POUŽITÍ STRUKTUR IT-TI V PRAXI

Ing. Pavel Ježek, Státní statek Karviná

Motto:

"Konstruujte jednoduše a přidávejte lehkost !"

1. Úvod

Současný stav vývoje výpočetní techniky a programového vybavení je odborníkům dostatečně znám a není třeba jej rozvádět. V době, kdy vývoj hardware předstihuje vývoj software, kdy tvorba izolovaných programů byla nahrazena realizací velkých programových celků a individuální programování se změnilo v týmovou práci, hledají se nové prostředky a metody, jak tuto práci zefektivnit. Vznikla řada nových přístupů a teoretických prací /Dijkstra, Hoare, Parnas, Wirth, Jackson .../ a vývoj ve světě pokračuje dále. Některé z těchto principů byly vysvětleny v /1/ a /2/. Je jistě vzrušující objevovat a formulovat nové myšlenky a teorie, avšak neméně velkým dobrodružstvím je pokusit se tyto poznatky realizovat v praxi, v konkrétních podmínkách. Jak již řekl Karel Čapek: s novými věcmi je třeba nejprve udělat špatné zkušenosti, aby se s nimi daly udělat zkušenosti dobré.

Použití struktur IT-TI jako dalšího zobecnění struktur dle IT-TI, tak jak bylo nastíněno v /2/, se jevilo jako velmi výhodné, protože slibovalo nalézt jednotný a univerzální přístup k tvorbě řídících programových struktur. Po řadě pokusů o ověření se toto nakonec potvrdilo, a to nejen při tvorbě programů v klasických programovacích jazycích, ale byly nalezeny možnosti i v jiných oblastech. Princip IT-TI je blízký rozhodovacím tabulkám a přitom v sobě zahrnuje tři základní prvky strukturovaného programování - sekvenci, iteraci a selekci. Na základě těchto pokusů jsem se pokusil shrnout a zobecnit v tomto příspěvku.

2. Definice IT-TI a návazné úvahy

Struktura IT-TI je logický součet pravidel uzavřený mezi značky "IT" /=začátek/ a "TI" /=konec/. Každé pravidlo se skládá jednak z podmínkové části /jedna podmínka nebo logický součin více podmínek - označme je "P"/, jednak z výkonné části /jedna nebo několik činností - označme je "C"/. Výkonná část může místo činností obsahovat další vnořenou strukturu IT-TI. Každé pravidlo končí buď návratem na začátek struktury "/" nebo přechodem na její konec "\". Například:

```
IT
      P1 AND P2              --> C1,C2 /
   OR P1 AND NOT P2          --> C1    /
   OR NOT P1 AND P2          --> C2    /
   OR NOT P1 AND NOT P2      --> C3    \
TI
```

P1	1	1	0	0
P2	1	0	1	0
C1	X	X		
C2	X		X	
C3				X
znovu	X	X	X	
konec				X

Uvedený příklad je také znázorněn pomocí rozhodovací tabulky. Po vyhodnocení podmínek se provádějí odpovídající činnosti, a to tak dlouho, až se provede pravidlo končící přechodem na konec. Vnořená IT-TI pak odpovídá situaci "proveď jinou rozhodovací tabulku a vrať se". Pro další výklad označme pravidla končící přechodem na "IT" jako "PART-IT" a pravidla končící přechodem na "TI" jako "PART-TI". Může nastat několik případů:

1.
```
IT
PART-TI
IF P1
 činnosti 1
PART-TI
IF P2
 činnosti 2
TI
```

2.
```
IT
PART-IT
IF P1
 činnosti 1
PART-IT
IF P1
 činnosti 2
TI
```

3.
```
IT
 činnosti 0
PART-TI
IF P1
 činnosti 1
PART-IT
IF P2
 činnosti 2
TI
```

4.
```
IT
PART-TI
IF P1
 činnosti 1
PART-IT
IF P2
 činnosti 2
PART-IT
 činnosti 3
TI
```

1. Končí-li všechna pravidla přechodem na konec, nastane pouze jeden průchod strukturou - je to selekce.
2. Končí-li všechna pravidla návratem na začátek, nemá struktura východ - je to nekonečný cyklus.
3. Činnosti 0 se provádějí vždy před vyhodnocením podmínek - je to obdoba úvodního pravidla - viz /4/.
4. Za poslední "PART" nenásleduje podmínka - je to obdoba pravidla "JINAK" - viz /4/.

