Ing.Miloslav VĚŽNÍK
Adast n.p. Adamov

ŘÍZENÍ PROGRAMU EXTERNĚ DEFINOVANÝMI PRVKY

V oblasti hromadného zpracování dat při použití sekvenčních souborů, je možno sledovat zpracování pro uzavřenou množinu prvků. Pod pojmem prvek rozumějme elementární údaj datové báze, mající vazbu na ostatní údaje.
Provedeme-li předprogramovou analýzu na co možná největším celku vzájemně propojených programů, t.j. úloze, subsystému (agendě) či systému zpracování informací, zjistíme množinu zpracovávaných údajů a jejich vlastností. Důležitou vlastností je právě i vzájemný vztah mezi jednotlivými údaji. Potom vlastně algoritmus zpracování uvedeného celku je definice aritmetických a logických operací s jednotlivými údaji.Tento algoritmus rozdělen dává algoritmy jednotlivých programů.
Další postup je možný v podstatě několika způsoby.
Například vyhrnout rukávy a po řadě přepsat algoritmy do některého z programovacích jazyků.
Nebo také vytypovat si prvky a nalézt co možná nejjednodušší systém záznamu vzájemných vazeb a vlastností.
Sestavíme-li tento záznam do tabulky-matice, a tuto matici umístíme jako soubor na snadno dostupné medium,např.uživatelskou knihovnu, můžeme algoritmus celku a tedy i jednotlivých programů postavit na spolupráci s tímto souborem.
Logické vlastnosti prvků, zakódované v souboru,řídí průchod prvku programem. Existuje-li k prvku konstanta nebo tabulka

konstant, je možno řídit i aritmetické zpracování.
Matice může obsahovat i definiční obor údaje.
Tento je dán buď tabelovaně na uzavřené množině přístupných hodnot nebo zadáním intervalu reálných čísel.
Zavedením definičního oboru nebo kontrolního předpisu je potom možno údaj na vstupu kontrolovat, aniž by jej bylo nutno definovat v programu. Bylo by nereálné sestavit všechny vytypované prvky do jediné matice. Rozdělení na n-matic je nutno provést s ohledem na snadnou manipulaci a aktualizaci souboru.
Pro jeden prvek může tedy existovat více než jedna matice.
Je-li na počítači k dispozici pouze sekvenční přístup, je možno zvolit následující řešení. Zvolíme pevnou délku věty, z níž část bude kód matice. Tento kód identifikuje matici pro program při výběru ze souboru. Zároveň udává i popis obsahu (sloupců) matice, protože každému kódu jsme v předprogramové analýze určili i obsah matice viz příklad 1.

Příklad 1

Kód	Obsah matice	Řádkový index
100	a_{11} a_{12} a_{13} ..... a_{1m}	1
100	a_{21} a_{22} a_{23} ..... a_{2m}	2
100	a_{n1} a_{n2} ..... a_{nm}	n
300	b_{11}	n + 1
300	b_{21}	n + 2
300	b_{k1}	n + k
700	c_{11} c_{12}	n + k + 1
700	c_{21} c_{22}	n + k + 2
700	c_{31} c_{32}	n + k + 3
700	c_{p1} c_{p2}	n + k + p

Je-li jedna matice používána více programy a vznikne potřeba jejího rozšíření pro jeden z těchto programů, přidáme matici další sloupec, bez zásahu do ostatních programů. Délku věty je tedy nutno rezervovat optimálně s ohledem na velikost souboru a možnosti rozšíření.
Pokud již další rozšíření sloupců matice není možné, vytvoříme novou matici s jiným kódem, přičemž v oblasti pracovní paměti programu obě matice spojíme.
Část věty je nutno rezervovat na pořadové číslo řádku-řádkový index viz příklad 1.
Aktualizace souboru je prováděna stejným způsobem jako u knihovny zdrojových modulů, t.j. s odvoláním na řádkový index nebo interval indexů.
Na vyžádání je celý soubor přečíslován.
Nevýhodou popsaného způsobu je nutnost manipulovat s celým řádkem matice při změně jediného elementu např. $a_{i,j}$.
Je to nejen pracné, ale vzniká tím i možnost zanesení chyb do ostatních údajů řádku.
Z tohoto důvodu je výhodné, aby program aktualizace umožňoval i manipulaci s jediným elementem nebo sloupcem matice.
Zadání je pak provedeno řádkovým indexem nebo intervalem indexů a počátkem plus délkou elementu.

Dalších možností stavby souboru je mnoho. Od využití knihovny zdrojových modulů až po databanku. Záleží na každém, kdo uvedený způsob využije.
A nyní několik praktických příkladů obsahu uvedeného souboru.

1) Přiřazení podprogramu

Pro zpracování množiny prvků existuje n-algoritmů. V klasicky psaném programu to jsou větve více nebo méně složitých rozhodovacích struktur. Poskládáme-li uvedené algoritmy za sebe, je nutno přiřadit prvku správný podprogram přepínačem. Přepínač je ovládán hodnotou, definovanou pro každý prvek v matici řídícího souboru. Zavedením více takovýchto hodnot (kódů) pro prvek, je možné i další větvení uvnitř algoritmu (podprogramu).

