RNDr. Stanislav DVOŘÁK
Tesla n.p. Rožnov

UNIVERZÁLNÍ PROGRAMY PRO VYTVÁŘENÍ INFORMAČNÍCH PŘEHLEDŮ A AKTUALIZACI SOUBORŮ

1. ÚVOD

Řízení podnikových činností často vyžaduje operativní zajišťování podkladů pro rozhodování na všech stupních řízení. Proto rychlé dodání požadovaných informací je jeden ze základních požadavků na výpočtová střediska závodů a podniků. V těchto střediscích jsou informace soustředěny ve formě informačních souborů dat. Z těchto souborů musí být požadované informace vybrány a případně dále zpracovány.

Takové použití výpočetního prostředku předpokládá, že
- používané informační soubory budou trvale v aktuálním stavu,
- bude k dispozici relativně jednoduchý aparát, který bez zdlouhavého programování a ověřování dovolí uživateli přístup k jednotlivým položkám těchto souborů.

Uvedené požadavky na výběr položek se konkrétně jeví jako víceméně jednorázové požadavky řídících pracovníků, odborných útvarů nebo i nadřízených útvarů na vytvoření různých informačních přehledů, výkazů a evidenčních sestav. Členění a obsah těchto přehledů je nejrůznější, nelze je předvídat a proto také ani zařadit do rutinních prací. Podobná je proble-

matika aktualizace souborů.

Obvyklý přístup k řešení této problematiky vyžaduje sestavení individuálního programu pro každý jednotlivý požadavek. Tento postup je náročný na čas programátora i počítače (např. už při generování třídícího programu a při ověřování), i když se jedná o velmi podobná zpracování. Řešení úloh tohoto typu nebylo usnadněno podstatně ani programovacími jazyky vyšší úrovně, které jsou orientované procedurálně (COBOL, FORTRAN) a které jinak pro zpracování hromadných dat vyhovují. Důvodem toho je právě odlišnost požadavků a nutnost zápisu všech procedur v každém individuálním případě. Proto s růstem počtu úloh uvedeného typu byly a jsou vytvářeny programovací systémy, které jednodušší formou dovolují specifikovat a zadat výběrová nebo aktualizační kriteria. V následující části si některých takových systémů všimneme přehledně.

2. PROGRAMOVÉ SYSTÉMY PRO AKTUALIZACI SOUBORŮ

Aktualizace souborů dat je nezbytně řešena v systémech řízení databází. Soubory tohoto typu se v referátu nezabýváme; tato problematika by vyžadovala samostatné pojednání. Pro soubory na páskách byly v poslední době vytvořeny systémy, které dovolují obecně několik vstupních a výstupních souborů, které jsou aktualizovány dle zadaných kriterií. Procedurálním systémem tohoto typu je např. systém [1], určený pro počítač Tesla 200.

Systém GZP, který byl pro jednoduchou aktualizaci sestaven v n.p. Tesla Rožnov prom. mat. B. Kropáčem, řeší na základě zadaných parametrů tuto úlohu: Soubor změn se nachází na děrných štítcích, základní soubor, do něhož jsou změny promítány, je na magnetické pásce. Procedura aktualizace vypadá takto: Změnový soubor se čte, setřídí se dle zadaných klíčů a přiřazuje se k souboru kmenovému, přičemž se modifikují záznamy, vkládají nové záznamy a vynechávají původní zá-

znamy kmenového souboru. Vstup změnového souboru je ještě spojen s různými kontrolami (numeričnost, rozsah údajů atd.). Tyto procedury jsou implicitní, uživatel pro ně dodává jen parametry (např. pracovní pásky po třídění), ale žádné procedury nepopisuje.

Program GZP je sestaven pro počítač GE 400 v jazyku FORTRAN a generuje program v jazyku COBOL. Generovaný aktualizační program může být doplněn ještě textem v jazyku COBOL, takže uživatel má k dispozici i celý aparát jazyka COBOL. Program, který GZP generuje, je podle požadavku uživatele buď vypsán, nebo vyděrován do štítků ve zdrojové formě, nebo přeložen a eventuelně ihned proveden.

Systémy aktualizace informačních souborů pro počítače třetí generace urychlují přípravu příslušných programů a zpracování. Pro tyto úlohy lze však ještě docela dobře využít vyšší programovací jazyky, které obvykle jsou v programovém vybavení. Jde totiž o programy relativně stabilní, u kterých dojde k vícenásobnému, opakovanému použití a proto i delší doba přípravy ve vyšším programovacím jazyku nemusí být žádnou zvláštní nevýhodou. Také jde obvykle o menší počet souborů, takže i počet úloh tohoto typu je omezený, přičemž ještě aktualizační procedury mohou být dostatečně standardizovány. Vznik programů tohoto typu je přitom soustředěn zejména na dobu náběhu nových zpracování.

3. PROGRAMOVÉ SYSTÉMY PRO VYTVÁŘENÍ TISKOVÝCH SESTAV A VÝBĚRY INFORMACE

Pravděpodobně první typy specializovaných systémů pro vytváření tiskových sestav jsou různé varianty RG (Report Generator) [2,3]. Tento generátor, navržený v původním tvaru u firmy IBM, patří dnes ke standardnímu programovému vybavení systémů pro zpracování hromadných dat.

