Deklarační část programu

Pascal řadíme mezi vyšší programovací jazyky. Program napsaný v tomto jazyku se nejdříve přeloží pomocí kompilátoru do jazyka, kterému rozumí procesor počítače, strojového kódu. Musíme si uvědomit, že kompilátor je pouze program, po kterém nemůžeme chtít, aby nám přeložil neúplný, nepřesný či chybný zdrojový kód programu. Proto je nezbytně nutné znát a dodržovat syntaxi a sémantiku jazyka. Syntaxe je soubor pravidel, popisujících přípustné tvary dílčích konstrukcí, příkazů a celého programu (např. když zapomenete na konci řádku napsat středník - je to syntaktická chyba). Delphi nerozlišuje malá a velká písmena, jednotlivé příkazy musí být od sebe odděleny středníkem. Sémantikou je určen význam jednotlivých konstrukcí. Zatímco syntaktické chyby nám kompilátor objeví, sémantické chyby odhalit nemůže.

Symboly

Symboly jsou nejmenší významové jednotky v programu jazyka Object Pascal. Rozeznáváme zvláštní symboly, klíčová slova, identifikátory, návěští, číslice a řetězcové konstanty. Program v Object Pascalu je tvořen symboly a oddělovači, jako je mezera nebo komentář. Dva přilehlé symboly musí být odděleny jedním či více separátory.

Speciální symboly

Používá následující množinu ASCII znaků:

- Písmena anglické abecedy , A až Z případně a až z
- Číslice arabské 0 .. 9
- Mezery a všechny řídící znaky
- Speciální symboly : +-*/=><[].,():;@{}#
- Také tyto párové znaky jsou speciální symboly

  <= >= := .. (* *) (. .) [ ] { }

Klíčová slova a standardní direktivy

and	**array**	**file**	**mod**	**as**	**string**	
not	**then**	**begin**	**for**	**case**	**function**	
off	**to**	**goto**	**or**	**try**	**const**	**if**
type	**unit**	**program**	**until**	**div**		**uses**
do	**var**	**downto**		**rekord**	**while**	
else	**interface**	**repeat**	**with**	**end**	**set**	**xor**

Kromě klíčových slov jazyka Object Pascal lze použít i direktivy – nebude nyní probíráno. Klíčová slova doporučujeme psát malými písmeny.

Proměnné a identifikátory, deklarace proměnných

Proměnné mají bezesporu největší význam při programování. Během nějakého výpočtu jsou v nich uloženy zpracované hodnoty. V programu používáme libovolný

počet proměnných. Vzájemně je pak rozlišíme - identifikujeme jmény. Jména proměnných se nazývají **identifikátory.** Identifikátor ukazuje na paměťové místo, ve kterém je uložena hodnota, která se muže během výpočtu měnit. Na obr. je znázorněná proměnná, která se jmenuje Cislo a má hodnotu 5.

Identifikátor	paměť RAM
Cislo	5

V programu musíme pojmenovat nejen všechny proměnné, ale i konstanty, procedury, funkce, unity, programy, datové typy atd. - musíme tedy pro ně pojmenovat identifikátory.

Identifikátory:
Stavebními kameny celého programu jsou proměnné procedury a funkce. Až je začneme používat, bude nutné jim dát nějaké jméno. Tomuto jménu se říká identifikátor. Identifikátorem může být libovolné slovo, které začíná písmenem z anglické abecedy, nebo podtržítkem. Platí zde však jistá omezení:

- Identifikátor nesmí obsahovat žádné jiné znaky než písmena, čísla a podtržítko
- V identifikátorech se nesmí používat čeština
- U identifikátoru se rozlišuje 63 znaků
- Identifikátorem nesmí být klíčové slovo
- Nepsaná pravidla používání identifikátorů:
- Psát stejný identifikátor pokaždé stejnými písmeny
- Používat CODE INSIGHT
- Každý příkaz se píše na nový řádek
- Slova begin a end se píší na samostatný řádek
- Každý vnořený blok jde o 2 znaky doprava
- Pokud je příkaz rozdělen na více řádků, je každý další řádek o jeden znak odsazen doprava
- Příkaz se nikdy nerozděluje uprostřed výrazu
- Jména identifikátorů se volí tak aby odpovídaly významu proměnné nebo procedury.

Proměnné: Každá proměnná je pojmenována identifikátorem, pomocí kterého se přistupuje k jejímu obsahu. Bloky paměti, které slouží k ukládání potřebných dat se nazývají proměnné. Dříve než se v programu použije jakákoliv proměnná musí se na začátku programu deklarovat. Deklarace proměnné začíná klíčovým slovem **var**.

