T U R B O P a s c a l

RNDr. Ing. Ivan Lexa, CSc. - Školní statek 525 v Roškově

Úvod

Přesto že se jazyk Pascal od svého zrodu nepřetržitě a dopěšně šířil, o jeho skutečně masovém nástupu, zejména na stolních počítačích, mohly být ještě začátkem osmdesátých let pochybnosti. Převrat přinesla až firma Borland (do té doby skoro neznámá) svou implementací Pascalu nazvanou TURBO Pascal. Brzy se ukázalo, že přívlastek "TURBO" (značící v přeneseném významu vysokou výkonnost) si jejich výrobek plně zaslouží.

Firma neusnula na vavřínech. Po první verzi následovaly postupně další čtyři verze, ve kterých byl TURBO Pascal dále rozvíjen podle zkušeností rychle se zvyšujícího počtu uživatelů. Jeho dopěšnost je dnes tak výrazná, že ji lze vysvětlit jedině tím, že dobře vystihl skutečné potřeby programátorů. Odborníci většinou očekávají, že TURBO Pascal bude mít zpětně vliv na standard Pascalu (z kterého původně vycházel), pokud se dokonce nestane sám standardem.

Všimněme si dále hlavních rysů jednotlivých verzí TURBO Pascalu tak, jak po sobě následovaly.

Verze 1

Standard jazyka Pascal (daný normou na drovni 0) je v této verzi respektován s následujícími omezeními :

- nejsou povoleny procedurální a funkcionální parametry podprogramů,
- nejsou předdeklarovány procedury PUT a GET a není povolen přístup k souborům pomocí přístupové proměnné (původní možnosti těchto prostředků jsou však realizovány jinak),
- dynamické proměnné nelze rušit jednotlivě (není předdeklarována procedura DISPOSE), ale pouze skupinově pomocí

nově zavedené procedury RELEASE (všechny nové proměnné od určité počínaje),
- předdeklarované proceduře NEW nelze zadávat varianty variantních záznamů (podobný efekt lze docílit nově zavedenou procedurou GETMEM),
- klíčové slovo PACKED nemá vliv na uložení dat, nejsou předdeklarovány standardní procedury PACK a UNPACK (zhušťování se provádí automaticky všude, kde je to efektivní a možné),
- není implementována procedura PAGE,
- příkazem GOTO nelze skočit do nadřazeného bloku.

Rozšíření, která přináší verze 1 oproti standardu, je podstatně více. Nejdůležitější z nich jsou :
- datový typ "string" (dynamický znakový řetěz) a možností odvolávat se na jednotlivé znaky pomocí indexu (viz příklad 1),
- řízení překladu direktivami, zařazovanými do zdrojového textu,
- vkládání úseků zdrojového textu z diskových souborů během překladu (INCLUDE),
- volné řazení a několikanásobný výskyt úseků v deklarační části bloku (spolu s INCLUDE umožňující pronikavé zvýšení přehlednosti programů),
- možnost přímého (nesekvenčního) přístupu k souborům,
- programovatelné ošetření chyb vstupu a výstupu,
- předávání řízení překrývným Pascalským programům s možností sdílení proměnných (EXECUTE, CHAIN),
- rozšíření služeb příkazu CASE (možnost použít intervalů hodnot ve výběrech, možnost zařadit alternativu ELSE - viz příklad 1),
- strukturované konstanty (konstanty strukturovaných typů),
- přímý přístup k celé paměti a k datovým portům,
- manipulace s bity a byty v proměnných, možnost vkládat příkazy "ve strojovém kódu" (INLINE),
- možnost sdílení stejného místa v paměti několika proměnnými,
- několik desítek nových užitečných předdeklarovaných procedur a funkcí.

Vedle rozšíření jazyka je neméně významným přínosem TURBO Pascalu tzv. integrované prostředí, obvyklé do té doby snad jen u interpretů jazyka BASIC. Skloubení překladače, kvalitního editoru, ladicích a služebních prostředků do jednoho celku se značným komfortem a zautomatizováním služeb poskytlo fascinující zvýšení produktivity programátorské práce. Tak například překladač po zjištění syntaktické chyby nejen hlásí příčinu vysvětlujícím textem, ale také automaticky předá řízení editoru, který zobrazí příslušný úsek zdrojového textu a kurzorem označí pozici chyby.

Verze 2

Proti verzi 1 došlo pouze k těmto třem rozšířením :
- pro práci s dynamickými proměnnými byly doplněny předdeklarované podprogramy (DISPOSE, FREEMEM, MAXAVAIL), umožňující jednotlivé rušení dynamických proměnných,
- možnost segmentace programů na úrovni podprogramů (OVERLAY PROCEDURE, OVERLAY FUNCTION),
- doplnění předdeklarované funkce UPCASE (převod znaku z malé abecedy na velkou).