Základním rysem tohoto generátoru je implicitní chápání procedur vstupu a výstupu. Uživatel se zabývá jen rozhodovacími kroky a jim příslušnými operacemi, ale nikoliv algoritmem celého procesu, který obvykle připouští
- několik vstupních a několik výstupních souborů,
- přiřazení záznamů dle určitých položek, tzv. klíčů,
- kumulaci dle změn hodnot několika řídících klíčů,
- dvoučlenné aritmetické operace,
- logické operace pomocí tzv. indikátorů,
- řízení formátu výstupu (vytváření záhlaví a pat sestav, číslování stránek atd.).

Obvykle lze zařadit procedury popsané v jiných programovacích jazycích, které se základním programovým modulem komunikují prostřednictvím globálních symbolů.

Úspěch tohoto systému, postaveného vlastně na realizaci vícenásobných součtů, vedl i k jeho zařazení do jazyka COBOL, kde pro popis údajů v sestavách byla vyčleněna tzv. tisková sekce (REPØRT SECTIØN). Pro řízení výstupních procedur jsou nezbytné ještě příkazy v oddílu procedur (INITIATE, GENERATE a TERMINATE) a případně ještě USE-procedury s určením USE BEFORE REPØRTING. Současně specifikace jazyka COBOL obsahují Report Generator ve velmi flexibilní formě, která byla vytvořena teprve revizí základního návrhu z roku 1969 [4].

Pro řadu účelů je RG ještě velmi širokým systémem. RG vyžaduje poměrně podrobný popis položek a jejich tiskových obrazů, řešení aritmetických a rozhodovacích operací není příliš pružné a často je třeba znalosti jiného jazyka pro realizaci zadaných požadavků na obsah výstupní sestavy.

Proto vznikly i jiné systémy pro rychlé řešení výběrů z informačních souborů, kde je podíl implicitních popisů a procedur dále zvětšen.

Sem patří např. programový systém FIND pro řadu ICT 1900 nebo projekt rozsáhlé soustavy IMRADS (Information Management Retrieval and Dissemination System) u firmy UNIVAC. Univerzál-

ní systémy tohoto typu vznikají také pro řízení a údržbu souborů, sdružených v databázích ať už jako vlastní součásti řízení databáze nebo jako nadstavby nad základními přístupovými cestami k informacím v databázi. Jsou to různé dotazovací jazyky (tzv. QUERY-systémy). Jako konkretní příklad tohoto systému práce s databází jmenujeme jen systém MARS (Multiple Access and Retrieval System) firmy Control Data Corporation, realizovaný pro počítače CDC 3300 a CDC 3600. Rovněž jazyk vstupního modulu systému ISIS, vyvinutého ve VVS OSN v Bratislavě, je příkladem účelného řešení této problematiky.

Systémy řízení databází a přístupu k jejich obsahu jsou obvykle konstruovány jako velké samostatné systémy, např. s vlastním supervisorem. Řada výpočetních středisek však byla v uplynulých letech vybavena počítači, které nemají možnosti práce na takové úrovni. Jde o počítače, které tvoří přechod na systémy 3. generace, pracují převážně v monoprogramování a s magnetickou páskovou pamětí jako hlavní externí pamětí (např. T200). Na těchto počítačích je řešení problematiky informačních výběrů méně obecné a v realizaci řady funkcí skromnější, přesto však (zejména v roce 1974) byla uveřejněna řada podkladů o systémech generování tiskových sestav. Specielně, na počítači T200 dnes existuje několik takových systémů, které se odlišují svým pojetím i oblastí použití. Jde o tyto programové systémy:

1° GUS - Generátor uživatelských programů [5];
2° KOMPITA - Generátor tiskových sestav, vycházející z maticové formy výstupu [6,7];
3° GPP - Generátor tiskových sestav, který má nejblíže k systémům typu Report Generator [8].

Je jistě škoda, že úsilí v tomto směru nebylo lépe koordinováno, aby vznikl universálně použitelný systém, který by navazoval těsněji na všeobecně přijímané standardy programovacích jazyků, než to činí systémy 1° - 3° a který by spojoval vhodnou formou jejich pozitivní rysy.

V tomto referátu pojednáme o specializovaném systému pro informační výběry, který byl realizován na počítači T200 ve VS Tesla Rožnov. Systém má označení ARS a vznikl ze snahy dát uživateli k dispozici jednoduchý a názorný aparát, který by bez dalších programovacích prostředků pokryl většinu atypických a nečekaných požadavků v automatizovaném zpracování podnikových činností. Tento systém nemá všechny vlastnosti RG, ale je doplněn jinými funkcemi, takže se dá použít zase tam, kde RG je třeba kombinovat s jinými prostředky. Použití systému ARS je přitom extrémně jednoduché a nevyžaduje kromě znalosti struktury vstupního souboru a obsahu sestavy žádné další znalosti programování. V dalších částech referátu pojednáme postupně o

- algoritmu informačního výběru v systému ARS,
- vstupním jazyku systému,
- programové realizaci systému ARS.

4. POPIS ČINNOSTI PROGRAMU ARS T200

Analýza požadavků na informační výběry v podnikové sféře ukázala, že generátor informačních přehledů musí dovolit

- zadání podmínek, při jejichž splnění se záznam souboru převezme do zpracování;
- zpracování záznamu, které spočívá ve výpisu položek záznamu, resp. ve vyhodnocení aritmetických výrazů, které obsahují tyto položky;
- setřídění výstupních záznamů podle zadaných klíčů, pokud je požadováno;
- kumulování položek ve skupinách, charakterizovaných stejnou hodnotou řídících klíčů.