Z uvedených charakteristik vyplývá, že IT-TI jsou vhodné jako řídící programové struktury. Pokud bychom je chtěli realizovat pomocí současných programovacích prostředků, je nejlépe zvolit formu předprocesoru /předkompilátoru/. Tímto způsobem lze hostitelský jazyk doplnit o potřebné pseudopříkazy. Předprocesor je musí umět rozpoznat a vygenerovat z nich takové příkazy hostitelského jazyka, které realizují požadované funkce. Zdrojový text se tedy bude skládat z pseudopříkazů a z běžných příkazů. Pro zvýšení přehlednosti textu /aby struktury nemusely obsahovat dlouhé sekvence příkazů/ je také vhodné vytvořit obdobu příkazu "PERFORM", tj. přechod na podřízenou programovou část, její provedení a návrat zpět. Hostitelský jazyk pak musí dovolit vyhodnocení logických podmínek a možnost odkázat se na stanovené body v programu /návěstí/. Jelikož vyhodnocování logických podmínek bude postupné, pak abychom zabránili zacyklení struktury, je nutné nejprve uvádět všechna pravidla "PART-TI" a pak teprve všechna pravidla "PART-IT".

Příklad ruční interpretace struktur IT-TI v jazyce PL/I byl uveden v /2/. Dalším vhodným jazykem pro realizaci předprocesoru je FORTRAN, který normálně nepodporuje strukturované programování a k řízení používá skokové příkazy, což se je vyloučeno. Přesto je stále hojně používán. Doplněním řídících struktur se z něho může opět stát moderní jazyk. Princip IT-TI je možno uplatnit i v jiných programovacích jazycích. Lze jej však, tam kde je to možné, použít i pro řízení prací /job control/ nebo pro jiné účely.

Několik ukázek praktického použití IT-TI na příkladě stávajících softwarových prostředků je uvedeno v části 3. Dotažení této metody do důsledků pro účely programování v programovém systému vyžádalo vytvořit zcela nové programové vybavení nového, multitaskingově vyššího typu.

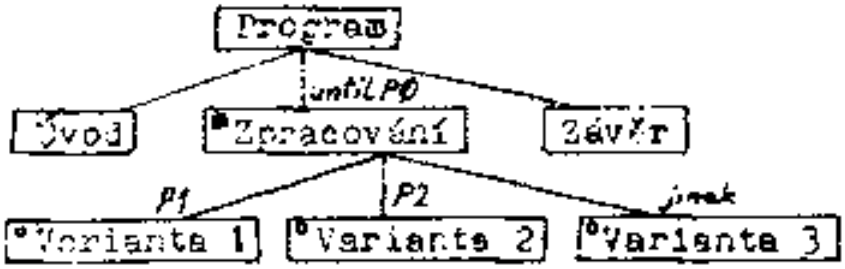
3. Zpracování s různými metodami

Porovnejme některé základní metody strukturovaného programování na následujícím příkladu. Jde realizovat program, zajišťující tyto funkce:

- nejprve se provedou úvodní činnosti /úvodní dialog, zadání parametrů, otevření souborů, čtení vzorků/;

- pak se bude cyklicky provádět vlastní zpracování, a to až do splnění koncové podmínky "PO" /např. konec dat/; toto zpracování bude mít 3 varianty: při splnění podmínky "P1" se provede varianta 1, při "P2" varianta 2, jinak se provede varianta 3;
- nakonec se uskuteční závěrečné činnosti /uzavření souborů, hlášení o ukončení programu apod./.

Tento algoritmus lze znázornit pomocí strukturního diagramu:



K popisu algoritmů nepoužijeme žádný konkrétní programovací jazyk, pouze pseudopříkazy.

a/ Metoda shora-dolů:

Problém byl postupně rozkládán na jednodušší části na jednotlivých hierarchických úrovních.

```
/* HLAVNI CAST */
   PERFORM UVOD.
   PERFORM ZPRACOVANI UNTIL PO.
   PERFORM ZAVER.

/* UROVEN 1 */                          /* UROVEN 2 */
   UVOD. ...                               VARIANTA1. ...
   ZPRACOVANI.                             VARIANTA2. ...
     IF P1       PERFORM VARIANTA1         VARIANTA3. ...
     ELSE IF P2  PERFORM VARIANTA2
     ELSE        PERFORM VARIANTA3.
   ZAVER. ...
```

b/ Jacksonova metoda:

Vede ke vnořování iterací a selekcí. Sekvence příkazů lze psát buď přímo do struktury nebo se na ně vytvoří odkazy.

```
/* HLAVNI CAST */
   PERFORM UVOD.
   DO-UNTIL PO
     SELECT IF P1                      /* PODRIZENE CASTI */
       PERFORM VARIANTA1                  UVOD. ...
     OR IF P2                             VARIANTA1. ...
       PERFORM VARIANTA2                  VARIANTA2. ...
     OR                                   VARIANTA3. ...
       PERFORM VARIANTA3                  ZAVER.
     END
   END-DO
   PERFORM ZAVER.
```

c/ Struktura IT-TI:

Příkazy lze opět začlenit přímo do struktur nebo se vytvoří odkazy. Složitější problémy vedou k soustavám rozhodovacích tabulek a k vnořeným IT-TI.

```
/* RIDICI CAST */
   PERFORM UVOD.
   IT
   PART-TI IF P0
   PART-IT
      IT
      PART-TI IF P1
         PERFORM VARIANTA1
      PART-TI IF P2
         PERFORM VARIANTA2
      PART-TI
         PERFORM VARIANTA3
      TI
   TI
   PERFORM ZAVER.
/* PODRIZENE CASTI */
   UVOD. ...
   VARIANTA1. ...
   VARIANTA2. ...
   VARIANTA3. ...
   ZAVER. ...
```

RT0:

P0	0	1
proveď RT1	X	
znovu	X	
konec		X

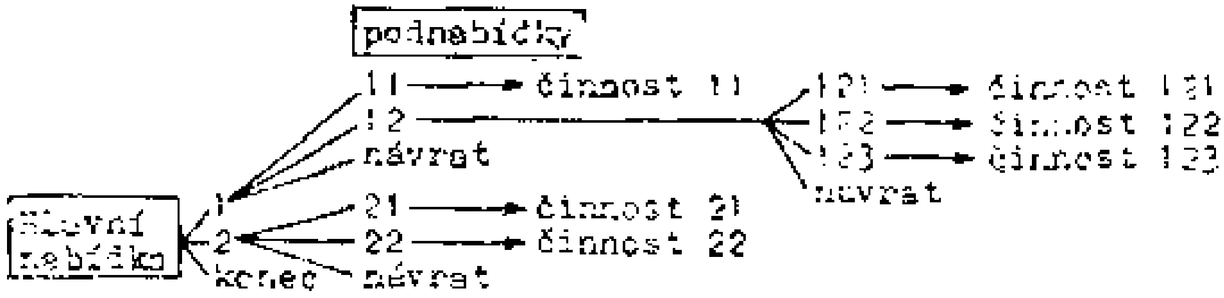
RT1:

P1 P2	1 0	0 1	ji- nak
VARIANTA1	X		
VARIANTA2		X	
VARIANTA3			X
návrat do RT0	X	X	X

Uvedené příklady ukazují, že metoda IT-TI je rovnocenná jiným metodám strukturovaného programování. Pokud máme tvořit složité programové systémy, nevyhneme se zavádění typizace a unifikace v projektování a programování. Použití IT-TI se může stát jednou z cest vedoucích k tomuto cíli. Kromě toho se ukazuje možnost tyto metody navzájem kombinovat.

4. IT-TI a interaktivní programy

V /3/ byly shrnuty zkušenosti, týkající se interaktivní komunikace s počítačem. Kromě mnoha podnětných zásad byl také vysvětlen princip průchodu stromem nabídek. Mějme např. strom:



Průchod tímto stromem může být buď nezkrácený /uživatel zadá např. volby 1-12-122/ nebo zkrácený /přímo volba 122/. Ukážeme nyní, jak tento problém řešit pomocí IT-TI.

a/ Nezkrácený průchod:

```
PERFORM úvod
IT
  PERFORM hlavní nabídka
PART-TI
IF (koncové varianta)
  PERFORM závěr
PART-IT
IF (varianta 1)
  PERFORM činnost 1   /*/
...
PART-IT
IF (varianta n)
  PERFORM činnost n   /*/
PART-IT
  neexist.volba
TI
```

Při víceúrovňové nabídce /strom nabídek/ se na místo provedení i-té činnosti /*/ vloží další vnořená struktura IT-TI, která má stejný tvar, jen místo "hlavní nabídky" je "podnabídka".
Poslední PART-IT zajišťuje případ neexistující volby, pokud tento není ošetřen přímo při zobrazení nabídky a čtení volby.
"Úvod" se provádí na začátku programu /přípravné činnosti/, "závěr" na konci programu - ve vnořených IT-TI může být nahrazen koncovým hlášením dané podnabídky.

b/ Zkrácený průchod:

```
PERFORM úvod
IT
  PERFORM zpracování voleb
PART-TI
IF (koncová volba)
  PERFORM závěr
PART-IT
  SWITCH var
  (seznam výkonných bloků)
TI
```

"Úvod" a "závěr" mají stejnou funkci jako výše. "Zpracování voleb" zajišťuje zobrazení stromu nabídek, přečtení volby, její identifikaci a přiřazení odpovídající hodnoty proměnné "var". Přepínač SWITCH pak dle hodnoty "var" zajistí přechod na konkrétní blok příkazů, provádějících zvolenou činnost, nebo na podnabídku.

