A N A L Ý Z A Š T R U K T Ú R Y D Á T

Ing. Ivan Schnapp, Výpočtové laboratórium SlPK

1. Úvod

Syatémy na spracovanie dát sa dodnes vyznačujú značnou chybovosťou. Najvážnejšie chyby vznikajú v raných fázach vývoja systému - pri tvorbe funkčných špecifikácií a pri prevode špecifikácií do návrhu systému. Pretože sa týkajú značča základných čŕt systému, majú za následok prerábanie veľkých častí systému; sú to teda drahé chyby. Jedna z možností ako obmedziť ich výskyt je používať metódy, ktoré poméhajú systematicky vytvárať funkčné špecifikácie používajúc jednoznačný, matematicky traktovateľný, no pre užívateľa zrozumiteľný zápis, metódy, ktoré umožňujú automatickú generáciu programov priamo zo špecifikácií.

Predkladaná metóda ponúka systematický postup pre zápis funkčných špecifikácií v tvare orientovaného grafu a dovoľuje z tohto zápisu automaticky generovať veľké časti programu, resp. systému na spracovanie dát. Metódu možno použiť aj pri manuálnom návrhu a vtedy dobre dopĺňa Jacksonovu /1/ a Warnierovu metódu návrhu programov /2/.

2. Popis metódy

Graf štruktúry dát

Do analýzy systému patrí aj stanovenie vstupných a výstupných dát ako aj transformácie vstupných dát na výstupné. Výhodné je začať s výstupnými dátami, pretože predstavujú užitočný výsledok práce systému a tak užívateľ obvykle dokáže ich úplne určiť.

Prekaždý výstupný údaj zistíme, z ktorých dát je odvodený. Ak sú to medzivýsledky, zisťujeme, z ktorých dát sú samy odvodené. Takto pokračujeme v rozklade, kým neprídeme až na vstupné dáta.

Takýto rozklad môžeme znázorniť orientovaným grafom, kde vrcholy zodpovedajú jednotlivým dátam v systéme a hrany grafu vyjadrujú vzťahy odvodenosti medzi dátami. Ak

vrchol A zodpovedá údaju a, vrchol B údaju b, potom graf obsahuje orientovanú hranu (A,B) z A do B vtedy a len vtedy, keď údaj b je odvodený od údaja a. Tento graf nazývame grafom štruktúry dát. Graf je acyklický. Údaje môžu byť skalárne alebo vektorové veličiny.

Ako príklad budeme v ďalšom používať systém, ktorý má mesačne rátať pre každého zamestnanca hrubú mzdu, daň zo mzdy a čistú mzdu, pre jednotlivé oddelenia a celý podnik súčet hrubej mzdy zamestnancov. Štvrťročne bude rátať pre celý podnik súčet hrubej mzdy zamestnancov za uplynulý štvrťrok. Na konci roka bude rátať pre celý podnik a pre jednotlivé oddelenia súčet hrubej mzdy zamestnancov za uplynulý rok; okrem toho vyráta za uplynulý rok pomer súčtu pohyblivej a súčtu pevnej mzdy zamestnancov podniku. Zamestnanci majú mesačnú mzdu. Jedni sú odmeňovaní, tj. mesačne môžu dostať ako odmenu určitú sumu; druhí sú prémiovaní, tj. mesačne dostávajú ako prémiu percentá zo základnej mzdy. Jeden z možných grafov štruktúry dát pre tento systém je na obr. 1.

Spolu s grafom zostavujeme slovník dát, ktorý môže obsahovať názvy dát, ich identifikátory a typy. Pre náš príklad sa nachádza v tab.1.

