nejlépe PL/1), nikoliv v assembleru. Výhody oproti předchozímu způsobu jsou:

a) Poměrně optimální činnost výsledného programu
b) Jednoduché zadávání parametrů pro makroprocesor
c) Jsou k dispozici všechny možnosti cílového jazyka

Nevýhodou je, že prostředky normovaného programování zde nejsou tak nekompromisně zakotveny jako je to možné v parametrickém systému.

ad 4)

Nejvýhodnějším způsobem je aplikace normovaného programování pomocí výkonného programovacího jazyka, obsahující jazykové prostředky plně zajišťující principy normovaného programování. Nejjednodušší možností je rozšíření výkonného programovacího jazyka se silnými vstupy a výstupy. Rozšíření musí odpovídat duchu jazyka a zajišťovat dobrou přehlednost programu. Překlad nových příkazů řešíme předkompilátorem do hostitelského jazyka. Výhody oproti předchozím způsobům jsou:

a) Možnosti značné optimalizace výsledného programu

b) Maximální čitelnost a přehlednost programu zapsaného v rozšířeném jazyku

c) Jsou k dispozici všechny možnosti hostitelského jazyka, obdobně jako u způsobu 3

d) Je možno značně optimalizovat zápis programu a to ve všech oblastech

e) Ucelenost celého systému

f) Prostředky NP je možno zakotvit silněji než u způsobu 3

g) Minimalizace programových chyb

3 Návrh rozšíření jazyka PL/1 o prostředky normovaného programování (NPL)

Jazyk NPL dovoluje definovat :

1) Vstupní soubory , jejich vlastnosti, způsob čtení vět, strukturu vět, klíče, případně součtované položky
2) Výstupní soubory, jejich vlastnosti, způsob výstupu vět, případně stránkování, strukturu vět
3) Tabulkové soubory, jejich vlastnosti, způsob čtení vět, strukturu tabulkových vět

Na základě těchto definic vytváří kompilátor NPL systémovou část (kostru) cílového programu v jazyku PL/1. Vytváří základní obsah - systémové činnosti všech oblastí cílového programu. Programátor má možnost v každé oblasti doplnit systémové činnosti svými činnostmi, které budeme nazývat akce programátora. Určuje např. činnosti při skupinové změně, zpracování věty atd. Akce programátora se programují v jazyku PL/1.

V návrhu NPL je sledována snaha minimalizovat zápis programu v jazyku NPL, učinit jej co možná nejpřehlednější, provádět většinu deklarací systémově (oddělit programátora od deklarací souborů, vět, prac. buněk a polí v jazyku PL/1).

Nové příkazy jsou syntakticky podobné s příkazy jazyka PL/1 a jsou předznačeny znakem "*".

3.1 Jazyk NPL

Základní překladovou jednotkou systému NPL je vnější procedura obdobně jako v jazyku PL/1. V jednom programu lze libovolně kombinovat vnější procedury v PL/1 s vnějšími procedurami v NPL.

3.2 Tvar vnější procedury

Vnější procedura v jazyku NPL má tvar :

```
[předpona-podmínka:]...jméno-vstupu:
PROCEDURE [OPTIONS(seznam-doplňků)][(seznam-parametrů)];
oblast-INITIAL
oblast...
```

V seznamu doplňků může být navíc doplněk CSORT(jméno). Tento doplněk určuje systémovou kontrolu setříděnosti vstupních souborů. Je-li zjištěna nesetříděnost, předá se řízení na místo programu určené jménem v doplňku CSORT. Jméno je buď konstanta-návěští nebo proměnná-návěští. Na takto určeném místě umístí progamátor reakci na nesetříděnost.

3.3 Rozšířená deklarace souboru

Rozšířená deklarace souboru dovoluje definovat vstupní, výstupní a tabulkové soubory tak, aby kompilátor NPL mohl vytvořit v jazyku PL/1 :

- deklarace souborů a příslušných vět
- deklarace pracovních buněk a polí
- čtení tabulkových souborů

} v oblasti INITIAL

- systémové činnosti oblastí INPUT, SORTING,...,FINAL

Musí být uvedena rozšířená deklarace alespoň jednoho vstupního souboru.