Tyto požadavky se objevují v naprosté většině uvažovaných zpracování. Požadavky jako

- vyhledávání v tabulkách (table-lookup) a indexace,
- operace se součty na jednotlivých úrovních klíčů, zejména ve vztahu k předchozím nebo k následujícím součtům,

- tisk položky jen při prvním výskytu ve skupině,
- tisk položek z posledního záznamu v součtovém řádku,
- stanovení frekvence výskytu jednotlivých hodnot položky,
- tisk více řádků ze záznamu,
- tisk více řádků v součtovém záznamu

a některé další specializované požadavky dále popisovaný systém neřeší. Záznamem se při tom rozumí jako obvykle skupina údajů, které k sobě logicky náleží.

Konkrétní realizace systému ARS na počítači jiného typu závisí podstatně na použitém třídícím programu nebo generátoru. Např. na T200 je použití standardního třídícího programu na magnetických páskách (TRI 140) nepříliš vhodné, protože jde o program, který omezuje periferní operace v úseku uživatele. Protože systém ARS byl připravován v návaznosti na jiné části software, vyvíjené v souvislosti s instalací T200 v n.p. Tesla Rožnov, byl pro něj k disposici již vlastní třídící program DFSØRT, který
- je sestaven jako podprogram,
- není segmentovaný a zabírá cca 4KB,
- nemá žádné omezení na periferní operace ve vstupním a výstupním úseku uživatele,
- připouští proměnný počet pracovních pásek (3-7).

Tento třídící program užívá polyfázní metody ve fázi slučování řetězů. Řetězy jsou vytvářeny s užitím stromových struktur v první fázi třídění. Ukázalo se, že tento podprogram pracuje asi o 15% rychleji než standardní firemní program TRI140. Autorem třídícího programu DFSØRT je prom. mat. B. Kropáč (VS Tesla Rožnov).

Operace systému ARS probíhají takto:

1. Vstupní část čte definici vstupního souboru a zadání sestav. Tento text se analyzuje a dle něho se generují sekvence instrukcí pro testy výběrových podmínek a přenos položek dat a klíčů do výstupního záznamu. Sestavený výstupní záznam, který odpovídá přijatému záznamu vstupního sou-

boru, se spolu s kódem sestavy a kódem typu záznamu bude zapisovat do výstupního souboru tříděných nebo netříděných sestav (SF, NSF).
Definiční údaje výstupního záznamu sestavy a její další charakteristiky se uloží do souboru definic (DF).

2. Po ukončení analýzy vstupního textu se předá řízení na právě sestavený program, tj. otevře se vstupní soubor, čte se jeho další záznam a analyzuje se použitelnost záznamu v zadané sestavě. Při splnění aspoň jedné výběrové podmínky se vytvoří pro tuto sestavu výstupní záznam, který se dle požadavku třídění zapíše do souboru SF nebo NSF. To se opakuje pro každou zadanou sestavu; počet sestav, které se vytvářejí při jednom čtení vstupního souboru, je omezen pouze rezervovanou pamětí.

3. Po dočtení vstupního souboru se nejprve vypíší způsobem obvyklým v RG sestavy bez požadavku třídění, které jsou proloženě zapsány v souboru NSF. Tento soubor se opakovaně prochází a při každém průchodu se vypisuje jedna sestava. V této fázi se připojí ještě definice tříděných sestav k souboru SF.
Po skončení této fáze je mechanismus, který nese soubor NSF, volný a použije se ve třídění pro pracovní soubor.

4. Následuje setřídění sestav v souboru SF. Vzhledem k doplněné identifikaci (číslo sestavy a typ záznamu) se záznamy v SF roztřídí dle jednotlivých sestav a v nich dle zadaných klíčů. Na začátku záznamů sestavy jsou pak definiční záznamy, které se použijí pro specifikaci formátu tiskového řádku.

5. Výpis tříděných sestav se provádí stejným podprogramem jako výpis netříděných sestav. V tomto podprogramu se pouze opraví vstup do čtecí rutiny. V této fázi se čtou záznamy speciálním vstupem do třídícího programu, který dovoluje napojení výstupního úseku uživatele.

Pokud uživatel nezadal nějaký druh sestav, pak se příslušná část zpracování vynechá, např. třídění odpadne, když ani v jedné zadané sestavě není požadavek třídění uveden. Schematicky ukazuje sled operací v systému ARS obr. 1.

Vzhledem k popsanému rozsahu operací v systému ARS je zřejmé, že významnou složkou systému je vstupní jazyk, který musí dovolit jednoduché zadání všech výše popsaných operací. K popisu tohoto jazyka nyní přejdeme.

5. SYNTAX VSTUPNÍHO JAZYKA

Definice vstupního jazyka jsou uvedeny v Backusově normální formě (BNF); u každého bloku definic je vyložen jejich význam. Zadání požadavků na výběr informací ze vstupního (magnetopáskového) souboru (program) tvoří:

- definice vstupního souboru a jeho záznamu
- [illegible]
- zadání sestavy
- [illegible]

..... opakuje se pro každou sestavu

- standardní koncový štítek.

Pro jednoduchost uvažujeme jen zadání úlohy ze snímače děrných štítků. Počítač má paměť organizovanou v bytech (B) a užívá obvyklé typy interní reprezentace dat na počítačích 3. generace.