Graf algoritmu výpočtov

Graf štruktúry dát môžeme interpretovať aj ako graf algoritmu výpočtov. Ku každému vrcholu Y zodpovedajúcemu v grafe štruktúry dát údaju y priradíme operáciu O_Y, ktoré tento údaj počíta. Ku každej hrane (X,Y), ktorá v grafe štruktúry dát vyjadruje odvodenosť y od x, priradíme rad údajov typu FIFO /prvý dnu - prvý von/ a dva celočíselné nezáporné parametre U_{XY} a V_{XY} takto: Každé vykonanie operácie O_X zaradí na koniec radu U_{XY} údajov a každé vykonanie operácie O_Y odoberie zo začiatku radu V_{XY} údajov. Ak údaj y je odvodený z údajov x_i, i=1,2,..,n, a teda v grafe štruktúry dát sú vrcholy Y, X_i, i=1,2,..,n spojené hranami (X_i,Y), i=1,2,..,n, potom operácia O_Y sa môže vykonať len vtedy, keď rad hrany (X_i,Y) obsahuje aspoň V_{X_iY} údajov, i=1,2,..,n. Operácie môžu byť jednoduché, napr. aritmetické, alebo zložité, vyjadrené podprogramami /daň zo mzdy/.

Časť grafu môže teda kolabovať do jedného vrcholu alebo, naopak, vrchol môže expandovať.

Hrany grafu algoritmu vyjadrujú precedenčné závislosti medzi operáciami: ak graf obsahuje vrcholy A a B spojené hranou z A do B, potom operácia O_B nemôže byť vykonaná skôr než O_A, lebo O_A ráta operand pre O_B. I pri zachovaní precedenčných vzťahov všeobecne existuje viac odlišných prípustných postupností vykonávania operácií a tak sa ponúka otázka, či výsledok nie je ovplyvnený poradím operácií; inými slovami, či graf algoritmu je determinovaný.

Radenie operácií

Graf algoritmu výpočtov je zvláštny prípad tzv. výpočtového grafu /3/, ktorý je determinovaný vďaka tomu, že ku každej hrane je priradený rad FIFO. Pretože tieto rady potrebujú teoreticky neobmedzenú kapacitu, nie je vhodné programovo simulovať takýto graf. Ak chceme šetriť pamäť a rezervovať pre každú premennú len jedno pamäťové miesto, musíme dávať pozor, aby sa operácia nevykonala predčasne, so starými dátami, keď má čakať na nové a aby dáta, ktoré sa ešte len majú použiť, neprepísali predčasne novými dátami.

Pre grafy algoritmov hromadného spracovania dát platí, že pre každú hranu grafu aspoň jeden z parametrov U, V sa rovná 1. Graf algoritmu najprv rozdelíme v dvoch krokoch na podgrafy:

1. Odstránime z grafu všetky hrany, pre ktoré $U \cdot V > 1$, takže graf sa rozpadne na množinu podgrafov, kde pre každú hranu $U=V=1$.
2. Pre každý podgraf z tejto množiny vyšetríme, či v grafe algoritmu existuje spojenie, ktoré vychádza z vyšetrovaného podgrafu, prechádza iným podgrafom alebo podgrafmi a znova vchádza do vyšetrovaného podgrafu; presnejšie, či existujú také vrcholy A a C vo vyšetrovanom podgrafe a vrchol B v nejakom inom podgrafe, že existuje spojenie z A do B a z B do C. Ak áno, potom v spojení z B do C existuje hrana (P,R) taká, že R je vrcholom vyšetrovaného podgrafu a P je vrcholom iného podgrafu. Ak $U_{PR} > 1$, množinu vrcholov vyšetrovaného podgrafu rozdelíme na dve

podmnožiny takto: ak existuje spojenie z R do nejakého vrcholu S, potom S patrí do prvej podmnožiny; ak spojenie do S neexistuje, potom S patrí do druhej podmnožiny. R patrí do prvej podmnožiny. Každá z týchto podmnožín indukuje jeden podgraf vyšetrovaného podgrafu. Pre tieto podgrafy opakujeme krok 2, kým existujú popísané spojenia. Tým končí delenie grafu algoritmu na podgrafy.

Pre radenie operácií v podgrafe platí:

1. radenie musí vyhovovať hranám podgrafu,
2. ak sa má operácia podgrafu vykonať n+1-vý raz, musia byť všetky ostatné operácie podgrafu vykonané aspoň n krát.

