Ing.Egon KRATOCHVÍL
Výzkumný ústav matematických strojů Praha

PROTAB - 25 A RACIONALIZACE PROGRAMOVÁNÍ

Stěží bychom dokázali vyčíslit čas, který se věnuje programování. Přesto však všichni cítíme, že objem programátorských prací je natolik rozsáhlý, že stojí zato, zabývat se jejich efektivností. O aktuálnosti této úvahy svědčí konečně též existence celé řady metod a technik, které sledují jediný cíl - racionalizovat práci programátora. Strukturní a normované programování, metoda TOP-DOWN, HIPO metoda nebo Jacksonův přístup k programování - to je jen několik příkladů z poslední doby.

Každá z těchto metod má své racionální jádro a každá může sehrát při racionalizaci programování svoji nezanedbatelnou a často i nezastupitelnou roli. To ovšem platí pouze potud, pokud je aplikována tvůrčím způsobem a pokud není pojímána jako univerzální prostředek, který působí samočinně, vždy a za každých okolností.

Snaha o nalézání a popularizaci nových metod programování je záslužná a hodná uznání. Současně bychom však neměli zapomínat i na starší metody, které se v důsledku výskytu nových způsobů programování dostávají do stínu dříve, než jsme stačili pochopit jejich filosofii a všechny možnosti praktického využití.

Proto je správné a pro tento seminář i charakteristické, že se kromě propagace nových způsobů programování vrací i k těm starším, aby je oživil, připomenul a podpořil jejich další

aplikační rozvoj. To je i účelem tohoto příspěvku, který chce zdůraznit znovu příznivý vliv rozhodovacích tabulek (dále jen RT) na racionalizaci programovacích prací. Jeho celkové pojetí akcentuje především praktické hledisko. Racionálnost metody RT ve vztahu k programování je zde proto prokazována hlavně prostřednictvím příkladů, aplikovaných na konkrétním překládači RT - PROTAB-25, který byl vyvinut ve VÚMS PRAHA a jehož vlastnosti, možnosti a charakteristika je popsána v příspěvku J. Chlouby.

Zamyslíme-li se nad činnostmi, které musí programátor udělat při řešení průměrné úlohy, dojdeme celkem snadno k závěru, že je můžeme rozdělit do pěti relativně samostatných fází. Jsou to: DEFINOVÁNÍ, ANALÝZA, PROGRAMOVÁNÍ, LADĚNÍ a TESTOVÁNÍ a ÚDRŽBA.

Je nesporností předností metody RT, že ji lze ve všech těchto fázích úspěšně použít. Následující příklady se to pokoušejí potvrdit.

Fáze DEFINOVÁNÍ a ANALÝZA

Výhody metody RT v těchto fázích se projevují hlavně v tom, že formulace problému je přesnější, spolehlivější a srozumitelnější než u slovního popisu. Rychlá dekompozice řešené úlohy, možnost prověření algoritmu z hlediska úplnosti reálných variant a jednoznačnosti jejich popisu je pak hlavní výhodou ve fázi analýzy.

Ukažme si to na příkladě jedné z dílčích úloh programového řešení překládače RT PROTAB-25. Její slovní formulací můžeme vyjádřit např. takto:

Součástí překladu RT je též postupná analýza a zpracování textů jednotlivých podmínek (činností) uvedených na pokračovacích štítcích. Základem tohoto zpracování je postupná analýza všech relevantních sloupců pokračovacího štítku s cílem zjistit:

- číslo sloupce v němž text začíná a končí;
- zda se v textu nachází otazník, označující, že jde o rozšířený záznam;
- zda se ve formulaci podmínky (činnosti) nevyskytlo více

otazníků indikujících výskyt rozšířeného záznamu;
- číslo sloupce a štítku v němž se nachází první relevantní otazník, označující rozšířený záznam.

Poznámka: připouští se existence otazníku v literálu.

