Petr J I R Í Č E K , prom. mat.

ZVT - OKR Ostrava

Překládač tabulek a číselníků

Úvod

Tento příspěvek a také projekt, který je v něm popsán, jsou založeny na dvou hlavních myšlenkách:

- umožnit jednoduchý a přehledný zápis různých číselníků, kontrol a tabulek určených k prohledávání;
- vytvořit centrální číselník obsahující řadu jednotlivých modulů udržovaných nezávisle na programech; moduly odpovídají jednotlivým číselníkům a vazbám mezi nimi.

1. Všeobecné úvahy

Racionalizace programování v oblasti hromadných dat postupuje kupředu. Standardizují se jména (identifikátory), vztahy mezi analýsou a programováním, způsoby psaní programů.

Zpracování sekvenčních setříděných souborů je dokonale propracováno v metodě normalizovaného programování. To se týká aktualizačních a tiskových programů. Mnohem méně je uděláno pro kontrolní a nahrávací programy. Přitom typické programy tohoto typu jsou velmi těžké a pracné.

Když jsme se touto otázkou zabývali, viděli jsme jednu věc: Pokud jde o aktualizační a tiskové programy, tam metoda normalizovaného programování standardizuje jejich členění a logiku na nejvyšší úrovni, kdežto podprogramy v blocích G, H, I ponechává programátorovi. Programátor kontrolního programu potřebuje však pomoc opačného druhu - pomoc při zápisu jednotlivých kontrol.

Jde zde o adekvátnost způsobu zápisu. Každá úloha potřebuje prostředek umožňující jí přiměřený zápis. Na př. úloze nalezení vlastních čísel matice je fortran přiměřenější než cobol. Pro jinou úlohu - třeba pro aktualizaci kmenového souboru s mezd třemi jinými soubory - je tomu přesně naopak. Jak to je se zápisem typických kontrol?

Příklad typické kontroly: tříznakový údaj SU může nabývat hodnoty 000 nebo hodnot 141 až 146 nebo 149 nebo 151 až 156 nebo 159 nebo jakékoliv hodnoty začínající šestnáctkou nebo pětkou nebo sedmičkou.

Celý princip spočívá v tom, že poskytneme možnost co nejjednoduššího, nejčitelnějšího a přitom stručného zápisu podobných skutečností. To jsme také učinili. Na př. uvedenou úlohu zcela adekvátně zapíšeme tabulkou:

SU
000,141-146,149,151-156,159,16.,5..,7..

Z vyšších jazyků jedině cobol umožňuje obdobně jednoduchý zápis a to pomocí podmínkových jmen úrovně 88.

Zkusme úlohu dále komplikovat: to, co bylo napsáno, platí pro podnik 15, pro podniky 16, 19 jsou zadány jiné hodnoty. Vyjádříme to následovně:

PODNIK	SU
15	000,141-146,149,151-156,159,16.,5..,7..
16,19	000,14.,15.,16.,5..,703-721,8..

Zde již i cobolský zápis pokulhává. Když si uvědomíme, že podobné tabulky v praxi mohou mít stovky řádků, vidíme, že hlavní používané jazyky nám neposkytují prostředek pro scela adekvátní zápis uvedeného typu úloh.

Zde použitý náš způsob zápisu je naopak docela přiměřený a velmi lehce srozumitelný.

Můžeme jej rozšířit i mimo oblast kontrol. Třeba úloha zní vyhledat k číslu příslušný text. Zapíšeme:

ČÍSLO	TEXT
900	FOND ZÁKL. PROSTŘ. A INVESTIC
905	FOND VÝSTAVBY
906	FOND DROBNÝCH PODNIKOVÝCH INVESTIC
⋮	⋮

To již by bylo možné lépe než v předchozích případech zapsat i v PLI, ale stále hůře, než odpovídá jednoduchosti úlohy.

Jestliže způsob zápisu, který jsem naznačil, popíšeme pomocí soustavy pravidel, můžeme vytvořit programové vybavení, které jej umožní. K tomu jsme také přikročili a o tom pojednává tento příspěvek.