A. Vstupní soubor je zadán rozsahem záznamu (v B), počtem záznamů v oloku a identifikací. Ze vstupního záznamu stačí zadat pouze použité údaje; každý z nich se zadá identifikátorem, počáteční adresou (relativně k začátku záznamu), typem údaje a rozsahem.
Identifikátor položky je v rámci zadání jednoznačné označení položky, tvořené maximálně 6 znaky, jimiž mohou být písmena a číslice, přičemž první znak je písmeno.

Uvažují se údaje numerické zhuštěné (typ **P**), numerické znakové (**C**), binární (**B**) a alfanumerické, tj. texty (**A**). Pokud je položku možno sčítat, doplní se do typu **S**. Rozsah položky se zadává ve tvaru $p.z$, kde p (z) značí počet míst před (za) čárkou. U binární položky p udává rozsah v bytech (B), u znakové je p znakový rozsah položky (zde $z=0$). Všechny formáty zápisu jsou volné, zápis přechází spojitě ze štítku na štítek. Pro přehlednost lze ovšem definici každé použité položky uvést na zvláštní štítek. V definicích je možné překrytí údajů (redefinice).

Tedy

⟨identifikátor⟩ → ⟨písmeno⟩|⟨identifikátor⟩⟨písmeno⟩|⟨identifikátor⟩⟨číslice⟩
⟨typ⟩ → **A**|**B**|**C**|**P**|**BS**|**CS**|**PS**
⟨rozsah⟩ → ⟨cc⟩.⟨cc⟩
⟨adresa⟩ → ⟨cc⟩
⟨definice údaje⟩ → ⟨identifikátor⟩⟨adresa⟩⟨typ⟩⟨rozsah⟩

cc značí všude celé číslo bez znaménka.

B. Dalším prvkem jsou <u>konstanty</u>. Konstanty jsou buď numerické nebo alfanumerické a jejich definice v systému ARS souhlasí s definicemi v jazyku COBOL. <u>Numerická konstanta</u> je řetěz znaků, předcházený případně znaménkem + nebo − a obsahující jen číslice a nejvýše jednu desetinnou tečku, která není prvním ani posledním znakem řetězu. Znaménko + lze vynechat.

<u>Alfanumerická konstanta</u> je řetěz znaků ze souboru znaků počítače, který neobsahuje žádnou uvozovku a který je uzavřen v uvozovkách. Tyto uvozovky k řetězu nepatří a do rozsahu alfanumerické konstanty se nepočítají. Tedy v definicích:

⟨znak⟩ → znak ze souboru znaků počítače
⟨číslice⟩ → **0**|**1**|**2**|**3**|**4**|**5**|**6**|**7**|**8**|**9**
⟨cc⟩ → ⟨číslice⟩|⟨cc⟩⟨číslice⟩
⟨celé číslo⟩ → ⟨cc⟩|+⟨cc⟩|−⟨cc⟩
⟨znaménko⟩ → +|−

⟨číslo bez znaménka⟩ → ⟨cc⟩|⟨cc⟩.⟨cc⟩
⟨číslo⟩ → ⟨číslo bez znaménka⟩|⟨znaménko⟩⟨číslo bez znaménka⟩
⟨řetěz⟩ → ⟨znak⟩|⟨řetěz⟩⟨znak⟩
⟨α literál⟩ → "⟨řetěz⟩", kde řetěz neobsahuje ".

C. Definice aritmetického výrazu (aexp) také plně souhlasí s obvyklou definicí. Operace umocňování není zavedena. V BNF je

⟨člen⟩ → ⟨číslo bez znaménka⟩|⟨identifikátor⟩|((aexp))|Q⟨registr⟩
⟨faktor⟩ → ⟨člen⟩|⟨faktor⟩*⟨člen⟩|⟨faktor⟩/⟨člen⟩|⟨faktor⟩:⟨člen⟩
⟨aexp⟩ → ⟨faktor⟩|+⟨faktor⟩|-⟨faktor⟩|⟨aexp⟩+⟨faktor⟩|⟨aexp⟩-⟨faktor⟩

Dělení : značí dělení s odříznutím desetinné části podílu, / dělení se zaokrouhlením.
Pro logické výrazy (lexp) je podobně

⟨relop⟩ → <|=|>|N<|N=|N>
⟨relace⟩ → ⟨aexp⟩⟨relop⟩⟨aexp⟩
⟨lčlen⟩ → ⟨relace⟩|NOT⟨relace⟩|(⟨lexp⟩)|L⟨registr⟩|B0|B1
⟨lfaktor⟩ → ⟨lčlen⟩|⟨lfaktor⟩ AND ⟨lčlen⟩
⟨lexp⟩ → ⟨lfaktor⟩|⟨lexp⟩ OR ⟨lfaktor⟩

Do systému jsou nyní zavedeny podmíněné výrazy, které dovolují alternativní volbu výsledku (např. podmíněné obsazení výstupní položky):

⟨exp⟩ → ⟨aexp⟩|⟨lexp⟩
⟨e⟩ → ⟨exp⟩|IF ⟨lexp⟩ THEN ⟨exp⟩ ELSE ⟨e⟩

exp a e vpravo musí být výrazy stejného typu (tj. současně aritmetické nebo současně logické výrazy). Q-registry a L-registry jsou zóny pro uchování mezivýsledků. Jejich obsazení není lokalizováno v sestavě, kde jsou definovány, ale platí až do nového obsazení.