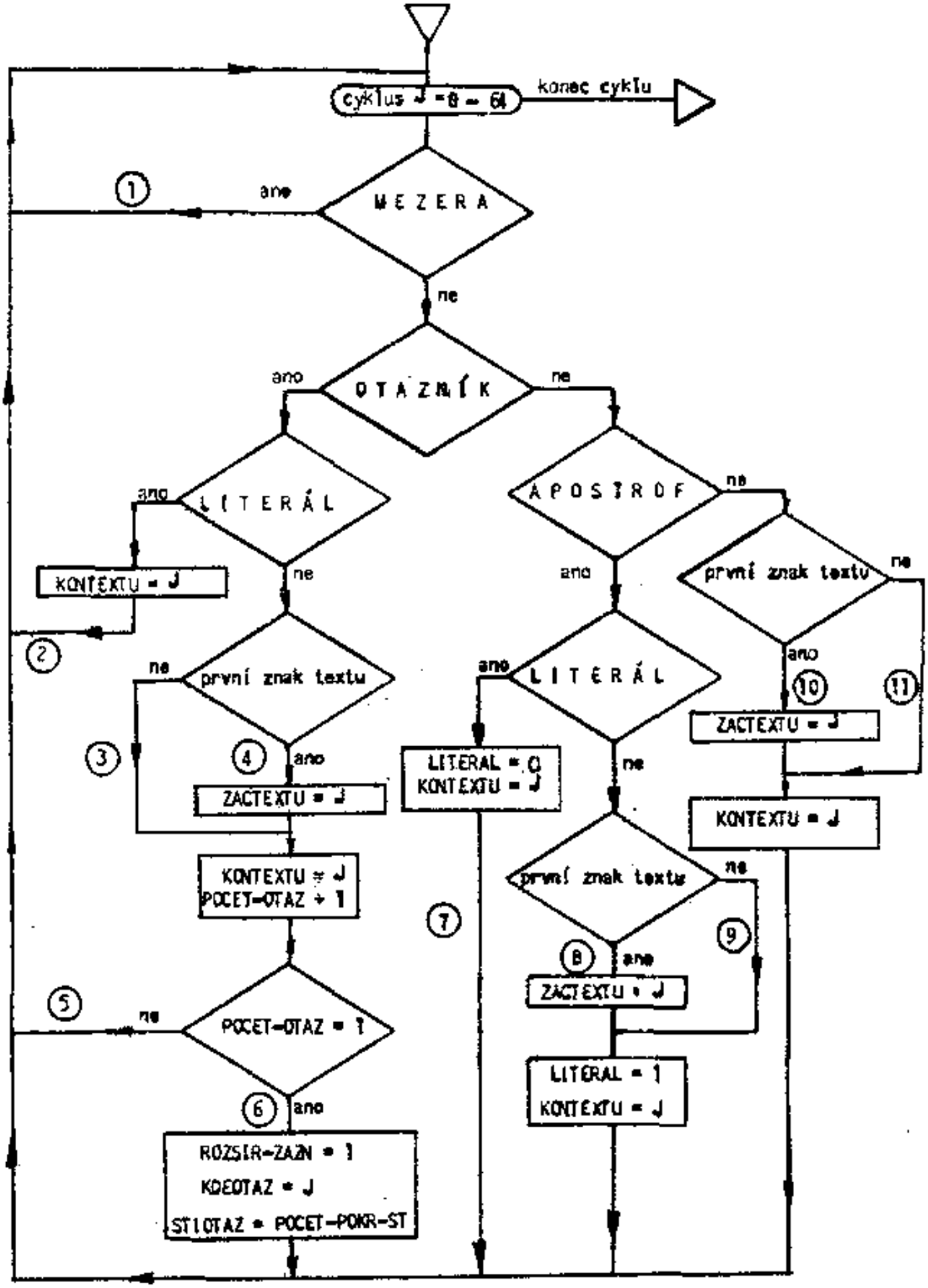
Tento poměrně vágní a ne zcela jednoznačný slovní popis úlohy můžeme popsat pomocí smíšené RT daleko stručněji a přitom zcela jednoznačně (viz obr.1). Kromě jednoduchosti a srozumitelnosti zápisu algoritmu se můžeme navíc jednoduchou kontrolou ubezpečit, že zápis je nerozporný, nenadbytečný a přitom úplný, tzn., že zahrnuje všechny reálné kombinace stavů uvažovaných nezávislých faktorů. To bychom např. o vývojovém diagramu téhož problému již tak jednoznačně říci nemohli (obr.2).