Verze 3

Tato verze je poslední, která ještě existuje i pro počítače s 8-bitovým procesorem (Z80). Zde došlo opět jen ke třem menším rozšířením :
- doplnění předdeklarované procedury EXIT (opuštění běžného bloku),
- užitečné rozšíření možností příkazu INLINE,
- možnost zpracovat parametry programu, se kterými byl spuštěn z monitoru operačního systému.

Zato však verze 3 pro počítače s 16-bitovým procesorem získala kromě toho ještě předdeklarované funkce WHEREX, WHEREY pro sejmutí okamžité polohy kurzoru na displeji a bohatou kolekci předdeklarovaných podprogramů pro práci s barevnou grafikou.

Verze 4

Tato verze existuje již jen pro počítače s 16-bitovým procesorem a proti verzi 3 znamená podstatný kvalitativní skok. Integrované prostředí je zde vyřešeno moderním způsobem s plným využitím barev, okének a hierarchicky vnořovaných nabídek. Komfort práce výrazně vzrostl. Systém "helpů" obsahuje několik set stránek dokumentace a umožňuje zobrazit na displeji rychle vždy to, co nás v dané situaci může zajímat. Verze 4 přináší tato zásadní rozšíření jazyka :

- možnost používat samostatně zkompilované moduly (units), které se pak sestavují do cílového kódu v době překladu (moduly mohou být vytvořeny kromě Pascalu také v assembleru),
- možnost vnořovaného vkládání desek zdrojového textu (INCLUDE) až do hloubky 8,
- použití direktiv pro podmíněný překlad ve zdrojovém textu,
- rozšíření jednoduchých datových typů o další formáty celých a reálných čísel s různou přesností (SHORTINT, WORD, LONGINT, SINGLE, DOUBLE, EXTENDED),
- několik desítek nových standardních podprogramů, které jsou však již seskupeny (spolu s většinou dosavadních) do standardních modulů (SYSTEM, DOS, CRT, GRAPH, PRINTER).

Naproti tomu ve verzi 4 již nejsou prostředky pro segmentaci programů (CHAIN, OVERLAY).

Verze 5

Tato verze je z roku 1988 a proti verzi 4 přináší následující tři rozšíření :

- nový základní datový typ "podprogram", který umožňuje nejen používat procedurálních a funkcionálních parametrů v procedurách a funkcích, ale používat i proměnných nebo složek proměnných typu "podprogram", přiřazovat jim hodnoty, ... (viz příklad 2),
- možnost segmentace programu na úrovni celých modulů,

- v integrovaném prostředí je navíc snímací ladicí prostředek, umožňující na úrovni zdrojového textu krokovat program, sledovat hodnoty zvolených proměnných či výrazů, měnit hodnoty proměnných, volit body zastavení, atd.

Dvě obecnější úvahy

1. Znám člověka, který když dostal svou první výplatu, rozplýval se, jak je to moc peněz. Veden tímto svým dojmem dokázal vše utratit během deseti dnů a do konce měsíce si pak vypůjčoval od spolupracovníků na obědy. Něco podobného se s autory (a pravděpodobně i uživateli) TURBO Pascalu opakovalo dokonce dvakrát.

   U verze 1 byli autoři ohledně kapacity operační paměti zřejmě optimisty a tudíž možnost šetření paměti překrýváním celých programů (CHAIN, EXECUTE) považovali za dostatečnou. "Tvrdá realita" je však brzy přesvědčila o opaku a tak již verze 2 dovoluje segmentovat na úrovni podprogramů (OVERLAY).

   V příručce k verzi 4 si můžete přečíst (volně přeloženo) : "Ve verzi 3 vytvářel kompilátor soubor COM a program proto nemohl být větší než 64Kb. Aby jste s tím vystačili, museli jste používat CHAIN a OVERLAY. Ve verzi 4 je velikost vašeho programu omezena pouze operačním systémem, takže překrývání pomocí CHAIN a OVERLAY je zbytečné a nebylo proto implementováno". Ale po nějaké době se ukázalo, že ani 640 Kb operační paměti není nevyčerpatelných, a tak ve verzi 5 již máme opět možnost používat překryvů, tentokrát na úrovni celých modulů.

2. Každý jazyk je ve své pozdější podobě jakýmsi kompromisem mezi potřebami jeho uživatelů (tedy programátorů) na straně jedné a pracností vývoje kompilátoru na straně druhé. Na příkladu TURBO Pascalu můžeme dokumentovat, že někdy se vhodný kompromis nalezne až experimentováním.

   Tak například ve verzi 1 chybí oproti standardu Pascalu možnost jednotlivého rušení dynamických proměnných (DISPOSE) a autoři nás dokonce v manuálu přesvědčují, že je to tak lepší (s ohledem na rychlost a zabra-

nou operační paměť). Ve verzi 2 je však tato možnost již implementována, zřejmě pod tlakem připomínek uživatelů.