3.3.1 Vstupní soubory

Rozšířená deklarace vstupních souborů má tvar:

```
%FILE(jméno)[TITLE(skal.-výraz)]INPUT
            [jádro-GET
            [CALL jméno-proc[(seznam-par)]]
             STRUCTURE(popis-věty)
             KEYS(seznam-klíčů)
            [ACCUMULATE(seznam-sečtovaných-dat)];
```

jméno určuje programové jméno souboru

TITLE(skal.-výraz) má obdobný význam jako v příkaze OPEN jazyka PL/1

jádro-GET určuje, že soubor je typu STREAM a zároveň určuje čtení věty ze souboru. Má tvar:

```
{LIST(seznam-dat)
{EDIT[(seznam-dat)(seznam-formátů)]...}
```

Tvar jádra-GET odpovídá přípustnému tvaru příkazu GET danému implementací jazyka PL/1

CALL jméno-proc[(seznam-par)] určuje, že čtení jednotlivých vět ze souboru bude realizováno tímto příkazem. Procedura jméno-proc musí naplnit větu určenou doplňkem STRUCTURE

Pokud není jádro-GET uvedeno, jde o soubor typu RECORD!

popis-věty určuje složení věty souboru. Jeho tvar a obsah je shodný s popisem struktury v jazyku PL/1. Na úrovni 1 je uvedeno jméno věty bez jakýchkoliv atributů. V akcích programátora se dovoláváme jednotlivých položek naposledy načtených vět pomocí jmen určených v popis-věty. Deklarace věty a jejich položek je provedena systémově

seznam-klíčů obsahuje určení jednotlivých třídících klíčů, dle kterých je daný vstupní soubor setříděn. Třídící

klíč může být obecně výraz, který není zapsán pomocí více jak 30 znaků. Toto omezení lze obejít tak, že jako třídící klíč zapíšeme volání funkce. Ve většině případů zadáváme klíč jako proměnnou, která je položkou věty. Jednotlivé proměnné ve výrazu určující klíč nemusí být položky věty souboru. Jestliže nejsou, musíme je sami deklarovat pomocí deklarace DECLARE jazyka PL/1 v oblasti INITIAL.

seznam-součtovaných-dat určuje jednotlivé součtované hodnoty. Součtování je systémová činnost, která se provádí v oblasti DETAIL(jméno) - nejnižší stupeň součtu a v oblastech TOTAL(i) - vyšší stupeň součtu. Pokud doplněk ACCUMULATE neexistuje, součtování se v detailních činnostech pro větu tohoto souboru neprovádí.

Prvkem seznamu součtovaných dat může být obecně výraz, který není zapsán pomocí více jako 30 znaků. Obvykle zadáváme součtovanou hodnotu jako proměnnou, která je položkou věty.

Jednotlivé proměnné ve výrazu, určující součtovanou hodnotu, nemusí být položky věty souboru. Pokud nejsou, musíme je sami deklarovat pomocí deklarace DECLARE jazyka PL/1 v oblasti INITIAL.

Konečné součty za skupinu určenou i-tým klíčem jsou k dispozici v oblasti TOTAL(i) v poli $Wi. Součty $Wi(1), $Wi(2), ..., $Wi(s) jsou součty hodnot určených 1, 2, ..., s-tým prvkem v seznamu součtovaných dat.

Vstupní soubory deklarované rozšířenou deklarací musí být sekvenční a vzestupně setříděné.

3.3.2 Výstupní soubory

Výstupní soubory se deklarují podobným způsobem jako soubory vstupní. Výstupní větu je možno překrýt se vstupní větou některého vstupního souboru. Pro tiskové soubory je možno v rozšířené deklaraci určit proceduru pro tisk záhlaví a délku logické stránky.

3.3.3 Tabulkové soubory

Tabulkové soubory jsou vzestupně setříděné sekvenční soubory, které se v oblasti INITIAL systémově načtou do paměti a jsou k dispozici po celou dobu výpočtu. Jejich rozšířená deklarace má opět podobný tvar jako u vstupních souborů.