Analýza pokračovacího štítku 1		2	3	4	5	6	7	8	9
Ve zkoumaném sloupci je ?	mezera	otazník			apostrof			ELSE	
Znak je součástí literálu	-	1	0	0	1	0	0	-	-
Jedná se o 1.znak textu	-	-	1	0	-	1	0	1	0
ZAČTEXTU = číslo sloupce			X			X		X	
KONTEXTU = číslo sloupce		X	X	X	X	X	X	X	X
Přičti 1 k POCET-OTAZ			X	X					
Pro POCET-OTAZ = 1 dosaď: ROZSIR-ZAZN = 1 KDEOTAZ = číslo sloupce STIOTAZ = POCET-POKR-ST			X	X					
LITERAL = ?					0	1	1		
Prověř další sloupec	X	X	X	X	X	X	X	X	X

Obr. 1

Fáze PROGRAMOVÁNÍ

V předchozích fázích jsme použili metody RT, nikoliv však překládače RT. Toho můžeme plně využívat až od této fáze. V našem případě to bude překládač RT PROTAB-25 vyvinutý pro operační systém DOS-3/JS.


cyklus J = 0 - 64
konec cyklu
MEZERA
ano
ne
OTAZNÍK
ano
ne
LITERÁL
ano
ne
KONTEXTU = J
2
ne
první znak textu
ne
3
4
ano
ZACTEXTU = J
KONTEXTU = J
POCET-OTAZ + 1
5
ne
POCET-OTAZ = 1
6
ano
ROZSIR-ZAZN = 1
KODOTAZ = J
STIOTAZ = POCET-POKR-ST
7
APOSTROF
ano
ne
LITERÁL
ano
ne
LITERAL = 0
KONTEXTU = J
první znak textu
ne
8
ano
ZACTEXTU = J
9
LITERAL = 1
KONTEXTU = J
první znak textu
ne
ano
10
11
ZACTEXTU = J
KONTEXTU = J

Obr. 2

Pokud bychom tuto úlohu formulovali pomocí vývojového diagramu uvedeného na obr. 2 a chtěli ji naprogramovat v jazyce COBOL, pak by odpovídající zdrojový text tohoto úseku programu mohl vypadat např. takto (viz obr, 2a).

```
PERFORM VYK VARYING J FROM 0 BY 1 UNTIL J > 64 GO TO PA.
VYK. IF MEZERA NEXT SENTENCE ELSE
     IF OTAZNIK
     IF LITERAL
     MOVE J TO KONTEXTU ELSE
     IF ZAVTEXTU = 0
     MOV  J TO ZAVTEXTU, KONTEXTU
     MOV  1 TO POCET-OTAZ ELSE
     MOV  J TO KONTEXTU
     MOV  1 TO POCET-OTAZ
     IF POCET-OTAZ = 1
     MOV  1 TO REZSIK-ZAZN
     MOV  J TO KONECTAZ
     MOVE POCET-POKR-ST TO STICTAZ ELSE
     NEXT SENTENCE ELSE
     IF APOSTROF
     IF LITERAL
     MOVE J TO KONTEXTU
     MOVE 0 TO LITERAL ELSE
     IF ZAVTEXTU = 0
     MOVE J TO ZAVTEXTU
     MOVE J TO KONTEXTU
     MOVE 1 TO LITERAL ELSE
     MOVE J TO KONTEXTU
     MOV  1 TO LITERAL ELSE
     IF ZAVTEXTU = 0
     MOVE J TO ZAVTEXTU
     MOVE J TO KONTEXTU ELSE
     MOVE J TO KONTEXTU.
PA. EXIT.
```

