

3 Podstawowe pojęcia

3.1 Budowa programu

Program pascalowy buduje się z elementarnych jednostek języka zwanych symbolami. Symbol składa się ze znaku lub ciągu znaków alfabetu .

Alfabet zawiera :

- ✓ duże i małe litery alfabetu łacińskiego i znak _
- ✓ cyfry od 0 do 9
- ✓ symbole specjalne (jednoznakowe)
- ✓ + - * / = < > { } [] () . : ; ^
- ✓ Ze znaków tworzy się symbole specjalne (wieloznakowe)
- ✓ <> <= >= := ..
- ✓ spacja
- ✓ i słowa kluczowe (zastrzeżone)

W taki sam sposób tworzymy identyfikatory stałych, zmiennych, typów, podprogramów etc. Symbole specjalne, podobnie jak znak spacji są separatorami jednostek leksykalnych.

3.2 Słowa kluczowe

Słowa kluczowe są to spójne ciągi liter tworzące zarezerwowane słowa angielskie o ustalonym znaczeniu. Używa się ich w z góry określony sposób. Są zastrzeżone, tzn. nie mogą być zmieniane przez programistę. Ich nazw nie można w programie użyć jako identyfikatora do jakiegoś elementu. Poniżej ich lista:

And	Array	Asm	Begin	Case
Const	Constructor	Destructor	Div	Do
Downto	Else	End	File	For
Function	Goto	If	Implementation	In
Inherited	Inline	Interface	Label	Mod
Nil	Not	Object	Of	Or
Packed	Procedure	Program	Record	Repeat
Set	Shl	Shr	String	Then
To	Type	Unit	Until	Uses
Var	While	With	Xor	

Oprócz słów kluczowych w Turbo Pascalu występują **Dyrektywy** języka. W odróżnieniu od słów kluczowych, nie są one zastrzeżone - podane wyrazy mogą więc być identyfikatorami zdefiniowanymi przez programistę, co jednak nie jest zalecane. Pewnym wyjątkiem są dyrektywy **private** i **public**, które w obiektach, ale tylko w nich, są słowami zastrzeżonymi. Dyrektywą jest każdy z następujących wyrazów :

Absolute	Far	Near	Public
Assembler	Forward	Private	Virtual
External	Interrupt		

Przykłady identyfikatorów

Ala	ala	Do
Ola1_1		Ola-1
Koniec		Początek
Liczba	Integer	String
Pascal	Far	Typ danych
_For	Private	23cia

3.3 Podstawowe typy danych

Język Pascal pozwala na abstrahowanie od reprezentacji danych w pamięci komputera.

Osiągnięto to poprzez wprowadzenie pojęcia typu. Przyjęto założenie, że każda zmienna, wyrażenie lub funkcja jest pewnego typu.

Typ ten określa

- ✓ zbiór wartości stałej, zmiennej etc.
- ✓ zbiór operacji jakie można wykonać na obiektach

3.3.1 Typy całkowite

TYPY CAŁKOWITE - są to wartości liczbowe, które mieszczą się w podanych zakresach, im większy zakres to automatycznie zwiększa się zapotrzebowanie liczby na pamięć:

SHORTINT	(-128..127)	1 bajt
INTEGER	(-32768..32767)	2 bajty
LONGINT	(-2147483648..2147483647)	4 bajty
BYTE	(0..255)	1 bajt
WORD	(0..65535)	2 bajty

3.3.2 Typy rzeczywiste

Nazwa	Zakres	Długość	Dokładność (cyfr)
real	2.9e-39 ÷ 1.7e38	6 bajtów	11 ÷ 12
single	1.5e-45 ÷ 4.0e38	4 bajty	7 ÷ 8
double	5.0e-324 ÷ 1.7e308	8 bajtów	15 ÷ 16
extended	3.4e-4932 ÷ 1.1e4932	10 bajtów	19 ÷ 20
comp	-9.2e18 ÷ 9.2e18	8 bajtów	19 ÷ 20

Użycie typów Comp lub Extended wymaga deklaracji wykorzystywania w przez program koprocessora("Must be in 8087 mode to compile this." {\$N+}) W menu OPTIONS/COMPILER w dziale Numeric processing należy wybrać opcję 8087/80287

Opcja {\$E+} umożliwia uruchomienie programową emulację koprocessora, w przypadku braku jednostki sprzętowej.

3.3.3 Znaki i łańcuchy znaków

Zbiór znaków (typ char) został zdefiniowany przez tablicę kodów ASCII.