5. Příklady použití IT-TI

a/ FORTRAN:

Byl vytvořen předprocesor, který rozšiřuje FORTRAN-IV nebo FORTRAN-77 o příkazy řídících programových struktur. Příklad vstupního zdrojového textu:

```
      WRITE (UC,30000)              /zahajovací hlášení/
C
      @IT
      @PART-TI
      IF (INDION)
         WRITE (UC,40000)           /koncové hlášení/
      @PART-IT
      IF (EOF.OR.ERR)
         CALL ZADANI                /zadání tisku/
      @PART-IT
         CALL TISK                  /vlastní tisk/
      @TI
C
      STOP
```

```
      WRITE (UC,30000)
C
      GO TO 99999
99999 CONTINUE
      IF (INDKON) GO TO 99997
      GO TO 99996
99997 CONTINUE
         WRITE (UC,40000)
      GO TO 99998
99996 CONTINUE
      IF(EOF.OR.ERR)GO TO 99995
      GO TO 99994
99995 CONTINUE
         CALL ZADANI
      GO TO 99999
99994 CONTINUE
         CALL TISK
      GO TO 99999
99998 CONTINUE
C
      STOP
```

Zde je vygenerovaný výstup. Příklad ukazuje hlavní část programu, který postupně tiskne zvolené soubory až potud, kdy zadáme ukončení. V podprogramu "ZADANI" se určí způsob tisku a vstupní soubor, jenž se otevře a provede se jeho první čtení, nebo se zadá koncové volba /"INDKON"/. Podprogram "TISK" provádí výstup dle zvoleného způsobu tisku. Po výstupu věty se provádí další čtení souboru /metoda čtení vpřed/, které zároveň nastavuje příznaky konce souboru "EOF" a chyby při čtení "ERR". Počáteční nastavení logických proměnných je: INDKON,ERR =.FALSE., EOF=.TRUE.

b/ COBOL:

```
 PROCEDURE DIVISION.
*
 ZAHAJENI.
      OPEN INPUT VSTUP-1 VSTUP-2
           OUTPUT VYSTUP.
      PERFORM CTI-1.
      PERFORM CTI-2.
*
*     IT
 ZPRACOVANI.
*     PART-TI
      IF EOF-1 AND EOF-2
         GO TO END-ZPRACOVANI.
*     PART-IT
      IF NOT EOF-1 AND NOT EOF-2
         PERFORM MERGE THRU END-MERGE
         GO TO ZPRACOVANI.
*     PART-IT
      IF EOF-1 AND NOT EOF-2
         WRITE VETA-2-O FROM VETA-2-I
         PERFORM CTI-2
         GO TO ZPRACOVANI.
*     PART-IT
      IF NOT EOF-1 AND EOF-2
         WRITE VETA-1-O FROM VETA-1-I
         PERFORM CTI-1
         GO TO ZPRACOVANI.
*     TI
 END-ZPRACOVANI.
      EXIT.
*
 ZAVER.
      CLOSE VSTUP-1 VSTUP-2 VYSTUP.
      STOP RUN.
```

Pro tento jazyk prakticky předprocesoru není třeba. Jediným problémem je, že jednotlivé IT-TI se ručně nedají vkládat do sebe, aniž by se musel vymýšlet složitý systém návěští. Proto je lépe vytvořit a vyvolat vnořenou IT-TI jako samostatný PERFORM.