Podgraf sa teda musí vykonať celý /všetky jeho operácie/, než sa môže začať vykonávať znovu. Preto takýto podgraf môžeme ako celok považovať za "superoperáciu", ktorá sa vykonáva naraz. Pre radenie týchto superoperácií platí: Ak podgraf P_A obsahuje vrchol A a podgraf P_B obsahuje vrchol B, ktoré sú spojené hranou (A,B), potom pre radenie podgrafov P_A a P_B platí to isté pravidlo ako pre operácie O_A a O_B.

Hierarchia úrovní

Takto vzniknuté podgrafy sú na rôznych úrovniach a vytvárajú spolu hierarchiu úrovní. V našom príklade tieto úrovne zodpovedajú jednak hierarchii podnik - oddelenie - zamestnanec, ktorú nazveme organizačnou, jednak hierarchii rok - štvrťrok - mesiac, ktorú nazveme časovou. Ich kartézsky súčin dáva hierarchickú sústavu časovo-organizačných úrovní. Organizačné delenie zodpovedá hierarchickým úrovniam vo Warnierovej metóde. V Jacksonovej metóde sa zasa stretávame s myšlienkou, že pre návrh systémov /tj.časovej hierarchie/ možno použiť rovnaké postupy ako pri návrhu programov /tj. organizačnej hierarchie/.

V našom príklade máme 3x3=9 časovo-organizačných úrovní. Rozdelenie operácií do týchto úrovní je v tab.2.

Vezmime si podgraf na úrovni mesiac - zamestnanec. Operácie tohto podgrafu - ich popis a programová realizácia je v tab.3.

Možné radenie operácií je takéto: 30, 27, 25, 22, 21, 16, 15. Ak vtomto poradí zoradíme programové realizácie operácií, dostaneme tento podprogram /vzmyslo časť programu zodpovedajúca podgrafu/:

```
DENNA_MZDA=MESACNA_MZDA/FOND_PC;
PEVNA_MZDA=DENNA_MZDA*(DNI+SVIATKY);
PREMIA=PEVNA_MZDA*PERCENTO;
IF KATEGORIA='ODMENOVANY' THEN
  POHYBLIVA_MZDA=ODMENA;
ELSE
  POHYBLIVA_MZDA=PREMIA;
HRUBA_MZDA=POHYBLIVA_MZDA+PEVNA_MZDA;
DAN_ZO_MZDY=DAN(HRUBA_MZDA,VEK,POHLAVIE,
  RODINNY_STAV,POCET_VYZIVOVANYCH);
CISTA_MZDA=HRUBA_MZDA-DAN_ZO_MZDY;
```

Tento podprogram má určitý nedostatok v tom, že PREMIA sa neráta v ELSE vetve. Možno ho takto optimalizovať: Nech operácia vetvenia O_Y je priradená k vrcholu Y. Vieme priradiť k vetvám v programe hrany vchádzajúce do Y. Ak hrana (X,Y) zodpovedá vetve x v operácii O_Y, zistíme množinu všetkých takých vrcholov v grafe, že všetky spojenia z nich vychádzajúce končia v tejto množine alebo prechádzajú vrcholom X a až po X prechádzajú len vrcholmi množiny. Potom všetky operácie zodpovedajúce vrcholom množiny môžeme zlúčiť a presunúť do vetvy x operácie O_Y.

V našom prípade operácia O_{22} bude vyzerať takto:

```
IF KATEGORIA='ODMENOVANY' THEN
  POHYBLIVA_MZDA=ODMENA;
ELSE
  POHYBLIVA_MZDA=PEVNA_MZDA*PERCENTO;
```

Podobne možno zlúčiť niektoré sekvencie operácií, ako 15, 16 alebo 27, 30.

Je logické spájať podgrafy a im zodpovedajúce podprogramy podľa časových úrovní. Potom každej z úrovní mesiac, štvrťrok, rok zodpovedá 1 program. Cykly v časovej hierarchii sa dajú realizovať viacerými spôsobmi. Jedna možnosť

spočíva v tom, že prevádzkovateľ systému pri každom spracovaní sám vyberá vhodnú zostavu programov. Ďalšia možnosť je napísať hlavný program , ktorý privoláva programy jednotlivých časových úrovní napr. podľa čísla mesiaca alebo podľa stavu počítadla v systéme, ktorý sa zväčšuje o 1 pri každom spracovaní /mod 12/.