3.3.3.1 Znaki sterujące (kontrolne)

Znaki o kodach 0 - 31 , 127 noszą nazwę znaków sterujących . Ich graficzna reprezentacja nie zawsze będzie wyświetlana . Są to znaki użyteczne przy sterowaniu komputerem .

Np

```
program bell;
begin
  write(#7);           write(char(7));
end.
```

Wykonanie programu powoduje wydanie dźwięku dzwonka (ang. Bell).

3.3.3.2 Znaki podstawowego i rozszerzonego kodu ASCII

Są to znaki odpowiadające kodom 32 (spacja) - 126 (~). Mieszczą się tu znaki alfabetu Pascala (małe i wielkie litery łacińskie , liczby ,znaki operatorów itp.).

Znaki o kodach 128- 225 są to znaki tzw. rozszerzonego kodu ASCII .

Znaki te mogą być zmieniane , w celu uzyskania symboli specyficznych dla danego języka np. Å

Standardowo znajdują się tam znaki do kreślenia ramek , kilka znaków używanych w transkrypcji i inne .

Dostęp do tych znaków może być zrealizowany na trzy sposoby

1. #Nr_Znaku np. #65
2. wykorzystanie funkcji char
char (Nr_Znaku) ; np. char (65)
3. wypisanie symbolu znaku w apostrofach np. 'A'

3.3.3.3 Typ łańcuchowy -String

Umieszczenie pomiędzy apostrofami ciągu znaków (łańcucha) jest również dopuszczalne np.

```
Program string1;
begin
  write('PASCAL');
end .
```

Ciąg znaków jest związany z typem string; Oczywiście jest , że typ char jest w nim zawarty jako ciągi długości 1.

```
Program string_char;
begin
write('P','A','S','C','A','L');
end .
```

Inny specjalny napis to tzw. string pusty " (apostrof apostrof)

3.3.4 **Deklaracja stałych i zmiennych typów znakowych**

Stałe

```
const Litera = 'A';
const Imie = 'Anna';
```

Zmienne

```
var c : char;
var S : String;
var s10: String[10];
```

Zmienne z wartością początkową

(dotyczy TURBO PASCAL i BORLAND PASCAL)

```
const c : char = 'A' ; {1}
      s1 : String[1] = 'Q'; {2}
      S : String= 'A'; {3}
      s10 : String[10]= 'Kot'; {4}
      St : String = 'To jest napis !'; {5}
```

Jakie są podobieństwa i różnice pomiędzy deklaracjami {1},{2}i{3}?

3.4 Elementy programu

3.4.1 Struktura programu

```
Program Nazwa; {nagłówek programu}
Uses ...; {deklaracje użycia modułów}

Const ... {deklaracje stałych globalnych}

Type ... {deklaracje typów}

Var ... {deklaracje zmiennych globalnych}

Procedure Nazwa(paramerty); {nagłówek
    podprogramu - procedury}
Var ... {deklaracja zmiennych lokalnych}
Begin
    ... {instrukcje podprogramu}
End;

Function Nazwa(paramerty):Typ_wyniku;
    {nagłówek funkcji}
Var ... {deklaracja zmiennych lokalnych }
Begin
    ... {instrukcje podprogramu}
End;
BEGIN
    ... {instrukcje bloku głównego programu}

END. { zakończenie bloku głównego kropką}
Najkrótszy poprawny program w TP to:
Program Nazwa;
Begin
End.
```

3.4.2 Komentarze

Każdy program po pewnym czasie może stać się nawet trudy do zrozumienia nawet dla autora . Można jednak ten proces osłabić poprzez odpowiednie pisanie programów.

W tym celu należy opisywać wszystkie bardziej złożone fragmenty programów, a także używać identyfikatorów, których nazwy przybliżają ich zastosowanie.

Przykład 1:

```
(* komentarz *)
var n,m,k : integer;
begin
  n:=2;
  m:=6;
  k:=n*m;
  writeln(iloczyn { komentarz } );
  { to
    także
    jest
    komentarz }
end.
```

Wprawdzie to nie jest komentarz, ale kompilator potraktuje ten tekst jako komentarz.