Ovšem jen do té míry, aby nebyla narušena srozumitelnost programu. Další větvění je lépe řešit rozdělením podprogramu na několik samostatných částí.

2) Výběr podmnožiny prvků

Pokud program zpracovává je část z množiny všech prvků, je možno provést výběr tak, že v matici řídícího souboru zavedeme další údaj. Program pak pracuje jen s prvky, pro něž tento údaj má určenou hodnotu.
Např. Pro vybírané prvky je hodnota údaje rovna 1, pro ostatní je rovna 0.

3) Určení vzájemného vztahu mezi prvky

Tuto možnost je výhodné využít nejen při výpočtu, ale i při vstupní kontrole údajů (logické vazby). Spojovány jsou ty prvky, jejichž hodnota řídícího údaje je stejná.

4) Určení sumátoru

Máme-li v programu provádět součty údajů, uspořádáme všechny sumátory do vektoru (jednorozměrné tabulky). Rozměr elementu tabulky bude dán s ohledem na největší hodnotu součtu, která může nastat. V matici řídícího souboru bude pro každý prvek určen index sumátoru.
Porovnáním sumátorů lze údaj na vstupu kontrolovat na chyby při pořizování dat.
Např. prvky jednoho uživatele budou vždy dodány s celkovým součtem. Prvek-součet bude mít v řídícím souboru index jednoho sumátoru, ostatní prvky index druhého sumátoru. Po načtení všech prvků jednoho uživatele budou sumátory porovnány.

5) Formální kontroly vstupních údajů

Pokud každý prvek datové báze tvoří skupina údajů, pak vstupní soubor obsahuje obvykle sjednocení všech údajů s tím, že u každého prvku mají smysl jen některé z nich. Pro každý údaj sdružený s prvkem určíme jeden sloupec matice řídícího souboru.

Řídícímu údaji přiřadíme hodnotu např.

1 údaj se může
2 musí vyskytovat ve vstupním souboru
3 nesmí

Tím máme vytvořen obecný předpis formální struktury prvku a můžeme tedy na vstupu kontrolovat, zda prvek obsahuje všechny náležitosti.
Jeden sloupec řídících údajů může obsahovat i číslo dokladu, z něhož může prvek vstupovat. Vymezíme tím každému uživateli jeho pole působnosti.

6) Přiřazení textu

Označováním důležitých prvků textem ve výstupních sestavách dosáhneme nejen lepší srozumitelnosti, ale informace budou i lidsky přijatelnější.
Opačným příkladem je vytvoření slovníku, který psanému textu přiřazuje číselné kódy a umožňuje matematické zpracování textů. Významové ekvivalenty slov jsou spojeny rovněž kódem viz bod 3.

7) Externí popis souborů a záznamů

Použití externího popisu je výhodné hlavně tam, kde dochází k častým změnám vstupních údajů.
Vstupují-li údaje z různých formulářů, jejichž počet a formát se často mění, znamená každá změna zásah do programu. Změna formátu vstupního dokladu je zvláště citelná při použití děrné pásky nebo podobného spojitého media.
Udáme-li v tomto případě u každého údaje i pořadí, ve kterém vstupuje z dokladu, je možno změnu formátu nebo přidání nového vstupního dokladu zachytit pouze změnou v řídícím souboru. U děrných štítků je tento údaj ekvivalentní adrese počátku údaje.
Popis je dále dán typem údaje, jeho délkou a event.počtem desetinných míst.

Další možnosti použití řídícího souboru záleží na tom, kdo se jej rozhodne využívat a v které oblasti hromadného zpracování dat.
Soubor může obsahovat adresář zákazníků či dodavatelů, pracovní kalendář, tabulky konstant, čísla účtů, zakázek, druhy mezd, druhy výrobků a jejich vlastností a pod.

Koncepce stávajících jazyků je založena na tom, že relativně stálé informace jsou zakotveny v programu a proměnné informace tvoří vstupní soubory dat.
Prohlédneme-li na program, který využívá uvedený řídící soubor zjišťujeme, že na řízení programu se podílí přepínač řízený hodnotou proměnné. Aritmetické i logické operace probíhají mezi symbolicky definovanými konstantami a proměnnými. Texty a hlavičky sestav jsou vlastně také proměnné a je těžko rozlišit tiskový soubor od ostatních. To znamená, že program určuje jen cíl a způsob jak tohoto cíle dosáhnout. Obsah event.i formu a vlastnosti dat určuje uživatel. Je zřejmé, že velmi záleží na schopnostech uživatele a jeho snaze nabízené možnosti využívat.

V tomto směru je však třeba začít od sebe.

Najít k uživateli cestu a zasvětit jej do práce s výpočetní technikou.
V počáteční etapě postačí, pokud zásahy do řídícího souboru bude provádět programátor ve spolupráci s uživatelem.
Je-li tento soubor dobře definován, lze zásahy do něj provádět i po delší době bez ladění a obav o výsledek.