Cykly v organizačnej hierarchii, teda v programoch, sa realizujú ľahko, pokiaľ V je konštanta, napr.:

DO I=1 TO V; .... END;

Ak sa V mení, tj. vstupné súbory sa menia, potom pre realizáciu v konvenčnom jazyku treba podmienky skončenia cyklov. Odvodenie týchto podmienok rieši tak Warnierova ako Jacksonova metóda. Z grafu sa odvodiť nedajú, preto sa nimi tu zaoberať nebudeme.

Hrana z nižšieho podgrafu do vyššieho znamená aj to, že operácia na nižšej úrovni ráta operandy pre operáciu na vyššej úrovni, ktoré ich potrebuje presne V. Tieto operácie takmer bez výnimky vykonávajú súčty. V zásade sa môžu realizovať dvomi spôsobmi. Buď sa rezervuje V pamäťových miest, na úrovni nižšieho podgrafu, teda v cykle, sa napĺňajú miesta a aktualizuje pointer a na vyššej úrovni sa urobí súčet; alebo sa súčet nuluje na vyššej úrovni a operandy sa pričítajú k súčtu na nižšej úrovni, tj. v cykle nižšieho podgrafu hneď za operáciou ich výpočtu. V organizačnej hierarchii sa vždy používa druhý spôsob. Prvý sa môže použiť v časovej hierarchii keď chceme uchovať hodnoty operandov až do nového cyklu.

Prebrali sme spojenie úrovní smerom z nižšej do vyššej. Spojenie smerom naopak je triviálne. Určité problémy predstavuje spojenie dvoch podgrafov tej istej úrovne, tj. takých, ktoré spája hrana s parametrami U=V=1. Takýto prípad vzniká napr. pri výpočte podielov na hospodárskych výsledkoch. Podiel zamestnanca nacelkovej sume je daný pomerom jeho ročnej mzdy k súčtu ročných miezd všetkých zamestnancov. Graf algoritmu výpočtu podielov je na obr.2a. Vrcholy sú k dátam priradené takto:1 je súčet ročných miezd, 2 je pomer ročnej mzdy zamestnanca a súčtu ročných miezd, 3 je podiel, 4 je suma určená na podiely /číta sa

len raz pre všetkých zamestnancov, preto $U_{4,3}$=z, kde z je počet zamestnancov/, 5 je ročná mzda zamestnanca. Najprv rozdelíme graf na podgrafy. V kroku 1 sú to podgraf s vrcholmi 1 a 4 a podgraf s vrcholmi 2, 3, 5 a hranami (2,3), (5,2). V kroku 2 zistíme spojenie 5,1,2 a druhý podgraf rozdelíme na podgraf s vrcholmi 2, 3 a hranou (2,3)a podgraf s vrcholom 5.

Pretože existuje spojenie 5, 1, 2, O_5 musí pre O_1 pripraviť z operandov, ktoré sa musia uložiť do radu hrany (5,2), kým O_1 neskončí a nezačne O_2. Číslo z je súčin parametrov V všetkých hrén spojenia 5, 1, 2. Sú možné 3 riešenia:

1. Pokiaľ je z malé, uložiť operandy v operačnej pamäti.
2. Uložiť operandy do externej pamäti.
3. Transformovať graf algoritmu tak, že sa duplikuje tá časť grafu, ktorá ráta údaj 5. V našom prípade je to len vrchol 5 /pozri obr.2b/.