D. Formát uložení údaje se zadává počtem dekadických míst před čárkou a za čárkou. COLUMN označuje sloupec sestavy, od kterého má být položka umístěna. Příkazy výstupu a pří-

kazy zadání klíčů jsou spolu s přiřazovacími příkazy jediné výkonné příkazy v systému. V této skupině definic máme

<formát> → F <cc>.<cc>
<sloupec> → COLUMN <cc>
<přiřazovací příkaz> → <format> Q<registr> ← <e>. | L<registr> ← <e>.
<příkaz klíčů> → CONTROL KEY <seznam klíčů>.
<seznam klíčů> → <klíč> | <seznam klíčů>; <klíč>
<klíč> → <typ klíče> <formát> <e> [<text>]
<typ klíče> → C | P | S | <prázdný řetěz>
<text> → <řetěz>
<příkaz výpisu> → IF <lexp> THEN WRITE <seznam>.
<seznam> → <položka> | <seznam>; <položka>
<položka> → <sloupec> <formát> <e> [<text>]

C označuje řídící klíč, při jehož změně se tisknou součty položek s S v definici, P je stránkovací klíč, který funguje jako C, ale navíc tisk pokračuje na nové stránce. S je součtový klíč, tj. funguje jako C, ale způsobí vynechání tisku detailních záznamů. Pokud typ klíče není uveden, jde o prostý třídící klíč. Texty v lomených závorkách se doplní do hlavičky sestavy, resp. k hodnotám třídících klíčů. Ostatní formáty mají zřejmý význam.

E. Konečně systém dovoluje předčasné ukončení zpracování v sestavě, v celém zadání a také návrat na čtení dalšího vstupního záznamu. Příslušné příkazy jsou

IF <lexp> THEN EXIT FROM REPORT.
IF <lexp> THEN END OF PROCESSING.
IF <lexp> THEN READ NEXT RECORD. [1]

Některé drobnější úpravy sestavy zadávají ještě následující tři příkazy:

---

[1]) Podtržení ——— označuje části formátu, které jsou volitelné. Jejich vynechání nemění smysl programu jen pokud jde o části zavedené pro lepší srozumitelnost zdrojového textu.

SPACING 1 LINE. .... tisk s jedním volným řádkem;

TOTALS IN KEY LINE.... součty budou umístěny do řádku s hodnotou klíče a počtem případů (a ne do zvláštního řádku);

HEADING FULL. .... na sestavu se opíše celé zadání sestavy. Pokud tento příkaz chybí, opíší se jen štítky komentáře, které začínají uvozovkou v 1. sloupci.

V systému jsou ještě realizovány funkce $INT(a) = [a]$, $MOD(a,b) = a - [\frac{a}{b}]b$ a $ABS(a) = |a|$ ([a] je celá část čísla a). Je také přípustné zkracování zápisu logických výrazů ve smyslu jazyka COBOL.

Zadání sestavy je posloupností přiřazovacích příkazů, příkazů klíčů a příkazů výpisu, přičemž musí obsahovat aspoň jeden příkaz výpisu.

Příklad: Ze souboru DATAMAT se má pořídit sestava s těmito údaji:

číslo materiálu (CMAT),
poč. stav (PSTAV),
příjem (PRIJEM),
výdej (VYDEJ),
konečný stav
procento změny stavu
cena za jednotku (JCENA)
cena zásoby materiálu.

Údaje bez identifikátoru (v závorce) se počítají, ostatní jsou přímo v souboru DATAMAT. Přitom se sestava pořizuje ze záznamy, které splňují současně tyto podmínky:

a) číslo skladu je 12,14,16-22,23 nebo 25 (identifikátor SKLAD),
b) procento změny a stavu je větší než 10,
c) konečná cena zásoby je aspoň 250000 Kčs.

Třídění výstupní sestavy je

a) za sklad ↑ (každý začíná na nové stránce pro možnost rozdělení),

b) dle ceny zásob ↓.

Zadání sestavy může vypadat takto:

```
F8.0 Q1← PSTAV + PRIJEM − VYDEJ.
F8.2 Q2← Q1 * JCENA.
F8.0 Q3← ABS(Q1 − PSTAV).
F3.2 Q3← IF PSTAV = 0 THEN 999 ELSE Q3/PSTAV * 100.
IF (SKLAD = 12 OR 14 OR > 15 AND < 23 OR = 23 OR 25 ) AND
(Q3 > 10) AND (Q2 > 250000) WRITE COL 5 CMAT [CMAT];
PSTAV [POC. STAV]; PRIJEM ; VYDEJ ; COL 46 Q1 [KO
N. STAV]; Q3 [POHYB]; F6.2 JCENA [C/J]; Q2 [CENA].
KEY P SKLAD ; F8.2 99999999.99 − Q2.
```

4. JAZYKOVÁ ÚROVEŇ IMPLEMENTACE

Systém ARS T200 je v současné době implementován v assembleru počítače T200 (jazyk APS). Toto řešení omezuje přenosnost (portabilitu) systému. Proto v souvislosti s požadavkem přenesení na počítače Jednotné řady se uvažuje o versi, kde by

- program ARS byl zapsán v jazyku COBOL,
- generovaný program by byl sestavován v jazyku COBOL a pak zpracováván jako každý jiný program v tomto jazyku.

Toto řešení by snadno dovolilo rozšíření vnějšího media i na diskové soubory a uživatel by mohl ve vlastních úsecích užívat i jiné příkazy jazyka COBOL (cykly, indexace, vyhledávání atd.).