Druhým příkladem by mohly být procedurální a funkcionální parametry. Standard Pascalu tuto možnost má, verze 1 až 4 TURBO Pascalu ji postrádali, ale verze 5 ji opět zavádí. Dá se tedy usuzovat, že je to věc přece jen potřebná.

Konečně třetím příkladem je vztah TURBO Pascalu k assembleru. Verze 1 až 3 kombinování obou překladačů nepodporovaly. Vycházelo se z toho, že TURBO Pascal je tak "mohutný" jazyk, že si vystačí většinou sám a těch pár výjimek se zvládne ve strojovém kódu příkazů INLINE. Praxe však dokázala opak a tak ve verzích 4 a 5 je již možnné začlenit a volat podprogramy, vytvořené v assembleru.

Co přinesl TURBO Pascal

Pokusme se na závěr shrnout hlavní zkušenosti, které přineslo "hnutí TURBO Pascal" jednak samotnému Pascalu, jednak programování vůbec :

1. Dynamické znakové řetězy (string) jsou tak přirozeným a užitečným datovým typem, že jejich absenci ve standardu Pascalu lze považovat za omyl.
2. Předepsané pořadí úseků deklarací a jejich neopakovatelnost v bloku měly své opodstatnění asi jen u "školních" programů malého rozsahu. U složitých programových systémů se stává překážkou přehledné dekompozice.
3. Konstanty strukturovaných typů se jeví jako velmi přirozené rozšíření standardu Pascalu a zejména pro přehlednou konstrukci tabulek jsou těžko postradatelné.
4. Zavedení datového typu "podprogram" je v Pascalu skutečně pozoruhodným experimentem.
5. Kvality "integrovaného prostředí" jazyka mají nemenší důležitost než kvality samotného jazyka.

Použitá literatura

Jinoch,Müller,Vogel: Programování v jazyku PASCAL
/SNTL 1988/
Manuály Turbo Pascal verzí 1.0 až 5.0
/Borland International, California 1983 až 1988/

Příklad 1

```
type Klad  = 1..20 ;
     Nezap = 0..30 ;
procedure Konstatuj ( Potreba : Klad ; Je : Nezap ) ;
     var T : string[25] ;
     begin
        T:='␣žáků,␣' ;
        case Potreba of
           1 :    T[5]:='a' ;
           2..4 : T[5]:='y'
        end ;
        if Potreba<>Je then T:=T+'ale␣' ;
        case Je of
           0 :    T:=T+'není' ;
           1 :    T:=T+'je' ;
           2..4 : T:=T+'jsou' ;
           else   T:=T+'je␣jich'
        end ;
        T:=T+'␣tam␣' ;
        write('Potřebujeme␣',Potreba,T) ;
        if Je=0 then write('žádný')
                else write(Je)
     end ;
{příklad demonstruje některé možnosti dynamických znakových
řetězců a příkazu CASE v TURBO Pascalu}
```

Příklad 2

```
const PocetFunkci = 50 ;
      MaxBodu     = 30 ;
type  CisloFunkce = 1..PocetFunkci ;
      CisloBodu   = 1..MaxBodu ;
var   F : array [CisloFunkce] of function (X:real) : real ;
      Nf: CisloFunkce ;
procedure Sejmi (var X,Y:real) ;
      begin ... end ;
function F1 (X:real) : real ;
      begin ... end ;
function F2 (X:real) : real ;
      begin ... end ;
      :
function F50 (X:real) : real ;
      begin ... end ;
function Nejlepsi (N:CisloBodu) : CisloFunkce ;
      var I : CisloBodu ;
          Cf: CisloFunkce ;
          X,Y : array[CisloBodu] of real ;
          Sc,Min : real ;
      begin
        for I:=1 to N do Sejmi(X[I],Y[I]) ;
        Min:=1E30 ;
        for Cf:=1 to PocetFunkci do
          begin
            Sc:=0 ;
            for I:=1 to N do Sc:=Sc+sqr(F[Cf](X[I])-Y[I]) ;
            if Sc<Min then begin Nejlepsi:=Cf ; Min:=Sc end
          end
      end ;
begin
  F[1]:=F1 ; F[2]:=F2 ; ... ; F[50]:=F50 ;
  ...
```

```
  ...
  ...
  Nf:=Nejlepsi(17);   {sejme se 17 bodů a z funkcí F1 až F50
                       se vybere ta, jež nejlépe aproximuje
  ...                  sejmutou křivku. Pořadové číslo takto
  ...                  vybrané funkce se uloží do Nf.}
  ...
end .
```

Příklad ukazuje možnost použití datového typu "podprogram" v TURBO Pascalu verze 5.