Pamäť systému

Doteraz sme predpokladali, že systém nemá žiadnu pamäť /okrem medzisúčtov v časovej hierarchii/ a všetky dáta potrebné na výpočet sa zadávajú ako vstupy pri každom chode systému. V praxi to zväčša býva inak. Ako príklad uvedieme zjednodušený výpočet nemocenských dávok za práceneschopnosť. Musíme ho rozdeliť na výpočet dávok za PN, ktorá vznikla v účtovanom mesiaci a dávok za PN, ktorá začala skôr a trvá až do účtovaného mesiaca; na výpočet sa totiž používa tá mzda, daňová kategória a dĺžka zamestnania ktorá bola pri vzniku nároku na nemocenské. Na výpočet dávok za PN, ktorá pokračuje do účtovaného mesiaca teda potrebujeme poznať denné dávky, ktoré boli pri začatí PN. To značí, že medzi dvomi chodmi ich musíme mať niekde poznačané. Logické je teda vybaviť systém pamäťou.

Graf algoritmu výpočtu dávok je na obr.3. Vrchol 1 je súčet dávok, 2 je dávky za novú PN, 3 je dávky s nižšou sadzbou /v prvých 3 dňoch/, 4 je počet dní s nižšou sadzbou, 5 je denná dávka nižšej sadzby, 6 je dávky vyššej sadzby, 7 je počet dní vyššej sadzby, 8 je denná dávka vyššej sadzby, 9 je dĺžka zamestnania, 10 je maximálna

dávka, 11 je mesačná mzda, 12 je daňová kategória a vrcholy 13 až 19 pre pokračujúcu PN zodpovedajú vrcholom 2 až 8 pre novú PN.

Pamäť pre dávky nemôžeme realizovať tak, že spojíme hranami vrcholy 5 a 16, 8 a 19 a priradíme im rady FIFO, lebo nevieme určiť parameter U. PN môže mať rôznu dĺžku, takže je potrebný rôzny počet údajov; ak sa behom jednej PN nespotrebujú, budú sa nesprávne používať aj pri pokračovaní ďalšej PN. Pamäťový údaj sa raz zapisuje, raz číta, preto by sa mal v grafe objaviť raz ako vstupný údaj, raz ako výstupný. Graf algoritmu v takej forme však nemusí byť determinovaný. V našom príklade dve prípustné postupnosti operácií 10,9,5,8,3,6,2,14,17,13,1 a 14,17,13,10,9,5,8,3,6,2,1 dávajú rôzne výsledky. Preto musíme graf doplniť hranami /napr. (14,10) a (17,10)/, ktoré neznamenajú, že operácia O_{10} potrebuje operandy z O_{14} a O_{17}, ale len zaručujú správne radenie operácií.

Ak teda chceme zaručiť determinovanosť grafu algoritmu pre systém s pamäťou, musíme zvlášť analyzovať precedenčné vzťahy medzi zápisom a čítaním pamäťových údajov.