5. LOGICKÁ IMPLEMENTACE

Systém ARS T200 je rozdělen do dvou pracovních překryvů, a to generačního a tiskového. Hlavní překryv obsahuje jen společná data a definici sestavy. Generační překryv čte definici vstupního souboru a definice položek v jeho záznamu, které zařadí do rozptylové tabulky (hash-table). V této tabulce jsou i všechna rezervovaná slova (IF, WRITE, AND, OR, NOT,...) Dále se připraví globální parametry pro celé zpracování.

Poté následuje blok, který probíhá stejně pro každou zadanou sestavu. V něm se provede inicializace parametrů sestavy a čte se první štítek zdrojového programu. Podle zjištěného atomu vstupního textu přejde řízení na některý vstupní blok.

Typické akce těchto vstupních bloků jsou:

1) Alfanumerický literál - přečte se do zóny konstant a jeho charakteristiky se uloží do zásobníkové paměti operandů (stack ADS). Jeho formát se použije jako výstupní, pokud již jeho výstupní formát nebyl specifikován dříve.
2) COL α obsadí poziční registr hodnotou α místo automaticky počítané hodnoty po každé položce.
3) C, P, S obsadí typ klíče v definici výstupního záznamu.
4) Logický registr (Lr) se buď převezme do stacku logických hodnot (Lr na pravé straně) nebo se jeho číslo pro pozdější uložení uchová (Lr vlevo od ←). Stack logických a aritmetických hodnot je oddělen vzhledem ke zkracování. Aritmetický registr (Qr) se zpracovává analogicky, přistupují ale formátové úpravy.
5) Identifikátor položky ze vstupního záznamu způsobí vyhledání odpovídajících charakteristik a generování sekvence pro převzetí do zásobníku aritmetických operandů. Toto převzetí může proběhnout i s konverzí typu, všechny aritmetické operandy se převedou na zhuštěný dekadický tvar.

6) Aritmetické i logické operátory způsobí převzetí údajů o tzv. porovnávací a ukládací prioritě operátoru a testování jeho porovnávací priority proti prioritě vrcholu zásobníku operátorů.
Systém ARS používá této tzv. dvojí priority, což dovoluje jednotně zpracovat všechny operátory. Je-li π porovnávací priorita nového operátoru, π_{top} priorita operátoru na vrcholu zásobníku, jsou tyto možnosti při neprázdném zásobníku operátorů (ØPS):

a) $\pi > \pi_{top}$; pak se nový operátor uloží do ØPS, přesněji: uloží se jeho ukládací priorita a číselný ekvivalent (tím se rozliší operátory stejné priority).

b) $\pi \leq \pi_{top}$; pak se operace příslušná k vrcholu zásobníku ØPS může provést a řízení se předá na generační blok, patřící k číselnému ekvivalentu na vrcholu ØPS. Jakmile generační blok ukončí svoji činnost, předává řízení znovu na řídící mechanismus zásobníku (tj. na porovnání priorit vrcholu zásobníku a nového operátoru; vrchol je nyní snížen). Tento proces pokračuje, pokud neproběhne uložení nového operátoru, nebo pokud není konec příkazu.
Při ukládání logických operátorů se ještě plní operace související se zkracováním.

Také pro plnění programu se užívá dvou zásobníkových pamětí.

Příklad: Jako ilustraci činnosti generačního bloku uvedeme sled operací při sčítání a odčítání. Blok odebere parametry operandů z vrcholu zásobníku ADS; tyto parametry jsou

R_1 P_1 Z_1 T_1 A_1 pro 1. operand,
R_2 P_2 Z_2 T_2 A_2 pro 2. operand.

Význam parametrů je: R_i - rozsah v B, P_i - počet míst

před čárkou, Z_i - počet míst za čárkou, A_i - počáteční adresa údaje. T_i je typ, který se zde neuplatňuje. Generování probíhá takto:

Určení charakteristik výsledku:
$P \leftarrow 1 + \min(30, \max(P_1, P_2))$;
$Z \leftarrow \max(Z_1, Z_2)$;
$R \leftarrow [(P+Z+2)/2]$ (rozsah čísla $P.Z$ v B);

Seřízení operandů:
ADJ (2): $A \leftarrow A_1 + R_1 - R$ (poč. adresa výsledku);
$A'' \leftarrow A'$, $R'' \leftarrow R'$ (uchování adresy a rozsahu);
ADJ (1) a pro $R > R'$ φ_y[XC A(δ), A], $\delta = R - R'$;

Vlastní generování: φ_y[ADD A(R), A''(R'')].

Zde ADJ(i) je podprogram pro seřízení i-tého operandu, který provádí tyto operace:

ADJ(i): Když $Z > Z_i$, určí se $R' \leftarrow [(P_i + Z + 2)/2]$, $A' \leftarrow A_i + R_i - R'$ a generuje se levý posun φ_y[SLD A'(R'), $Z - Z_i$]. Je-li ještě současně $P_i + Z$ sudé a $P_i + Z_i$ liché, generuje se nulování prvního levého půlbytu položky φ_y[NI X'0F', A'].

Pro $Z \leq Z_i$ se pouze uloží $R' \leftarrow R_i$, $A' \leftarrow A_i$.

Nové parametry výsledku jsou $R, P, Z, 1, A$ ($T = 1$ označuje numerickou sčitatelnou položku, což se rozumí u aritmetických výrazů). Operace jsou zapsány v jazyku APS.

7) Po zjištění konce zadání sestavy se generuje sekvence, která zjistí, zda bylo něco zaznamenáno do výstupního záznamu. V kladném případě se generuje přenos výstupního záznamu do příslušného výstupního souboru (soubor tříděných sestav nebo soubor netříděných sestav). Současně se zaznamená definice sestavy do souboru definic. Generování programu tím buď končí (pak se uzavře přenosem řízení na nové čtení vstupu) nebo se přejde na generování programové sekvence pro další sestavu.