Tento příklad ukazuje hlavní část programu pro zatřídění dvou vstupních souborů do jednoho výstupního. "MERGE" je vnořená struktura IT-TI, v níž se porovnávají klíče vstupních vět, provádějí se s nimi operace, zápis a další čtení. Má tři části PART-TI, které testují

KLIC-1 > KLIC-2
KLIC-1 = KLIC-2
KLIC-1 < KLIC-2

e/ IT-procesor:

Tento příklad ukazuje použití IT-TI v oblasti řízení prací: program pro IT-procesor, jenž zpracovává nepřímé povelové soubory na počítačích SMEP.

```
.; --- HLAVNI CAST ----------       .; --- PODRIZENE CASTI -----
.;                                  .MENU:  CLR
        .ENABLE SUBSTITUTION                ; ******************
.;                                          ; 1=proveď činnost 1
.IT:    .GOSUB MENU                         ; 2=proveď činnost 2
        .GOSUB VOLBA                        ; 0=konec zpracování
.;PART-TI                                   ; ******************
.V0:    .IF V NE "0" .GOTO V1               .RETURN
        .GOSUB VOL0                 .;
        .GOTO TI                    .VOLBA: .ASKS[1:1:"0"]V VOLBA
.;PART-IT                                   .RETURN
.V1:    .IF V NE "1" .GOTO V2       .;
        .GOSUB VOL1                 .VOL0:  ; *** KONEC PRACE ***
        .GOTO IT                            .RETURN
.;PART-IT                           .;
.V2:    .IF V NE "2" .GOTO VX       .VOL1:  příkazy pro činnost 1
        .GOSUB VOL2                         .RETURN
        .GOTO IT                    .;
.;PART-IT                           .VOL2:  příkazy pro činnost 2
.VX:    .GOTO IT                            .RETURN
.;
.TI:    .STOP
```

f/ FAND:

Relační databázový procesor FAND - viz /5/ - je velmi účinným prostředkem na mikropočítačích. Kromě jiných obsahuje i příkaz "MENULOOP", který má hodně společného s IT-TI. Tento příkaz zobrazí na obrazovce nabídku, po volbě některého jejího bodu zajistí provedení odpovídajících činností a návrat zpět. Nabídky mohou být hierarchicky členěny. Oproti IT-TI je zde místo vyhodnocení logických podmínek text, který je volen kurzorem. Uvedeme zde jednak syntax příkazu a jednak příklad použití:

```
%menuloop ("NadpisNabídky"
            'TextVolby1' : příkazy procedur 1 %,
            'TextVolby2' : příkazy procedur 2 %,
             ...
            'TextVolbyN': příkazy procedur N %)

%menuloop ("Zpracování evidence ..."
            "----------------------"
            " "
            'Aktualizace základního souboru': P AKTUAL  %,
            'Výstupní informace            ': P VYSTUP  %,
            'Údržba číselníků              ': P UDRCIS  %,
            'Rekonstrukce po havárii       ': P REKONS  %,
            'K O N E C  zpracování úlohy   ': P KONEC
                                              %exit     %)
```

6. Závěr

Pokusil jsem se v tomto příspěvku objasnit některé nové pohledy na tvorbu programového vybavení. Máme-li shrnout vlastnosti struktur IT-TI, pak musíme konstatovat, že:

- při uplatňování metody ve stávající praxi se lze setkat s odporem konzervativně smýšlejících pracovníků, vyplývajícím z nechuti učit se něco nového;
- metoda je vhodná zejména pro řídící programové struktury a je dosti pohodlná /stanoví se podmínky, přiřadí se činnosti/;
- podporuje unifikaci v programování a zvyšuje kulturu projevu programátorů;
- nutí nezapomínat na některé z možností /což vyplývá z povahy rozhodovacích tabulek/;
- velmi usnadňuje ladění, protože statický zápis se mnohem snadněji ověřuje než zápis dynamický /GO TO/;
- značně přispívá k čitelnosti a udržovatelnosti programových děl;
- není všelékem, ale může se stát dobrým pomocníkem při projektování a programování;
- použití principu IT-TI v základním programovém vybavení počítačů slibuje značné přínosy.

Jestliže hovoříme o existující "softwarové krizi", pak je třeba, aby se hledaly cesty, jak jí čelit. Neboť i to nejnepatrnější tvoření má větší cenu než mluvení o věcech již vytvořených.

Literatura:

/1/ Honzík,J.: Dokazování programu,
Sborník Programování '83, DT ČSVTS Ostrava, 1983

/2/ Schnapp,I.: Použitie štruktúry IT-TI pri návrhu programov,
Sborník Programování '84, DT ČSVTS Ostrava, 1984

/3/ Bébr,R.: Recept na jídelníček aneb metodika dialogu formou menu, Sborník Programování '84, DT ČSVTS Ostrava, 1984

/4/ Chvalovský,V.: Rozhodovací tabulky, SNTL Praha, 1984

/5/ Tichá,B.: FAND - relační databázový prostředek,
MAA č. 11/1988, str. 416-418