Przykład 2:

```
Program iloczyn;
var czynnik1,czynnik2, { czynniki }
    iloczyn :      (*wynik mnożenia *)
    integer;      { typ zmiennych }
BEGIN
czynnik1:=2 ;      {przypisanie wartości do}
czynnik2:=2 ;      { czynników }
iloczyn:=czynnik1 * czynnik1;
                {wyliczenie wyniku mnożenia}
writeln('Wynik mnożenia ' , czynnik1,' przez
',czynnik2,' wynosi : ',iloczyn ) ;
END.
```

3.4.3 Instrukcja przypisania

Nadawanie wartości następuje bardzo często za pośrednictwem instrukcji przypisania, która ma ogólną postać :

```
zmienna := wyrażenie ;
```

Operator „ := ” nazywa się operatorem przypisania , symbol zmienna oznacza tu identyfikator; wyrażenie musi być typu konwertowanego do typu zmiennej. W wersji najprostszej:

```
zmienna := stała ;
```

```
liczba := 10 ;
czytamy : zmiennej liczba przypisz wartość 10 ;
Program Instr_przypisania_1 ;
var liczba :integer ;
BEGIN
liczba :=1;
writeln(liczba);
liczba :=100;
writeln(liczba);
liczba := -237;
writeln(liczba);
END.
{wynik wykonania programu
1
100
-237 }
```

Istnieje również inny schemat instrukcji przypisania mianowicie

```
zmienna1 := zmienna2 ;
```

Jednakże ta postać dopuszczalna jest gdy zajdzie jedno z:

- ✓ typy zmiennych są zgodne
- ✓ zmienna zmienna2 jest typu konwertowanego typu zmiennej

```
zmienna1
```

- typ real (zmienna1) i typ Integer (zmienna1)

Przykład

```
var a,b :integer;
begin
  a:=1;
  b:=a;
  writeln(c);
  writeln(b);
end .
```

Zmienne były tego samego typu .

Przykład

```
var      s : string ;
const   c : char = 'Z';
begin
  s := c ;
  writeln(c);
  writeln(s);
end .
```

zmienne miały różne typy

```
Program Instr_przypisania_2 ;
var liczba  : integer ,
    litera   : char;
```

```
BEGIN
liczba := 33;           {wynik wykonania
programu }
litera := char (liczba)      { ! }
writeln(litera);
END.
```

```
Program const_var;
const
  c   : char='A';
  s1  : string[1]='Q';
  s   : string='A';
  s10 : string[10]='kot';
  st  : string='To jest napis';
begin
  writeln('*****');
  writeln('s1 :',s1);
  writeln('c  :',c);
  s1:='kkk';
  writeln('s1 :',s1);
  c:='u';
  writeln('s10 :',s10);
  s10:=st;
  writeln('s10 :',s10);
  writeln('c  :',c)
end.
```

3.4.3.1 Różnice pomiędzy identyfikatorami stałych i zmiennych

1. wartości stałych są określone przed rozpoczęciem programu natomiast zmienne otrzymują wartość w trakcie wykonywania programu ;
2. (TP) tylko zmienne deklarowane w sekcji CONST mają nadaną wartość przed rozpoczęciem programu ;
3. wartość identyfikatora stałej nie może ulec zmianie podczas wykonywania programu , podczas gdy zmienna może przyjmować różne wartości

W instrukcji przypisania `zmienna := stała ;` wartość stałej musi być tego samego typu , co typ zmiennej . Jedynym wyjątkiem jest instrukcja przypisania zmiennej typu rzeczywistego stałej typu całkowitego np.

```
program Real_int;
var r :real;
begin
r := 100 ;
end.
```

Które z definicji są poprawnymi definicjami stałych :

- (a) CONST Licza = 250 ;
- (b) CONST Liczba = 1,2 ;
- (c) CONST Znak = 'A' ;
- (d) CONST Max = 250 ;min= -150.7;
- (e) CONST zakres = 250 ..500;
- (f) CONST L : integer = 250 ;
- (g) CONST L : real = 250 ;
- (h) CONST L : string = 250 ;
- (i) CONST L : char = 250 ;

Które ze zmiennych są zadeklarowane poprawnie :

- (a) VAR 1a,2a,3a : Real;
- (b) VAR a1,a2,a3 : Real;
b1 : char;
a3 : Integer
- (c) VAR a :Char ; b Integer ; c : real;
- (d) VAR a1;a2;a3 : Real ,
c : char ;

Zadanie

Napiszmy program , który wydrukuje tekst : $S = 1+2 + 3+4+5$, a w następnym wierszu wartość zmiennej S .

```
Program Komentarz_1;
var S : Integer;
ST : String;
begin
St := ' S = 1+2+3+4+5 ' ;
S := 1+2+3+4+5 ;
writeln(st);
writeln(s);
end.
```