8) Tiskový segment kumuluje sčitatelné položky způsobem známým z RG. Pomocí definice sestavy se chybějící sekvence (např. porovnání klíčů, sčítání položek, formátové úpravy atd.) doplní do programu.

Přijatý způsob implementace s užitím dvojích priorit a zásobníkových pamětí dovolil dát celému vstupnímu segmentu přehlednou strukturu, v níž jsou odděleny počáteční operace celého programu, počáteční operace na sestavě, reakce na vstupní abecedu s přenosem řízení na porovnávací mechanismus zásobníku operátorů a generační bloky s vrácením řízení na totéž místo. Zpracování zdrojového textu pro sestavu i celý zdrojový program zase končí přesně definovanými činnostmi, které jsou soustředěny v dalších blocích. Základní schema vstupního bloku je na obr. 2.

6. ZÁVĚR

Z popsaných funkcí systému ARS a ze způsobu jejich implementace je patrné, že řešení různých nahodilých požadavků systémem tohoto typu bude efektivní. Ponecháme-li stranou vedlejší přínosy popsaného řešení (jako úplná eliminace práce programátora, prakticky úplná eliminace analýzy, redukce objemu děrování a výrazné zvýšení operativnosti) a uvažujeme-li jen úspory strojového času, lze říci, že
- úplně odpadá kompilace programu a jeho sestavování,
- úplně odpadá ověřování funkce programu,
- při současném zpracování více sestav se vstupní soubor čte jen jednou pro všechny tyto sestavy.

Z hlediska snadnosti formulace běžných požadavků je podstatné, že ARS dovoluje
- aritmetiku na operandech, které nemusí být celými čísly,
- podmíněné aritmetické a logické výrazy,

- uchování mezivýsledků v aritmetických a logických registrech,
- elementární zadání třídění (pouze klíči).

Výhody tohoto systému, které se zdají zřejmé i z názoru, potvrdila i delší praxe využívání ve výpočtovém středisku n.p. Tesla Rožnov.

Literatura

[1] Jozífko J.: Dokumentace programu GIZD2. Ostroj Opava (1974).

[2] IBM Report Program Generator Specifications. IBM System Reference Library (1966).

[3] Table Generator CDC 3600 General Information Manual. Commonwealth Bureau of Census and Statistics, Adelaide (1968).

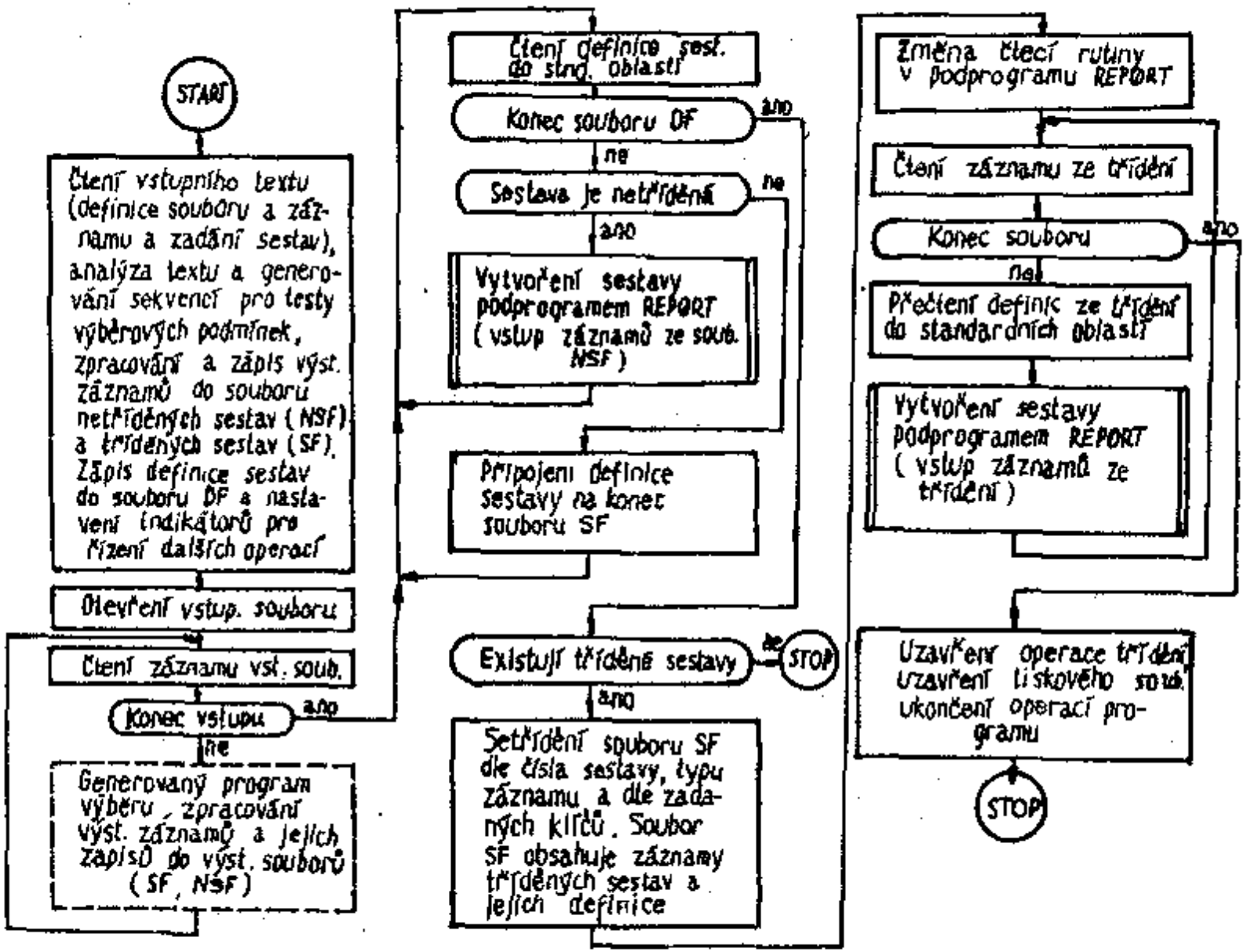
[4] Codasyl COBOL Journal of Development 1970, 110-GP-1b. Canadian Government Specification Board, Ottawa (1971).