Syntaktický diagram: **var** Identifikátor proměnné : datový typ proměnné;

Příklad:
var Cislo : Integer;
var Cislo1, Cislo2, Cislo3: Integer;

Deklarace konstant

Kromě proměnných požíváme také konstanty. Konstanty se podobají proměnným. Stejně jako proměnné, tak i konstanty mají rezervované kousky paměti RAM, ve

kterých je uložena nějaká hodnota. Konstantou rozumíme identifikátor, označující hodnotu, která se nemění. Konstanty deklarujeme v deklarační části unity nebo procedury za klíčovým slovem **const .**

Příklad :
Const
N = 5;
Nazev = ’Zaciname Delphi’;
Min = 0;
Polomer = 4;
Pi = 3,14;

Komentáře

Protože programátor také zapomíná a po určité době si již nevzpomene jak to naprogramoval je vhodné doplňovat příkazové řádky komentářem. Komentáře se nezúčastňují překladu programu. Delphi rozlišují 3 druhy komentářů:

1.
{Komentář který je rozdělen na více řádků se omezuje složenými závorkami}
2.
(* Stejný komentář lze vložit i mezi hvězdičku a závorku*)
3.
// dvě lomítka označují komentář na jeden řádek

Datové typy

Jistě jste si všimli, že při deklaraci identifikátoru proměnné za klíčovým slovem var musíte uvést za dvojtečkou také datový typ. Typ proměnné definuje množinu hodnot, kterých může proměnná nabývat a množinu operací, které lze s danou proměnnou provádět. Existují 2 druhy datových typů. Ty standardní, které jsou obsaženy v jazyku Delphi a uživatelem definované datové typy.

Datové typy je možné rozdělit do několika skupin:

- Celočíselné typy
- Reálné typy
- Znaky
- Logické hodnoty
- Ukazatele
- Řetězce
- Uživatelem definované datové typy

Ordinalita datového typu. Datový typ je ordinální pokud jsou pro něj definovány následující funkce:

1. Funkce Ord Zjistí ordinální hodnotu prvku např Ord (‘A’) = 65; Ord (‘B’) = 66;
2. Funkce Pred vrací hodnotu, která předchází hodnotě uvedené v závorce Pred (1) vrací nulu, Pred (‘C’) = ‘B‘;
3. Funkce Succ vrací následující hodnotu argumentu Succ (1) = 2;

Succ (‘B’) = ‘C‘;
4. High výsledkem této funkce je nejvyšší hodnota, kterou může tato funkce nabývat
5. Low výsledkem této funkce je nejnižší hodnota, kterou může tato funkce nabývat

Celočíselné datové typy

Integer	-2147483648 .. 2147483647	32 bitů se znaménkem
Byte	0..255	8 bitů
ShortInt	-128..127	8 bitů
Smallint	-32768..32767	16 bitů
Int64	$-2x10^{63}..2x10^{63-1}$	64bitů
Word	0..65535	16 bitů
Longword	0..4294967295	32 bitů

Reálné datové typy

Type	Range	Significant digits	Size in bytes
Real48	2.9 x 10^-39 .. 1.7 x 10^38	11-12	6
Single	1.5 x 10^-45 .. 3.4 x 10^38	7-8	4
Double	5.0 x 10^-324 .. 1.7 x 10^308	15-16	8
Extended	3.6 x 10^-4951 .. 1.1 x 10^4932	19-20	10
Comp	-2^63+1 .. 2^63 -1	19-20	10
Currency	-922337203685477.5808.. 922337203685477.5807	19-20	8

Typ currency nepracuje jako ostatní s pohyblivou řádovou čárkou, ale s pevnou řádovou čárkou a je určen pro finanční výpočty. Rozsah celočíselných a reálných datových typů není nutné si pamatovat, protože je snadno najdeme v nápovědě.

Datový typ Char

Pro uložení jednotlivých znaků do paměti se používají proměnné typu Char. Velikost typu Char je 1 byte, což znamená, že můžete použít 255 znaků. Každý znak je uzavřen mezí dvěma apostrofy (Např. 'C'). To je sice všechno hezké, ale stále zbývá jedna záhada. Jak může počítač, který je vyhlášený tím, že umí zpracovávat pouze čísla a nic jiného, pracovat se znaky. Odpověď je jednoduchá a je uložena v ASCII (American Standard Code for Information Interchance) tabulce. To je mezinárodní kódová tabulka, v níž je každému znaku, který lze v programu použít, přiřazena odpovídající číselná hodnota. Kompletní ASCII tabulku lze nalézt v každé druhé knize o programování, protože zabírá na stránce dost místa a každý autor s radostí uvítá možnost bezpracně zvýšit počet stran své publikace. Já se však touto možností zlákat nenechám a na ušetřeném prostoru raději upozorním na to, jak lze v případě potřeby zjistit znak, který odpovídá příslušnému číslu či naopak, jak určit ordinální hodnotu vybraného znaku. K převedení znaku na číselnou hodnotu slouží funkce **Ord**, o které již byla dříve zmínka. Jejím opakem je funkce **Chr**, která vrací znak z ASCII tabulky. Např. Chr (65) = 'A', protože znaku 'A' odpovídá hodnota 65. Je ovšem třeba upozornit na to, že sice každému číslu v rozsahu 0 - 255 odpovídá nějaký znak, ale zdaleka ne všechny jsou vhodné ke grafickému zobrazení. Zejména spodní část ASCII tabulky není příliš smysluplná!! Pokud potřebujete použít nějaký znak, který není na klávesnici, a nelze ho tedy napsat mezi dva apostrofy, je možné použít jeho ordinální hodnotu uvozenou znakem #. Např. místo znaku konce