3. Záver

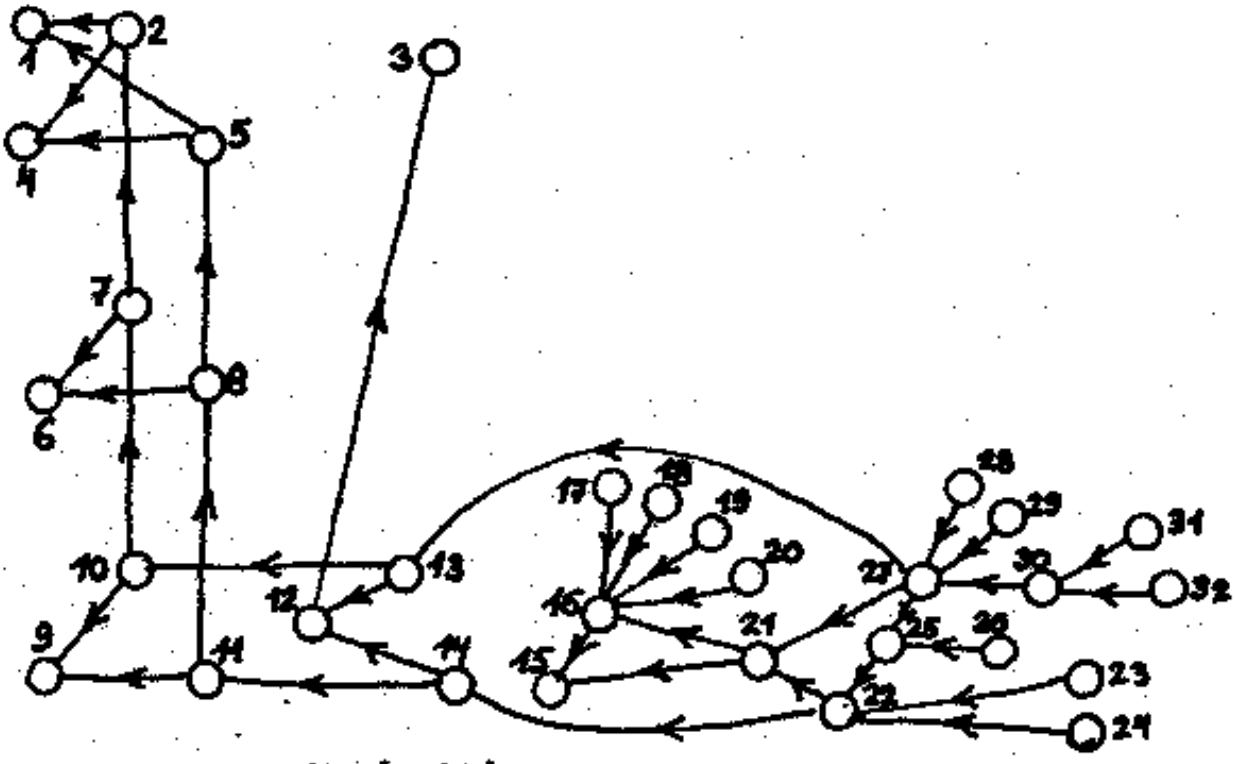
Predložená metóda je vhodný doplnok Warnierovej a Jacksonovej metódy. Dovoľuje automaticky generovať časti programov. V ďalšom chceme skúmať možnosti automatickej generácie podmienok skončenia cyklov a vytvárania logických vstupov z fyzických súborov.

4. Literatúra

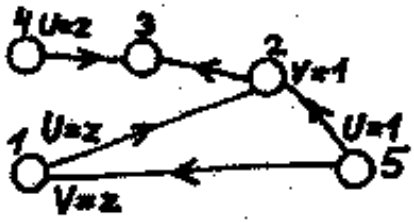
/1/M. Jackson: Principles of program design, Academic Press, 1975

/2/J. Warnier: Logical construction of programs, H. E. Stenfert Kroese, 1974

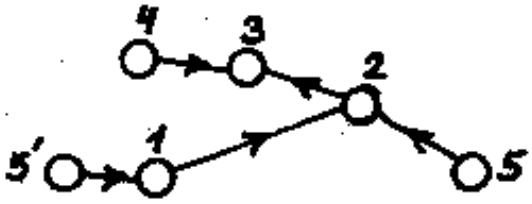
/3/R. Karp, R. Miller: Properties of a model for parallel computations: determinacy, termination, queuing, SIAM J. Appl. Math., Vol. 14, No. 6, November 1966



