

## 8 Podprogramy

### 8.1 Istota programowania proceduralnego

Historycznie ujmując pierwsze programy korzystały z programowania liniowego. Nie wspomina tu z oczywistych względów języków maszynowych, czy też „języków” programowania pierwszych komputerów. Programowanie proceduralne umożliwia podział jednego wielkiego projektu na kilka mniejszych zadań.

#### 8.1.1 Pojęcie procedury

**Procedure** NazwaProcedury ( *var* argument : Typ\_argumentu ) ;

*Deklaracje stałych i zmiennych lokalnych;*

Begin

*Ciało procedury*

End;

#### 8.1.2 Pojęcie funkcji

**Function** NazwaFunkcji ( *var* argument : Typ\_argumentu )

: Typ\_wyniku ;

*Deklaracje stałych i zmiennych lokalnych;*

Begin

...

NazwaFunkcji:=wyrażenie *typu* Typ\_wyniku;

...

End;

## 8.1.3 Deklaracje, definicja i wywołanie podprogramów

### 8.1.3.1 Procedury

```
Program podprogramy;
Procedura powieksz (var x:real;y:real);
Begin
  X:=x+y;
End;
  Var x,y : real;
BEGIN
  x:= 2.0; y:= 1.0;
  Writeln(x, ' ', y);
  Powieksz (x,y)
  Writeln(x, ' ', y);
END.
```

### 8.1.3.2 Funkcji

```
Program Funkcje;
Function Silnia(N:byte):longint;
Var i:Byte;
    S: longint;
Begin
  S:=1;
  For i:=1 to N do s:= s * i;
  Silnia:=s;
End;

Function Potega(x:real;N:byte):Real;
Var i:Byte;
    P: longint;
Begin
  P:=1;
  For i:=1 to N do p:= p * x;
  Potega:=p;
End;
Var w,a:real; b:byte;
Begin
```

```

a:= 2.0; b:= 3;
Writeln(b, ' !=' silnia(b) );
w:= pot(a,b);
Writeln(a, '^', b, '= ', w);

```

End.

## 8.1.4 Parametry podprogramów

### 8.1.4.1 Parametry formalne i aktualne, zmienne lokalne

Parametry podprogramów umożliwiają komunikację procedury ze „światem zewnętrznym”, czyli programem głównym bądź innym podprogramem, który go wywołał. Dzięki parametrom podprogramy działając na sparametryzowanych danych mogą być wielokrotnie używane do wykonywania zadań na różnych danych. Parametry procedur i funkcji stają się ich zmiennymi lokalnymi. Jeśli jakaś zmienna globalna ma nazwę tożsamą z nazwą parametru to w podprogramie będzie widoczna tylko zmienna lokalna związana z parametrem.

### 8.1.4.2 Przekazywanie parametru poprzez wskaźnik i wartość

```

Program Parametr_var;
Procedure Dordaj( var x:Integer; y : Integer);
Begin
  x := x + 2;
  y := y + 3;
  writeln('x =', x , 'y=', y );
End;

```

```

BEGIN
  x := 5;
  y := 6;
  writeln('x =', x , 'y=', y );
  dodaj( x, y);
  writeln('x =', x , 'y=', y );
END.

```

## 8.1.5 Inne przykładowe funkcje

- Zamiana dwóch zmiennych liczbowych
- Maximum i minimum
- Podprogramy operujące na elementach tablicy

## 8.1.6 Zalety programowanie z wykorzystaniem podprogramów

- Podział programu na mniejsze segmenty
- Wielokrotne wykorzystanie tego samego kodu
- Realizacja podobnych problemów za pomocą tego samego podprogramu – wykorzystanie parametrów

## 8.1.7 Rekurencja

### 8.1.7.1 Cechy rekurencji

### 8.1.7.2 Wywoływanie samej siebie

### 8.1.7.3 W pewnych warunkach następuje koniec wywoływania rekurencyjnego

### 8.1.7.4 Rekurencja a iteracja

```
Program Rekurencja;  
Function Silnia(N:byte):longint;  
Begin  
  If N>0 then silnia:= n*silnia(N-1)  
  Else silnia:=1;  
End;  
  
Function Potega(x:real;N:byte):Real;  
Begin  
  If N>0 then potega:= x*potega(x,N-1)  
  Else potega:=1;  
End;  
Procedure Hanoi(zrodlo,pomoc,cel,ilosc:byte);  
Begin  
  If ilosc>0 then  
  begin  
    Hanoi(zrodlo,cel,pomoc,ilosc-1);  
    Przenies(zrodlo,cel);  
    Hanoi(pomoc,zrodlo,cel,ilosc-1);  
  End;  
End;
```

Gdzie procedura przenies może mieć w najprostszym przypadku postać:

```
Procedure przenies (zrodlo, cel:byte) ;  
Begin  
  Write (zrodlo, '->' , cel, ' ');  
End;
```

Ciekawszym rozwiązaniem, byłoby wykorzystanie tablic do przechowywania krążków (liczb) i wyświetlania ich zawartości. To umożliwiło by zasymulowanie animacji.

Można też by zauważyć, że skoro zmienne `zrodlo`, `pomoc`, `cel` są liczbami powiedzmy 1, 2, 3 to łatwo zauważyć, iż

$$1 + 2 + 3 = 6 .$$

$$zrodlo + pomoc + cel = 6 \text{ czyli}$$

$$cel = 6 - zrodlo - pomoc$$

a procedurę mo zemy zamienić na:

```
Procedure Hanoi (zrodlo, pomoc, ilosc:byte) ;
```

### 8.1.7.5 Zalety i wady rekurencji

- ✓ Zastępuje iterację
- ✓ Prostota krótkiego kodu

— Łatwe przepełnienie stosu

— Kłopoty z rekurencyjną definicją problemu