Obr. 2a

Použitím překládače RT PROLAB-25 můžeme tutéž úlohu naprogramovat daleko jednodušeji pomocí RT z obr. 3. Úspornost práce se zápisem algoritmu ve srovnání s klasickým způsobem programování je evidentní. Není však jediným zdrojem zvyšování míry racionalizace v této - klíčové fázi práce programátora.

Překládač RT PROLAB-25 umožňuje racionalizovat programování rovněž tím, že:

a) Dovoluje automaticky optimalizovat vygenerovaný kód, takže se programátor nemusí zabývat otázkou nejvýhodnější posloupnosti jednotlivých příkazů, která by vedla k co nejnižšímu počtu podmínkových testů v programu.

```
      PERFORM CYK VARYING J FROM 0 BY 1 UNTIL J > 64 GO TO PA.
H         CYK                          40
*
C  ?                                |MEZERA|OTAZNIK|APOSTROF| ELSE
C  LITERAL                           -    1  0  0  1  0  0  -  -
C  ZACTEXTU = 0                      -    -  1  0  -  1  0  1  0
*  ==========================================================
A  MOVE J TO ZACTEXTU                     X        X     X
A  MOVE J TO KONTEXTU                  X  X  X  X  X  X  X  X
A  ADD 1 TO POCET-OTAZ                    X  X
A  MOVE ? TO LITERAL                            | 0| 1| 2|
A  IF POCET-OTAZ = 1                      X  X
S  MOVE 1 TO ROZSIR-ZAZN
S  MOVE POCET-POKR-ST TO STICTAZ
S  MOVE J TO KDEOTAZ
A  NEXT SENTENCE                    X
E
PA. EXIT.
```

Obr. 3

b) Umožňuje přímé zpracování RT, které byly konstruovány pro překládač PROTAB, používaný na počítači EC 1021 a to bez jakékoliv úpravy.

c) Umožňuje zpracovávat tzv. <u>dynamické RT</u>, které se vyznačují proložením podmínek činnostmi. Díky tomu můžeme jednou RT zpracovávat např. i procesy, jejichž podmínky se během řešení mění. Příklad dynamické RT je ukázán na obr. 4

```
H          KRIZOVATKA                  45
*
C  ZELENA > MAXDOBA                  1 1 1 1 1 1 1  1 1 1  0
C  AUTODETEKTOR > FRONTA              1 1 0 1 1 1 1  0 0 0  -
*  ----------------------------------------------------------
A  ADD 20 TO MAXDOBA                  X X - X X X X  - - -  -
A  GO AGAIN                                                   X
*  ----------------------------------------------------------
C  VARIANTA = ?                      |AUTA |   TRAMVAJE    | - |
C  TRAMDETEKTOR = 0                  - - - 1 1 0 0  1 0 0  -
C  MAXDOBA - ZELENA > 10              1 0 - 1 0 1 0  - 1 0  -
*  ==========================================================
A  ADD ? TO MAXDOBA                  |-|5| - | 5 |10|-|5|10|-|
A  PERFORM PROJEZDA                   X X   X X X X   X X
A  IF FAZE = 1 ADD 1 TO FAZE ELSE        X         X
S  MOVE 1 TO FAZE
A  GO TO ZMENA-SVETEL                    X         X
A  GO AGAIN                           X X   X X X X   X X
E
```