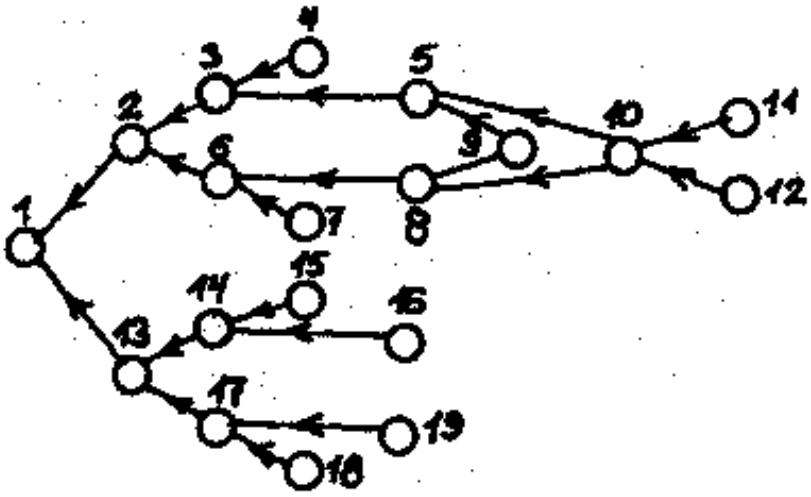
Obrázok 1



Obrázok 2a



Obrázok 2b



Obrázok 3

vrchol	názov údaja	identifikátor	typ
1	pomer poh. a pev.mzdy	POMER	FIXED DEC(4,3)
2	roč.pev.mzda za pod.	R_POD_PEVNA	FIXED DEC(7)
3	roč.hr.mzda za odd.	R_ODD_HRUBA	FIXED DEC(7)
4	roč.hr.mzda za pod.	R_POD_HRUBA	FIXED DEC(7)
5	roč.poh.mzda za pod.	R_POD_POHYB	FIXED DEC(6)
6	štvrť.hr.mzda za pod.	S_POD_HRUBA	FIXED DEC(6)
7	štvrť.pev.mzda za pod.	S_POD_PEVNA	FIXED DEC(6)
8	štvrť.poh.mzda za pod.	S_POD_POHYB	FIXED DEC(5)
9	mes.hr.mzda za pod.	M_POD_HRUBA	FIXED DEC(6)
10	mes.pev.mzda za pod.	M_POD_PEVNA	FIXXD DEC(6)
11	mes.poh.mzda za pod.	M_POD_POHYB	FIXXD DEC(5)
12	mes.hr.mzda za odd.	M_ODD_HRUBA	FIXED DEC(5)
13	mes.pev.mzda za odd.	M_ODD_PEVNA	FIXED DEC(5)
14	mes.poh.mzda za odd.	M_ODD_POHYB	FIXED DEC(4)
15	čistá mzda	CISTA_MZDA	FIXED DEC(4)
16	daň zo mzdy	DAN_ZO_MZDY	FIXED DEC(4)
17	vek	VEK	FIXED DEC(2)
18	počet vyživ. osôb	POCET_VYZIVOVANYCH	FIXED DEC(1)
19	rodinný stav	RODINNY_STAV	CHAR(4)
20	pohlavie	POHLAVIE	CHAR(4)
21	hrubá mzda zamest.	HRUBA_MZDA	FIXED DEC(4)
22	pohyb. mzda zamest.	POHYBLIVA_MZDA	FIXED DEC(3)
23	odmena	ODMENA	FIXED DEC(3)
24	kategória zamest.	KATEGORIA	CHAR(10)
25	prémia	PREMIA	FIXED DEC(3)
26	percento	PERCENTO	FIXED DEC(3,2)
27	pevná mzda zamest.	PEVNA_MZDA	FIXED DEC(4)
28	odpracované dni	DNI	FIXED DEC(2)
29	platené sviatky	SVIATKY	FIXED DEC(1)
30	denná mzda	DENNA_MZDA	FIXXD DEC(5,2)
31	mesačná mzda	MESACNA_MZDA	FIXED DEC(4)
32	fond pracovného času	FOND_PC	FIXED DEC(2)

Tabuľka 1

	podnik	oddelenie	zamestnanec
rok	1,2,4,5	3	
štvrťrok	6,7,8		
mesiac	9,10,11	12,13,14	15,16,21,22,25,27,30

Tabuľka 2

operácia	programová realizácia
o_{15}	CISTA_MZDA=HRUBA_MZDA-DAN_ZO_MZDY;
o_{16}	DAN_ZO_MZDY=DAN(HRUBA_MZDA,VEK,POHLAVIE, RODINNY_STAV,POCET_VYZIVOVANYCH);
o_{21}	HRUBA_MZDA=POHYBLIVA_MZDA+PEVNA_MZDA;
o_{22}	IF KATEGORIA='ODMENOVANY' THEN POHYBLIVA_MZDA=ODMENA; ELSE POHYBLIVA_MZDA=PREMIA;
o_{25}	PREMIA=PEVNA_MZDA*PERCENTO;
o_{27}	PEVNA_MZDA=DENNA_MZDA*(DNI+ SVIATKY);
o_{30}	DENNA_MZDA=MESACNA_MZDA/FOND_PC;

Tabuľka 3