PP Laboratorium 3.2

Funkcje i procedury

Funkcje i procedury są małymi podprogramami. Program składa się z wielu funkcji i procedur wykonujących różne zadania. Dobrze zaprojektowany program składa się z krótkich i przejrzystych funkcji i procedur o nazwach zgodnych z realizowanym przez nie zadaniem. Jeśli np. jakaś funkcja ma wykonać złożone zadanie powinna wywoływać inne funkcje realizujące poszczególne podzadania składowe.

Nagłówek procedury

```
procedure NazwaProcedury  ( lista, parametrow: Typ1 ;  par3: Typ2 )  ;
   ①          ②          ③                ④                    ⑤ ⑥
```

1. Słowo kluczowe **`procedure`**.
2. Nazwa procedury.
3. Nawias otwierający listę parametrów.
4. Lista parametrów.
5. Nawias zamykający listę parametrów
6. Nagłówek zawsze zakończony jest średnikiem, niezależnie czy jest Deklaracją czy nagłówkiem definicji procedury.

Nagłówek funkcji

Nagłówek funkcji jest podobny do procedury, różnią go tylko dwa elementy: słowo kluczowe function i typ zwracany.

```
function NazwaFunkcji  ( par1: Typ1 ; par2, par3: Typ2 ): TypWyniku ;
   ①          ②     ③                ④              ⑤   ⑥      ⑦
```

1. Słowo kluczowe **`function`**.
2. Nazwa funkcji.
3. Nawias otwierający listę parametrów.
4. Lista parametrów.
5. Nawias zamykający listę parametrów
6. Typ zwracanego przez funkcję wyniku.
7. Nagłówek zawsze zakończony jest średnikiem, niezależnie czy jest Deklaracją czy nagłówkiem definicji procedury.

Nazwy procedur i funkcji mogą składać się z liter, cyfr i znaku podkreślenia ale nie mogą zaczynać się od cyfry. Nazwy funkcji i procedur powinny być pisane w całym projekcie jednolicie. Zależnie od przyjętej konwencji rozpoczynając od małej lub dużej litery, bez znaków podkreślenia, zaczynając kolejne słowa dużą literą. np.: **`nazwaProcedury`** lub **`NazwaProcedury`**.

Lista parametrów

Może być pusta jeśli procedura nie przyjmuje żadnych parametrów. Może zawierać dowolne parametry różnych typów. Każdy parametr musi mieć podany typ. Jeśli kilka parametrów jest tego samego typu można umieścić je na jednej liście oddzielając przecinkami.

Przykłady:

1. `procedure BezParametrow ( );`
2. `procedure DwaRozneParametry ( a:Integer ; b:String );`
3. `procedure TrzyTakieSameParametry ( a, b, c: Integer );`
4. `procedure Czwarta ( const s: String ; var n: Integer );`
5. `procedure Piata ( var n: Real ; a, b : Real ; const c Real);`

W przykładzie trzecim wszyskie paramery są jednego typu więc mogły zostać umieszczone na wspólnej liście. W przykładzie czwartym użyto modyfikatorów **`const`** i **`var`**. W przkładzie piątym wszystkie parametry są jednego typu ale są przekazywane z różnymi modyfikatorami (const, var lub bez modyfikatora).

Przekazywanie parametrów

Parametry można przekazywać domyślnie jako **kopię** lub przekazać oryginalną zmienną unikając kopiowania. Ma to znaczenie przy złożonych zmiennych np. tablicach, stringach i rekordach. Do zmiany sposobu przekazywania parametrów służą modyfikatory takie jak **`const`** i **`var`**.

`CONST` – oznacza przekazanie do funkcji oryginału ale bez możliwości edycji. Przydatne np. w sytuacji kirdy chcemy wypisać całą tablicę (unikamy kopiowania a przecież nie będziemy w niej niczego zmieniać).

`VAR` – oznacza przekazanie oryginalnej zmiennej (przez zmienną – variable) ale uniemożliwia przekazania do funkcji stałych.