Přitom jednotlivé tabulky uvedeného druhu mohou být scela samostatné moduly. Je to umožněno prostředky, které software počítačů třetí generace poskytuje na podporu modularity. Výhody jsou zřejmé: moduly, jejichž zdrojová forma má vysokou dokumentační úroveň, jsou vytvářeny, udržovány a měněny nesrovnatelně lépe než programy, ve kterých jsou používány. Jeden tabulkový modul se dá použít pro řadu programů. Zdrojový výpis modulů slouží jak programátorům tak analytikům, kontrolním skupinám i uživatelům.

Přestože se tento příspěvek zabývá hlavně programovací stránkou celé věci, podívejme se krátce na věc z hlediska analytického. Přinejmenším proto, že podnět k uskutečnění

celého projektu vzešel z odboru systémové analýzy.

Z hlediska analytika jde vlastně o zápis číselníků. Zápis sice formalizovaný, ale natolik čitelný a blízký práci analytiků, že může být vhodnější, aby takové číselníkové moduly udržovali sami.

V souvislosti s přepracováním agendy účetnictví a s pracemi na integraci agend přišli s myšlenkou vést všechny číselníky centrálně na jednom místě. Žádný číselník není jednou provždy pevně dán. Naopak, jak život vyžaduje, podléhají častým, některé velmi častým změnám. Je proto výhodné zařídit, aby pracovníci z revíru hlásili každou změnu na jediné telefonní číslo. Dříve každá změna v číselníku způsobila nutnost oprav řady programů a to zatížilo řadu lidí a zvětšilo možnost opomenutí nebo chyby. Dnes je to vyřešeno tak, že stačí programy, kterých se změna číselníku týká, nanejvýš znovu spojit, "přelinkovat". Ani to není třeba, pokud je použito možnosti, aby program volal číselníkový modul dynamicky.

Lidem určeným k udržování centrálního číselníku slouží příslušné programové vybavení. V něm hraje hlavní úlohu číselníkový překladač.

Vraťme se opět do programování. Překladač lze použít i jinak než pro centrální číselník. Jestliže programátor potřebuje v programu prohledávat nějakou pevnou tabulku, může si napsat tabulkový modul. Ušetří si tak:

- těžkopádnější zápis tabulky v nějakém programovacím jazyku
- programování samotného prohledávání.

Je však třeba pečlivě rozvážit, zda takový tabulkový modul nemá obecnější použití než pro jeden určitý program a nepatří tedy spíše do centrálního číselníku. Kdyby tomu tak bylo, pak by existence soukromého tabulkového modulu zmenšila účinek centrálního číselníku.

```
**********************************************************************  00000100
*                                                                    *  00000200
*        MODUL UTX71 - ZPRACOVANI ZAKLADNICH PROSTREDKU  6.1.1976    *  00000300
*                                                                    *  00000400
*                                  POSLEDNI ZMENA  24.2.1976         *  00000500
*                                                                    *  00000600
**********************************************************************  00000700
@02UTX71,Y=1470-711,X1=3(1-10),X2=10(12-60)                             00000800

CINNOST   INVENTARNI CISLO                    KLIC ODVODU - ODPOVED     00000900

                                                                        * 00001000

...       1312676705,1810176700,1810216702,1910176701,                  9 C00001100
          1911176704,1512676702                                           00001200
          2117976675                                                    5 00001300
          2442632421,2476153960,2476176683,3123864968,                  4 C00001400
          3123865594,31704.....,31705.....,31706.....                     C00001500
          2170I.....,37502.....,624.....                                  00001600
          629.....                                                      3 00001700
          731.....,732.....,73401.....                                  4 00001800
          74905.....,74516.....,74917.....                              3 00001900
          7623.....,76231.....,76254.....,76263.....                    4 C00002000
          76294.....                                                      00002100
          764.....                                                      3 00002200
          .....                                                         1 00002300
635-636,  .....                                                         3 C00002400
721-722,                                                                  C00002500
732-733,                                                                  C00002600
741                                                                       00002700
819       2.....,3.....,4.....,5.....                                   5 C00002800
          6.....,7.....,8.....,9.....                                     00002900
```