Pro hledání v tabulce existují dva příkazy jazyka NPL : *SEARCH, *SEARI. První příkaz provádí hledání věty se zadanými klíči, druhý příkaz hledání věty určené intervalem, který odpovídá zadaným klíčům. Pokud se věta v tabulce najde, jsou její položky k dispozici pomocí jmén deklarovaných v popise věty tabulkového souboru.

3.4 Tvar oblastí

Každá oblast začíná příkazem začátku oblasti a končí začátkem další oblasti nebo koncem souboru zdrojového textu. Oblast INITIAL musí být uvedena jako první a musí být uvedena vždy, neboť obsahuje rozšířené deklarace souborů. Pořadí a výskyt ostatních oblastí je libovolný, odpovídá řešení dané úlohy.

Příkazy začátku oblastí :

```
*INITIAL;      začátek oblasti INITIAL
*INPUT;        začátek oblasti INPUT
*SORTING;      začátek oblasti SORTING
```

*TOTAL;	určení akcí programátora při skupinové změně libovolné úrovně (společné totálové činnosti)
*TOTAL(i);	určení akcí programátora při skupinové změně i-té úrovně
*DETAIL;	určení akcí programátora při detailním zpracování věty lib. vstupního souboru (společné detailní činnosti)
*DETAIL(Fj);	určení akcí programátora pro detailní zpracování věty ze souboru Fj
*OUTPUT;	začátek oblasti OUTPUT
*FINAL;	začátek oblasti FINAL

Uvnitř oblasti zapisuje programátor akce, kterými doplňuje systémovou činnost programu v této oblasti. Pokud není potřeba systémovou činnost některé oblasti doplnit, oblast není třeba uvádět (s výjimkou oblasti INITIAL). Její neuvedení ovšem neznamená, že oblast nebude v přeloženém programu existovat, ale to, že v ní budou pouze systémové činnosti.

3.5 Indikátory

Ze systémově deklarovaných pomocných proměnných jsou důležité indikátory, které určují stavy během zpracování dané úlohy. Indikátory dělíme :

1) Indikátory skupinové změny, které určují, že v oblasti TOTAL byla zjištěna změna třídících klíčů
2) Indikátory synchronizace, které určují, že při hledání věty s minimálními klíči bylo nalezeno několik vět se stejnými minimálními klíči
3) Indikátory výstupu, které povolují případně blokují výstup do výstupních souborů

3.6 Příklad použití NPL

Je třeba vytisknout položkovou sestavu materiálu (soubor MATER) ve třídění : zak. číslo, číslo materiálu. Na závěr je třeba vytisknout součet Kčs za celý soubor.

```
TISKS:PROC OPTIONS(MAIN);

%INITIAL;
%FILE(MATER)INPUT STRUCTURE(1 S,2(CM,ZC,KCS,UCET))
             KEYS(ZC,CM)ACCUMULATE(KCS);
%FILE(TISK)OUTPUT PRINT EDIT(CM,ZC,UCET,KCS)
            (SKIP,F(11),F(13),F(8),F(14))
            HEAD(ZAHL);69);
ZAHL:PROC;
     PUT FILE(TISK)EDIT('MATERIAL','ZAK. CISLO','UCET',
         'KCS','LIST',$QP1)(SKIP(3),X(3),A,X(3),A,X(4),
         A,X(9),A,X(24),A,F(5))PAGE;
     PUT FILE(TISK)SKIP(2);
    $QP1=$QP1+1;
END ZAHL;
CALL ZAHL;

%FINAL;
     PUT FILE(TISK)EDIT($SO(1))(SKIP(3),X(35),F(11));
     PUT FILE(TISK)PAGE;
/%KONEC PROGRAMU%/
```

$QP1 je systémová proměnná pro číslo stránky, $SO(1) systémově deklarovaný sumátor pro celkový součet za soubor.

Literatura

/1/ M.Zvonříček,A.Kolečný :Programovací systém pro úlohy hromadného zpracování dat

/2/ Fy dokumentace UNIVAC :Normierte Programierung