Obr. 4

d) Dovoluje snadnou a rychlou převoditelnost řešených algoritmů z jednoho programovacího jazyka do jiného, jak vyplývá z následujícího příkladu, uvedeného na obr. 5 a 6.
RT na obr. 5 znázorňuje algoritmus, který odpovídá konvencím pro překlad této RT do jazyka PL/1.

```
H          RT4                                    48
*
C HVEZDICKA = '*'                                 1 0 0 0 -
C HVEZDICKA = 'D' & VAM06.PORPRV = 1              - 1 0 0 -
S VAM06.TYP >= 11 & VAM06.TYP <= 17
C VS.JVS ¬= 'GEP'                                 - 0 - - -
C MEZERA3 = ' '                                   - - 1 0 -
C MEZERA4 = 'R' & VAM06.TYP >= 51 &               - - - 1 -
S VAM06.TYP <= 57
C FLAG = 1                                        - - 0 0 -
* ================================================
A WRITE (G&AM06)  FROM (VSM05)                      X X
A VAM06.KOD = 0                                   X
A VAM06.PORPRV = 0                                X
A WRITE (G&AM06) FROM (VAM06)                     X X X
A VAM06.PORPRV = VAM06.PORPRV + 1                 X X X
A READ FILE (G&SM08) INTO (V1)                    X X X X
A GOTO RT?                                     1     4
E
```

Obr. 5

Přeprogramování tohoto algoritmu do jazyka COBOL můžeme provést pomocí RT např. způsobem, který je ukázán na obr. 6.

```
H          RT4                                    61
*
C HVEZDICKA = '*'                                 1 0 0 0 -
C HVEZDICKA = 'D' AND PORPRV IN VAM06 = 1         - 0 1 0 -
S AND TYP IN VAM06 NOT < 11 AND TYP IN VAM06 NOT > 17
C JVS IN VS NOT = 'GEP'                           - - 0 - -
C MEZERA3 = SPACE                                 - 1 - 0 -
C MEZERA4 = 'R' AND TYP IN VAM06 NOT < 51 AND     - - - 1 -
S TYP IN VAM06 NOT > 57
C FLAG = 1                                        - 0 - 0 -
* ================================================
A WRITE G&AM06 FROM VSM05                         - X - X -
A MOVE 0 TO KOD IN VAM06                          - - X - -
A MOVE 0 TO PORPRV IN VAM06                       - - X - -
A WRITE G&AM06 FROM VAM06                         - X X X -
A ADD 1 TO PORPRV IN VAM06                        - X X X -
A READ G&SM08 INTO V1 AT END GO TO KO8            - X X X X
A GO TO RT?                                    |1|    5    |
E
```

Obr. 6

Fáze LADĚNÍ A TESTOVÁNÍ

Tuto fázi umožňuje PROTAB-25 racionalizovat tím, že přispívá k rychlému nalezení a odstranění vyskytujících se chyb. Předveďme si to opět na příkladě, který je popsán v RT na obr. 3. Při ladění této úlohy se ukázalo, že původní znění 3. podmínky ZACTEXTU = 0 je nesprávné. Na počátku analýzy textu pokračovacího štítku se položka ZACTEXTU nesmí nulovat, ale naopak je nutno ji naplnit, např. maximální možnou hodnotou. Správné znění třetí podmínky tutíž zní:

ZACTEXTU = HIGH-VALUE

V klasickém programu, který je uveden na obr. 2a bychom museli opravit tento podmínkový test celkem třikrát. V RT stačí vyměnit jediný řádek. Výhody tohoto postupu jsou nasnadě a není třeba je dále rozvíjet.

Další racionalizační přínos poskytuje PROTAB-25 při prověřování správnosti programu. Při prověřování správnosti všech větví programu potřebuje programátor znát zpravidla tyto údaje: počet větví programu a zda alespoň jednou každou větví prošel.

a) Znázorníme-li si řešenou úlohu pomocí RT, pak je počet větví dán počtem pravidel této RT. Tak u našeho příkladu z obr.3 je již z letmého pohledu patrné, že řešená úloha má celkem 9 možných cest. Vývojový diagram z obr. 2 má takových cest 11 a zdaleka se to nedovíme tak snadno a rychle jako u RT.