Ukázky zdrojových číselníků

```
//OUTX5   ,Y=20(33-49),X1=2(1-2),X2=2(9-10),X3=2(16-17),X4=2(21-22)
*************************************************************************
*                                                                       *
*        MODUL OUTX52D -VYBIRANI TEXTU K ORGANIZACNIM CELKUM 29.1.1976  *
*                                                                       *
*                                        POSLEDNI ZMENA   12.2.1976     *
*                                                                       *
*************************************************************************
//OUTX5   ,Y=20(33-49),X1=2(1-2),X2=2(9-10),X3=2(16-17),X4=2(21-22)

PODNIK  ZAVOD  HS    NS        TEXT HS (NS) - ODPOVED
--------*------*----*---------*-------------------------------------------

..                             $
50      51     11    00        SPRAVNI
               12    00        PROJEKCE
               13    00        UCNOVSKE ZARIZENI
               98    00        FIKTIVNI STR.
        52     21    00        TEZBA UHEL.KALU
               98    00        FIKTIVNI STR.
        53     31    00        STAVEBNI PRACE
               32    00        MECHANIZACE A DOPRA.
               34    00        SPRAVNI-HS
               35    00        ZASOBOVANI-HS
               36    00        OSTAT.CINNOST
               98    00        FIKTIVNI STR.
        54     41    00        STAVEB.PRACE
               42    00        MECHANIZACE A DOPR.
               43    00        PRUM. VYROBA
               44    00        SPRAVNI-HS
               45    00        ZASOBOVACI
               46    00        OSTATNI CINNOST
               98    00        FIKTIVNI STR.
55      00     01              HS-VYROBNI
               02              PLANOGRAFIE
               03              VEDA,VYZKUM A VYVOJ
```

2. Zápis tabulek

Zdrojový tabulkový modul obsahuje jednu tabulku nebo číselník nebo víceúrovňovou tabulku - několik číselníků i s vazbami mezi nimi. Jde o jednotku úplně analogickou zdrojovému programu v některém jazyku. Tabulkový modul je rovněž sekvenční soubor nebo člen knihovny a skládá se z osmdesátiznakových vět. Pozice 73-80 mohou být použity k číslování štítků. Vlastní náplň je v prvých 71 pozicích, přičemž 72. pozice slouží k případnému uvedení pokračovacího znaku.

Jsou zde prakticky stejné možnosti řízení úpravy a vkládání komentářů, jaké poskytuje assembler (EJECT,SPACE,hvězdička).

Zadávající - autor tabulky - musí sám navrhnout rozvrh štítků obsahujících samotnou tabulku a zapsat tento rozvrh do řídícího štítku začínajícího znakem @. Každá z úrovní (podnik, závod, číslo účtu a pod.) má své vlastní pole, t.j. svůj sloupec. Je možno uvádět neúplné hodnoty, na.př. 610.. znamená, že prvé tři znaky pěticiferného údaje mají být rovny 610. . Je možno uvádět rozmezí hodnot, na př. 17-24 nebo třeba 814..-817.. . Vejde-li do příslušného pole více údajů nebo rozmezí, můžeme je uvést za sebou oddělené čárkami. Na př. 3....,41...-42...,47...,51583,7....-84... . Hodnoty různých polí (úrovní) ve stejném řádku spolu logicky souvisí.

Další pravidla pro psaní zdrojových tabulek už nebudu uvádět. Drželi jsme se hlavní zásady: Jestliže některý zápis připouští z hlediska "zdravého rozumu" dvojí výklad, je tvrdě odmítnut. Chceme tím dosáhnout toho, aby byla zcela jasná a jednoznačná funkce každého modulu, jenž prošel bezchybným překladem.

K překladači není vydán chybovník, všechny diagnostické zprávy jsou samovysvětlující.