Przykłady:

```
procedure procStd( a: Integer );
begin
    a := 2 * a;  { nic ciekawego się nie stanie,
                   modyfikujemy kopię która w następnej linii przestaje istnieć }
end;

procedure procConst( const a: Integer );
begin
    // a := 2 * a;  { nie da się skompilować z tą linią w kodzie }
end;

procedure procVar( var a: Integer );
begin
    a := 2 * a;  { zmodyfikuje oryginalną zmienną przekazaną do procedury }
end;
```

Zwracanie wyniku funkcji

Funkcja w przeciwieństwie do procedury zwraca wynik. Wynikiem jest wartość typu podanego w nagłówku. Wartość zwracaną przypisujemy do specjalnej zmiennej której nigdzie nie deklarujemy. Zmienna do której przypisujemy ma nazwę taką samą jak funkcja (tak jakbyśmy przypisywali wartość funkcji). Można też skorzystać z nazwy **result**. Pisząc funkcję musimy zdecydować się na jedną z tych dwóch nazw, nie wolno ich mieszać.

```
function  StoDwadziesciaTrzy ( ) : Integer;
begin
    StoDwadziesciaTrzy := 123;
end;
```

lub

```
function  StoDwadziesciaTrzy ( ) : Integer;
begin
    result := 123;
end;
```

Funkcje matematyczne w Pascalu

Funkcje można traktować jako rozwinięcie funkcji znanych z matematyki. Funkcja przyjmuje jakieś wartości (jedną lub więcej) i przyjmuje (zwraca) wartość dla zadanych parametrów wejściowych.

Przykładowa definicja funkcji znana z matematyki:

$$f(x) = x^2$$

Odpowiednik tej definicji w Pascalu:

```
function  f ( x:Real ) : Real;
begin
    f := x*x;
end;
```

lub:

```
function  f ( x:Real ) : Real;
begin
    result := x*x;
end;
```

Przykład użycia:

```
Var
    a, b, c: Real;
Begin
    a := 2.0;         // a = 2
    b := f( a );      // b = 4
    c := f( b );      // c = 16
    a := f( f(a) );   // a = f( 4 ) = 16
End;
```

Zwracanie wyniku z procedur

Aby zwrócić wyniki działania procedury trzeba zmodyfikować jeden (lub więcej) z parametrów, podobnie jak robi to procedura **`ReadLn(zmienna)`**. Aby móc zmodyfikować wartośćprzekazaną do procedury trzeba ustawić sposób przekazania parametru z domyślnego kopiowania na przekazanie przez zmienną (czyli przekazanie oryginału) za pomocą słowa kluczowego **`var`** lub **`out`**. Metoda ta jest szczególnie korzystna przy modyfikacji dużych tablic, ponieważ pozwala uniknąć zbędnego kopiowania.

$$y \leftarrow x^2$$

```
procedure  proc1 ( out y: Real; x: Real );
begin
    y := x*x;
end;
```

$$x \leftarrow x^2$$

```
procedure  proc2 ( var x: Real );
begin
    x := x*x;
end;
```

Przykłady zastosowania funkcji f() i procedur podnoszących podaną liczbę do kwadratu:

```
b := 3;
a := f(b); // a = 9    Przekazanie wartości zmiennej b=3 (kopii)
a := f(2); // a = 4    Przekazanie wartości 2

b := 3;
proc1(a, b); // a = 9    Przekazanie zmiennej a jako miejsca na wynik
proc1(a, 2); // a = 4    Drugi parametr może być zmienną lub wartością stałą

a := 5;
proc2(a); // a = 25    Procedura modyfikująca wartość przekazanej zmiennej

//proc2(4);      //  Nie zadziała nie można zmodyfikować liczby 4
                 //  Jest to równoważne wywołaniu funkcji:
//4 := f(4);     //  Jak widać podobna próba dla funkcji jest oczywistym błędem.
```

Program laboratorium 3

- Moduły
  - Dodawanie własnego modułu
  - Sekcje w modułach
- Procedury i funkcje
  - Deklaracja procedury
  - Deklaracja funkcji
  - Przekazywanie parametrów (var, const)
  - Przekazywanie tablic (var, const)
  - Zwracanie wyniku funkcji