b) Informaci o tom, zda jsme testovanou větví programu skutečně prošli poskytuje překládač PROTAB-25 programátorovi prostřednictvím <u>trasovacího a statistického podprogramu</u>. Použijeme-li trasovací podprogram, pak se při každém průchodu RT vytiskne následující hlášení:

* RT-TRACE * jméno RT * R=nn * Pi/si

kde: nn = číslo pravidla (větve programu), které se uplatnilo
Pi = čísla testovaných podmínek
si = stavy testovaných podmínek

Pro náš příklad zobrazený RT na obr. 3 by takové hlášení mohlo vypadat např. takto:

* RT-TRACE * CYKLUS *R=3*1/2 2/0 3/1

Slovní interpretace tohoto hlášení je:
Při průchodu RT CYKLUS se provedla varianta popsaná ve třetím pravidle. Stav první podmínky byl OTAZNÍK, stav druhé podmínky byl 0 a stav třetí podmínky byl 1.

Při použití statistického podprogramu se po zpracování všech testovacích dat vytiskne tabulka s údaji o počtu průchodů jednotlivými RT. Příklad statistické tabulky, která se vytiskla po testování RT z obr. 3 je ukázán na obr. 7.

```
***** R T  -  S T A T I S T I K A *****
          JMENO  R T :    CYKLUS
          POCET PRUCHODU: 21
*************************************************
* CISLO:    | 1   2   3   4   5   6   7   8   9  *
*=======================================*********
*                                                *
* PRAVIDLO:  3   2   -   1   1   1   0   7   6  *
*                                                *
* PODMINKA: 21   3  15                           *
*                                                *
* CINNOST:   8  18   1   2   1   3               *
*                                                *
**************************************************
```

Obr. 7

Oba podprogramy, které lze použít buď odděleně nebo společně poskytují dostatek informací o tom zda, jak a kolikrát byly jednotlivé větve programu prověřeny.

Fáze DOKUMENTACE

Rovněž v této fázi ovlivňuje překládač PROTAB-25 příznivě racionalizaci programovacích prací. Tím, že opisuje do zdrojového programu RT v původní podobě a současně dovoluje (pokud si to programátor přeje) potlačit tisk vygenerovaného kódu, stává se zdrojový program, jehož procedurální část je vyjádřena soustavou vzájemně propojených RT současně i programovou dokumentací. Uvážíme-li dále, že každý zásah do algoritmu úlohy se realizuje zpravidla prostřednictvím změn v odpovídající RT, je výpis programu ze zdrojové knihovny vždy zaručeně aktuální a platnou dokumentací. To se o dokumentaci založené na vývojových diagramech obvykle říci nedá. Díky RT se tak programy stávají samodokumentovatelnými.

Fáze ÚDRŽBA

Velikou a vynikající vlastností metody RT je to, že umožňuje rozčlenit procedurální části programového řešení úloh do logicky uspořádaných souborů relativně samostatných částí, představovaných RT. Díky tomu získáme přehlednou strukturu dovolující jednak rychlou orientaci v programu, jednak snadnou manipulaci s programem.

Je nasnadě, že program, jehož struktura je tvořena soustavou RT, se dobře udržuje. Tabulková struktura totiž umožňuje, aby se požadované změny řešení prováděly doslova agregátovou metodou - tj. výměnou celých RT nebo doplňováním stávajících.

Závěr

Popsaný racionalizační přínos metody RT umocněný použitím překládače RT PROTAB-25 byl prakticky ověřen při řešení programů nejrůznějšího rozsahu i složitosti řešitelí tohoto překládače. Dobré zkušenosti získali v tomto smyslu též někteří uživatelé počítače EC 1021, využívající k programování překládače PROTAB. Tento překládač sice nedosahuje možností PROTAB-25, nicméně dovoluje využívat všech dobrých vlastností metody RT.

Vzhledem k výrazně dobrým výsledkům, kterých bylo pomocí RT dosaženo, můžeme tuto metodu všem plně doporučit bez jakýchkoliv obav a zábran.