3. Funkce překladače

Zpracování zdrojového modulu není zdaleka triviální. Co je lehce čitelné pro člověka, není stejně lehce čitelné pro program. Je třeba provést dost komplikovaný rozbor zapsaného - vyhledání oddělovačů (čárek,mínusů...), kontrola správnosti a pod. Proto nemůže být zdrojový modul zpracovávám přímo aplikačním programem. Na př. 200 tisíc hledání v číselníku by pak představovalo 200 tisíc těchto rozborů. Proto musíme nejdříve zdrojový modul přeložit.

Mnohem přirozenější, než pracovat se zdrojovým zápisem, pro program je, pracovat s jakýmsi* seznamem stromového typu. Překladač skutečně vyhotoví strom odpovídající zdrojovému modulu, ale tím jeho činnost nekončí.

Logika zápisu tabulek je taková, že ke každé kombinaci hodnot jednotlivých úrovní se dá najít jediná výstupní hodnota, odpověď. Odpověď může nabývat dvou hodnot (0,1 - 1 = chyba, 0 = správně) nebo více hodnot (ke každému číslu najít text, třeba k číslu podniku název podniku). Tabulkou je tedy definovaná činnost: prohledání, přiřazení, to, čemu se v matematice říká funkce. Proto je překladač koncipován tak, že výsledkem překladu je programový modul, provádějící právě toto prohledání, přiřazení.

To je tady podstatné: překladač vytvoří po přečtení zdrojové tabulky strom a ten pak presentuje jako programový modul, který něco dělá - prohledá a přiřadí. Jinými slovy: použití našeho způsobu nahradí jak zápis tabulky jiným způ-

---

*) slovo "jakýmsi" má zde přesnější význam: nepůjde o obecný vysoce redundantní seznam skládající se z dvojic (car,cdr), ale o speciálnější strukturu více "přišitou na míru"

sobem (třeba v PL1) tak podprogram pro hledání v ní.

Na př. nechť zdrojový číselníkový modul vypadá následovně:

1. úroveň (x_1)	2. úroveň (x_2)	odpověď (y)
....		9
101-102		0
2..		1
	11,13	2
3..	15	3
4..	15	4
5..		5
573		6

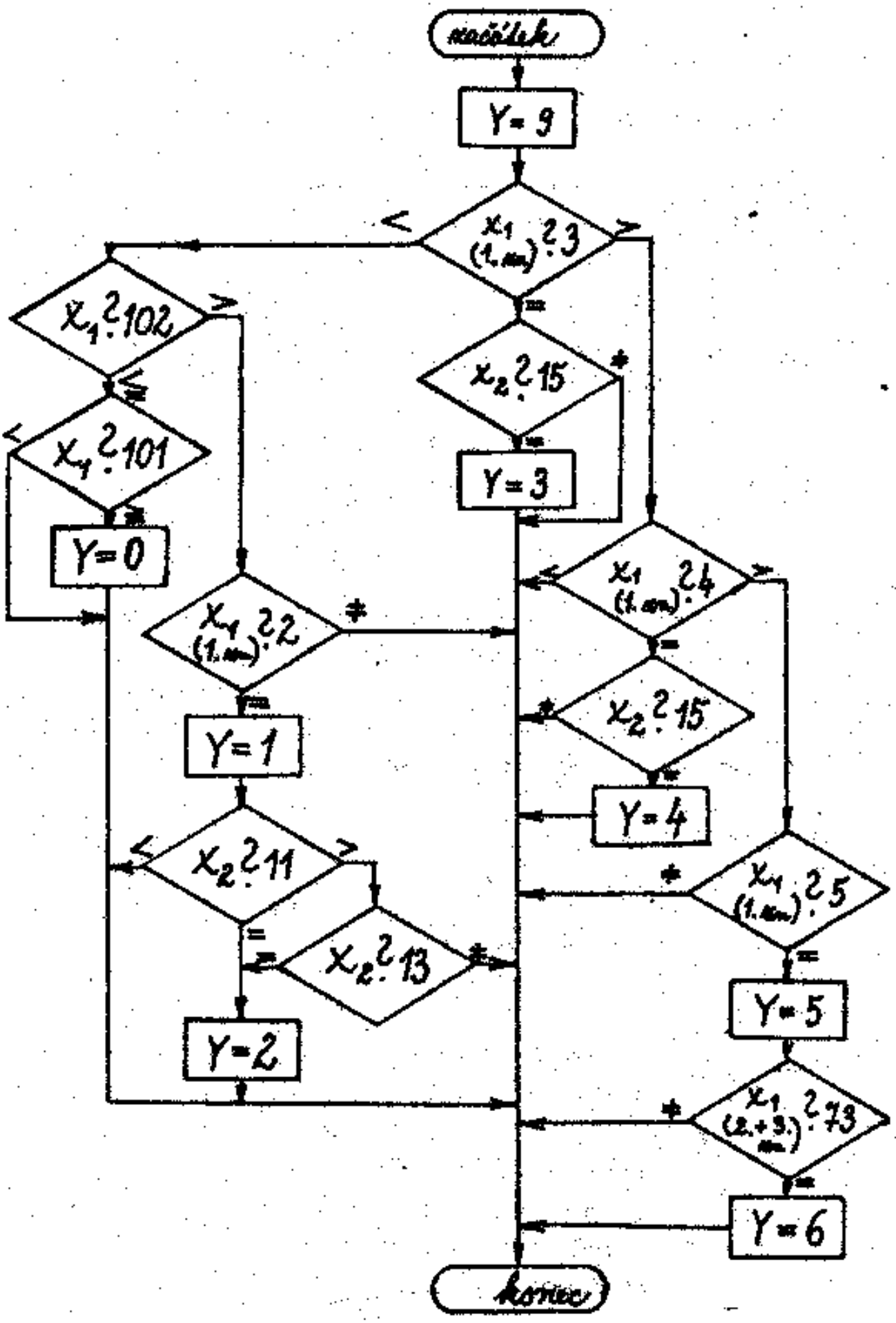
(vpředu musí být navíc štítek@, udávající rozvrh ostatních štítků a informace, jako že x_1 je trojciferné a pod.)