[5] Systém GUS. Vítkovické stavby, Ostrava (1973).

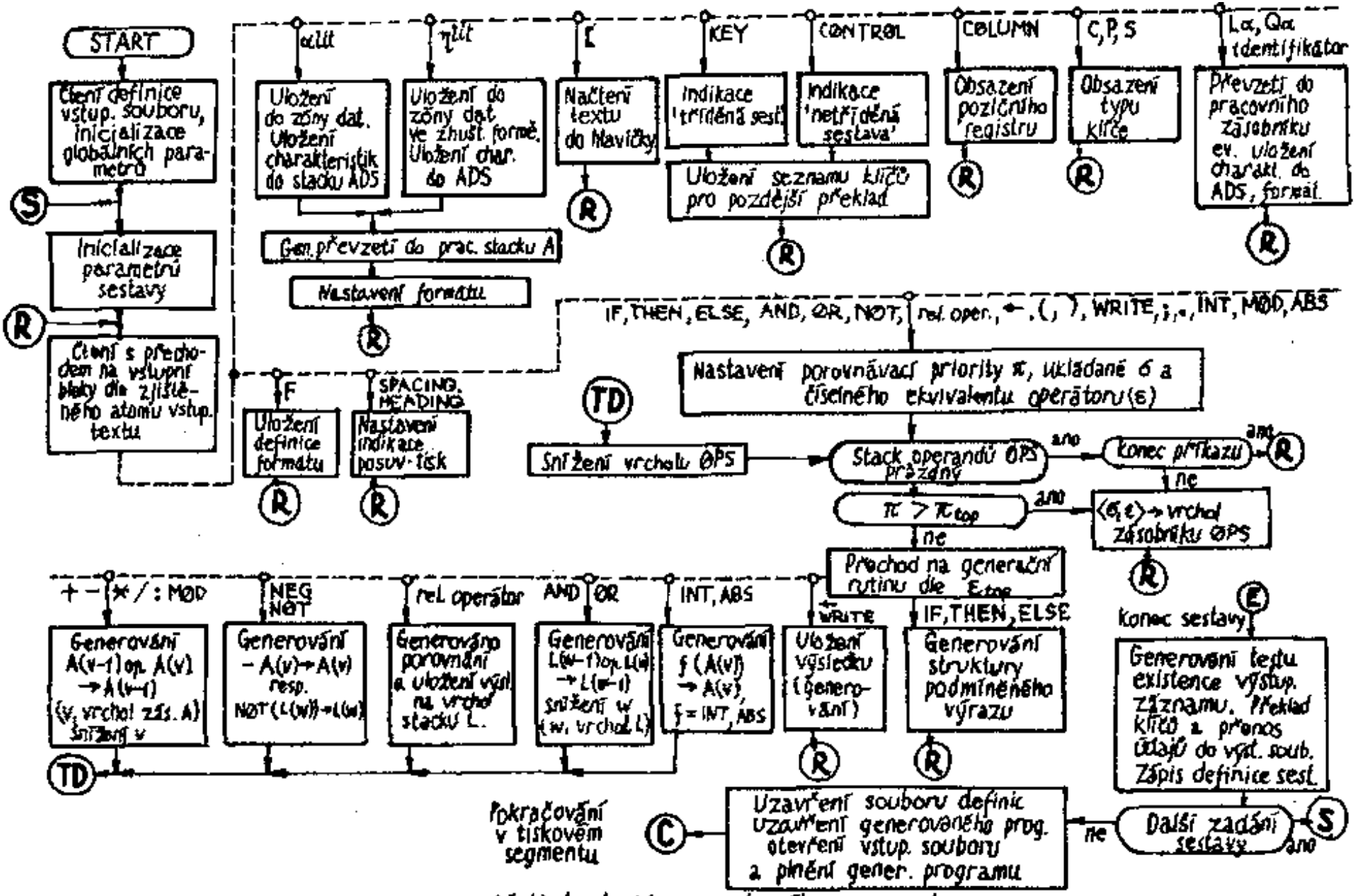
[6] Partyk P.: Tesla KOMPITA. ÚVTT Praha (1971).

[7] Partyk P. a kol.: Tesla KOMPITA T3. ÚVTT Praha (1972).

[8] Kopřiva F.: GPP - parametrické programování pro počítač T200. Dataservis VI., č. 5-6 (1974).



Obr. 1. Přehled operací v programu ARS T200.



Obr. 2. Přehled struktury vstupního segmentu