odstavce, který se vkládá po stisku klávesy ENTER, lze použít: #13. Poslední věc která v souvislosti se znaky stojí za zmínku, je kódování češtiny. Horních 128 znaků ASCII tabulky není univerzálních, ale jsou vyhrazeny pro národní znakové sady. Proto v programech může docházet (často skutečně dochází) k zobrazování nesmyslných znaků, na jejichž místě by v jiné znakové sadě byla umístěna písmena s diakritikou.

Logické datové typy

Proměnné tohoto ordinálního typu mohou nabývat pouze dvou hodnot. Pro ně jsou předdefinovány dvě konstanty True a False. Tyto proměnné se hojně využívají pro vyhodnocování nejrůznějších situací a budeme se s nimi často setkávat.

Type	range	size of
Boolean	True, False	1

Přetypování

Jazyk Delphi je přísně typový jazyk. To znamená, že za všech okolností přísně hlídá datové typy jednotlivých proměnných a nedovolí, aby se s proměnou prováděly operace mimo tento typ. Například není možné do proměnné typu Byte uložit znak a naopak, v proměnné Char nesmí být nic jiného než znaky. Nicméně někdy by bylo výhodné, kdyby se obsah proměnné určitého typu dal umístit do proměnné nějakého jiného typu, například číslo uložit jako řetězec znaků a pak s ním dále pracovat jako s textem.Takové operace jsou možné. Pokud ovšem oba typy mají stejnou velikost, je možné použít explicitní přetypování (Typecast). Uvažujme o možnosti, kdy je potřeba do proměnné typu Byte umístit nějaký znak, jak je to vidět v následujícím příkladu:

```
var
i:byte;
c:char;

begin
c := 'A';          //Do proměnné c se uloží znak 'A'
i := 'c';          {Toto je hrubá chyba!!!
                   Překladač oznámí chybu: "Incompatible types:
                   'Byte' and 'char' "}
i := Byte('c') ; {Toto je korektní pře typování. V i bude
                   číselná hodnota reprezentovaná znakem 'A'.}
end;
```

Celé kouzlo spočívá v tom, že se před proměnnou, která má být přetypována, napíše výsledný typ. Některé typy není nutné přetypovávat vůbec, protože to Delphi dělají implicitně. Jedná se například o přetypování v rámci stejné skupiny typů (reálné, celočíselné) nebo pokud se přetypuje celočíselný typ na reálný. Je dobré vyzkoušet si celý problém na několika případech pro různé typy, jak to je vidět třeba v následující ukázce:

```
var
i : Integer; r : Real;

begin
   i:=2; //do proměnné i se umístí číslo 2
   r := i; {Toto je v pořádku, provede se implicitní pře typování a v proměnné r je
   hodnota 2,OOO}
   i := r;  {Chyba!! Opačně nelze provést implicitní pře typování}
   i := Integer(r) ; {Chyba!! Nelze použít ani explicitní velikost, protože typ Real
  a Integer    mají různou velikost}
end;
```

Naprostou raritou mezi datovými typy Delphi je typ Variant. Jeho velikost je 16 bytů a do proměnné tohoto typu lze umístit hodnotu libovolného jiného typu. Pouze u typu Variant je tedy možné vidět takovýto kód:

```
var V: variant;
begin
   v:= 'nijaký text';
   v:= 0.222;
   v:= '%' i
end;
```

Mohlo by se tedy zdát, že používání ostatních datových typů je zbytečné a Variant je všechny nahradí. Ovšem není všechno zlato co se třpytí a je potřeba si uvědomit, že daní za tuto univerzálnost je jednak velké místo zabírané v paměti a také doba zpracování, která je u typu Variant až o několik řádů delší než u ostatních typů. Výše uvedený přehled typů není samozřejmě ani zdaleka kompletní. Delphi poskytují daleko více možností, než se na první pohled může zdát. Dříve než si uvedeme ostatní složitější typy, je vhodné znát ještě další rysy jazyka. Pokud jste zde dosud nenašli svůj oblíbený datový typ, nezoufejte. Budete-li mít ještě trošku trpělivosti, jistě nepřijdete zkrátka. Z předchozích kapitol plyne, že proměnné slouží k ukládání hodnot do paměti. Zde se však nabízí otázka, jaká hodnota je proměnné uložena bezprostředně po spuštění programu. Jazyk Delphi se na rozdíl od některých jiných programovacích jazyků (Basic, Fortran) nestará o to, jaká je počáteční hodnota v proměnné. Ta může být libovolná. Nikdy nespoléhejte na to, že na počátku je v proměnné nula. Rozhodně to nemusí být pravda a je jen otázka času, než se vám podobný předpoklad ošklivě vymstí.