Programátor užívá číselníkový modul velmi jednoduše. V našem příkladě nechť se uvedený modul nazývá UTX02P; Stačí napsat na př.

```
DECLARE 1 VĚTA  ...,
          :
          2 ÚDAJ1  CHAR(3),
          :
          2 ÚDAJ2  CHAR(2),
          :
        ODPOVĚĎ   CHAR(1),
          ;
CALL UTX02P (ODPOVĚĎ,VĚTA.ÚDAJ1,VĚTA.ÚDAJ2);
```

anebo zcela obdobně v cobolu (assembleru,fortranu). Kromě toho je třeba zajistit, aby byl modul UTX02P zahrnut do programu. To provedeme jednoduše připojením centrálního číselníku (nebo jiné knihovny) ke knihovně příslušného jazyka pomocí DD/štítků (způsob concatenate).

Přeložený vypadá modul UTX02P takto:


začátek
Y = 9
x1 (1. vst.) ? 3
<
>
=
x1 ? 102
>
=
<
x1 ? 101
=
Y = 0
x2 ? 15
*
=
Y = 3
x1 (1. vst.) ? 2
*
=
Y = 1
x2 ? 11
<
>
=
x2 ? 13
*
=
Y = 2
x1 (1. vst.) ? 4
>
=
<
x2 ? 15
*
=
Y = 4
x1 (1. vst.) ? 5
*
=
Y = 5
x1 (2.+3. vst.) ? 73
*
=
Y = 6
konec

Vývojový diagram znázorňuje funkci přeloženého tabulkového modulu. Můžeme v něm najít, jaké výstupní hodnoty (y) dostaneme pro určité kombinace vstupních hodnot x_1, x_2. Můžeme si třeba ověřit, zda to odpovídá našemu intuitivnímu chápání příslušného zdrojového zápisu.

Jedna věc je výsledek, jiná postup. Přeložený modul může pracovat zcela jinak nebo také přesně tak, jak je znázorněno. Všimněme si: uvedený vývojový diagram pracuje metodou půlení intervalu, přičemž struktura tabulky je přímo převedena do instrukcí. Tak se dá nejjasněji ukázat, jaké výsledky y dostaneme pro jednotlivé hodnoty x_1, x_2. Kromě toho jedna ze dvou verzí překladače, které dnes existují, dává moduly právě tohoto druhu.

Jiná verze překladače převede zdrojovou tabulku do vhodně strukturovaných dat a "přibalí" k tomu obecný podprogram prohledávání.

4. Kompatibilita a vývoj

Existují nesčíslné možnosti pokud jde o různé způsoby prohledávání pevných tabulek. Můžeme napsat různé překladače produkující z téhož zdroje různě pracující subrutiny.

Z hlediska výsledku ovšem musí přeložené subrutiny fungovat stejně. Postup, jak k výsledkům dojdou, je naopak jejich vnitřní věc. Přesněji, byla by, nebýt různých nároků na zdroje - prohledávací čas, paměť, čas překladu.

Při různých verzích překladače je možno operativně vyvažovat programy pracující s tabulkovými moduly. Na př. nějaký kontrolní program se časem rozrostl tak, že začíná mít kritické nároky na paměť. Aniž bychom dělali změnu v programu nebo ve zdrojových tabulkových modulech, pouhým jejich novým překladem a spojením snížíme rozsah celého programu, třeba za cenu pomalejšího prohledávání. Pro program

s malým nárokem na centrální jednotku je to vlastně zadarmo.

Jestliže někdo chce zavést novou metodu prohledávání pevných tabulek, usnadní se mu její zavedení. Připraví verzi překladače.

V některých případech bude kompatibilita pouze jednosměrná. Třeba pro metodu znáhodnění asi bude třeba vyloučit údaje typu 610... nebo rozmezí 17-24, 81..-83.. a pod.

Pokud jde o budoucí vývoj způsobu zápisu tabulek, řídíme se pravidlem: vše, co platí, bude platit nadále.

Díky osmdesáti členům centrálního číselníkového souboru máme dostatek námětů k rozšiřování možností překladače.

Překladač je napsán pro IBM370/VS1. Bude upraven pro Robotron EC1040/OS.

5. Popis centrálního číselníku

Jde o dva členěné soubory. Zdrojovou knihovnu UTX a load knihovnu UTXL.

Zdrojová knihovna UTX je na soukromém disku. Obsahuje zdrojový tvar čerstvě aktualizovaných modulů. Členy na UTX se přidávají a mění pomocí generátoru zdrojových programů. Jiné programy slouží k pořizování sekvenčních kopií souboru a ke kompresi. Zdrojová knihovna se často celá vypisuje s mnoha kopiemi. Jednotlivé exempláře dostávají zainteresovaní pracovníci.

Load knihovna UTXL je na permanentně nasazeném disku. Díky tomu je možno kdykoliv používat a ladit značné množství programů, které číselník používají. Členy UTXL vznikají překladem zdrojových modulů UTX. UTXL udržuje krok s UTX, pokud jde o aktuálnost. Za každou změnou na UTX následuje překlad

příslušného modulu do UTIL. Pouze chybové překlady neukládají do UTIL - chybný modul se musí opravit a zadat nový překlad.

Existuje testovací program číselníkových load modulů. Specifikujeme jméno load modulu na UTIL a zadáme vstupní data. Testovací program je vytiskne i s nalezenou odpovědí. Jedna z výhod testovacího programu spočívá v tom, že můžeme ověřit práci číselníkového modulu kdykoliv. UTIL je na systémovém disku. Nevýhodou testovacího programu je nutnost zadat vstupní hodnoty.

Uživatelé UTIL jsou ti, kdo píší volající programy. Jejich možnosti jsou dány sběhlostí v práci s linkage editorem. Před spracováním je třeba programy přelinkovat (znovu sestavit). Nejlepší způsob, jak to provést, je výměna modulů přímo v uživatelových programech. Kromě UTIL a uživatelovy load knihovny není třeba žádné další knihovny. Přelinkování (nové sestavení) lze provést hromadně - mnoho programů jedním krokem.

Od nedávna mají uživatelé možnost volat poslední verzi číselníkových modulů dynamicky, v čase GO. Do rutinních jobů je třeba přidat jeden DD-štítek specifikující UTIL. Dynamické volání se dá včlenit do programu bez jeho překompilování, takže napsaný uživatelův program ještě neví, jak bude volat moduly UTIL. Zda v čase linkage editoru nebo dynamicky v čase GO.

Je to založeno na následující myšlence: uživatel specifikuje, které load moduly se mají volat dynamicky. Příslušná procedura vytvoří object moduly stejného jména jako mají specifikované moduly z UTIL. Ty se přednostně začlení linkage editorem do programu před moduly z UTIL a tím vyřeší příslušné reference. Jde to provést opět výměnou modulů přímo v hotovém programu. Nově vygenerovaný modul má následující funkci: při prvním volání použije služby operačního systému a natáhne dynamicky stejnojmenný modul z UTIL. Přitom se modifikuje tak, aby příště přímo předal řízení nataženému jmenov-

ci. Také poprvé, po zmíněné modifikaci, mu předá řízení. Uvedenou proceduru lze použít obecně, bez vztahu k UTKL.

Jazyky volajících programů lze rozdělit do dvou skupin:
- PLl;
- cobol, assembler, fortran;
každá skupina jiným způsobem předává parametry. Proto je při překladu třeba specifikovat, zda bude výsledný modul volán z programu v PLl. Moduly volané jak cobolskými programy tak programy v PLl musí být přeloženy dvakrát, jednou pro cobol, po druhé pro PLl. Platí konvence, že v takovém případě si cobolská verze ponechá původní název z UTX i v UTKL, verze pro PLl má tentýž název rozšířený o písmeno P. Jelikož dvojí provedení v podstatě téhož modulu zabírá dvakrát více prostoru na UTKL, máme v plánu umožnit takový překlad, aby vzniklý load modul měl dva vstupy (entry) - jeden pro cobol, druhý pro PLl. Pro jejich názvy bude platit už zmíněná konvence. Ten z nich, jenž nebude současně jménem členu, bude veden jako alias. Uživatelů se nové pojetí nijak nedotkne.

Kromě toho lze specifikovat, aby modul pro dynamické volání přepracoval oblast parametrů z jednoho typu na druhý.

6. Význam

Již tady bylo řečeno, jaký význam má pro nahrávací a kontrolní programy možnost zapsat tabulky určené k prohledávání, a to zapsat je adekvátním způsobem. Také byl uveden význam centrálního číselníku pro racionalizaci a integraci agend.

Popisovaný projekt má ještě další, vedlejší význam. Je totiž dost vhodný jako první krok k tomu, aby analytici a programátoři začali myslet v dimenzích modularity.

Někdo může namítnout, že z tohoto hlediska nejde o nic

nového. Že uživatelův program je vždy volající a že volání číselníkového modulu je obdobná věc, jako když kompilátor použije knihovny příslušného jazyka (SYS1.COBLIB, SYS1.PL1LIB ...). Opravdu, obojí vyřeší linkage editor automatickým prohledáním příslušných knihoven. Tedy jaká modularita?

Ne, je zde výrazný kvalitativní rozdíl. Použití modulu z knihovny jazyka je vnitřní věc kompilátoru, o které programátor nemusí ani vědět. O použití modulu z UTXL naopak ví velmi dobře. UTXL se může změnit. Volající program beze změny pracuje díky UTXL jindy jinak. Účinkuje zde několik celků, z nichž jenom jeden napsal autor hlavního programu. V tom je ta modularita.

Styk volajících modulů s číselníkovými moduly je velmi jednoduchý. Číselníkové moduly nemají důvod špatně skončit. Prostě něco dosadí jako hodnotu prvního parametru, podle toho, co se v tabulce našlo nebo nenašlo. Není třeba ošetřovat návratový kód. Tím se zjednoduší vzájemný vztah volajícího a volaného. Proto jsou vhodné pro začátek širokého zavedení modulárního programování.