Politechnika Świętokrzyska
Wydział Elektrotechniki, Automatyki i Informatyki
Katedra Elektrotechniki Przemysłowej i Automatyki
Zakład Urządzeń i Systemów Automatyki

**Informatyka 2**

# Wprowadzenie do języka Pascal

Instrukcja laboratoryjna
(wersja robocza)

Paweł Strączyński
2018

## 1. Struktura programu w języku Pascal, podstawowe dyrektywy preprocesora, komentarze.

Szkielet najprostszego programu w języku Pascal przedstawiono na listingu 1.1. Program zaczyna się od słowa kluczowego **program** po którym należy podać nazwę programu. Linia z deklaracją nazwy programu musi kończyć się średnikiem. Główną część programu należy umieścić w otoczeniu słów kluczowych **begin** oraz **end.** Ponadto po słowie kluczowym **end** musi wystąpić znak kropki. Komentarzem jest każdy fragment kodu zawarty między znakami `{` oraz `}` lub `(*` oraz `*)`.

*Listing 1.1. Szkielet programu w języku Pascal*

```
program prog1;

begin
      {To jest komentarz}
      (* To też jest komentarz*)
end.
```

## 2. Zmienne – typy, deklaracje, zasięg widoczności

Zmienna to fragment pamięci, posiadający nazwę i służący do przechowywania wartości (danych) wykorzystywanych podczas działania programu. Każda zmienna w języku Pascal posiada typ określający zakres wartości jakie może przyjmować. Podstawowe typy zmiennych przedstawiono w tabeli 2.1.

*Tabela 2.1. Podstawowe typy zmiennych w języku Pascal*

| Typ | Zakres przechowywanych wartości | Rodzaj przechowywanej wartości |
|---|---|---|
| Byte | 0..255 | Liczba całkowita |
| Shortint | -128..127 | Liczba całkowita |
| Word | 0..65535 | Liczba całkowita |
| Integer | -32768..32767 | Liczba całkowita |
| Longword | 0..4000000000 | Liczba całkowita |
| Longint | -2000000000..2000000000 | Liczba całkowita |
| Real | $2.9 \cdot 10^{-39}$ .. $1.7 \cdot 10^{38}$ | Liczba rzeczywista |
| Single | $1.5 \cdot 10^{38}$ .. $3.4 \cdot 10^{38}$ | Liczba rzeczywista |
| Double | $1.5 \cdot 10^{-309}$ .. $1.7 \cdot 10^{309}$ | Liczba rzeczywista |
| Extended | $3.6 \cdot 10^{-4951}$ .. $1.1 \cdot 10^{4932}$ | Liczba rzeczywista |
| String | - | Łańcuch znakowy |

Deklarowanie zmiennych w języku Pascal odbywa się po słowie kluczowym **var** według schematu jak poniżej:

```
var nazwa_zmiennej: typ_zmiennej;
```

Przykłady deklaracji zmiennych przedstawiono na listingu 2.1. Zmienne które są zadeklarowane poza procedurami i funkcjami (jak pokazano na listingu) są zmiennymi globalnymi. Zmienne deklarowane wewnątrz procedur i funkcji to zmienne lokalne i są one widoczne wewnątrz tej procedury lub funkcji wewnątrz której zostały utworzone.

*Listing 2.1. Przykład deklaracji funkcji w języku Pascal*

```
program prog1;

var
      foo, bar: Integer; {Deklaracja zmiennych typu całkowitego}
      foobar: Real; (* Deklaracja zmiennej typu rzeczywistego *)

begin
      foo := 5;
end.
```

Jak widać powyżej do zadeklarowanej wcześniej zmiennej można dokonać przypisania. Operatorem przypisania w języku Pascal jest dwuznak :=. Należy zauważyć, że każda instrukcja w języku Pascal musi kończyć się średnikiem. Nazwa zmiennej może składać się z liter i cyfr, jednak zawsze musi zaczynać się literą. W nazwie zmiennej nie mogą występować spacje. Umownie przyjmuje się, że nazwy zmiennych pisane są małymi literami. Dla odróżnienia stałe zapisujemy umownie wielkimi literami. Deklaracja stałych następuje po słowie kluczowym const. Przykład deklaracji stałych przedstawiono na listingu poniżej.

*Listing 2.1. Przykład deklaracji funkcji w języku Pascal*

```
program prog2;

const
      SPAM = 10; {Deklaracja stałych}
      EGGS = 'Lorem ipsum';
var
      foo, bar: Integer;

begin
      foo := SPAM;
      bar := SPAM + 2;
end.
```

Wartości stałych nie mogą być modyfikowane – wartość przypisywana jest na stałe przy deklaracji. Stałe mogą być wykorzystywane z operatorem przypisania lub innym operatorem – jak przedstawiono powyżej.

### 3. Podstawowe operatory

W tabeli 3.1 przedstawiono operatory arytmetyczne w języku Pascal.

*Tabela 3.1. Operatory arytmetyczne w języku Pascal*

| Operator | Opis |
|---|---|
| + | dodawanie |
| - | odejmowanie |
| * | mnożenie |
| / | dzielenie |
| DIV | dzielenie całkowite |
| MOD | reszta z dzielenia |

Dodatkowo symbol - jest używany jako operator odwrócenia znaku – operator jednoargumentowy. Pozostałe operatory wymagają dwóch argumentów i realizują podstawowe operacje arytmetyczne. Przykładowe z wykorzystaniem operatorów arytmetycznych przedstawiono na listingu 3.1.

*Listing 3.1. Przykład wykorzystania operatorów arytmetycznych*

```
program prog3;

const
      SPAM = 10;
var
      foo, bar, foobar, quux, quuz : Integer;

begin
      foo := 1; {Przypisanie wartości}
      foo := foo + 1; {Inkrementacja zmiennej z użyciem operatora dodawania}
      foobar := foo MOD 2; {Przypisanie wyniku operacji reszty z dzielenia}
      quux := foobar / 4;
      quuz := foo * SPAM;
end.
```

Operatory multiplikowane (`*`,`/`,`DIV`,`MOD`) mają wyższy priorytet niż addytywne (`+`,`-`).
W tabelach 3.2, 3.3 oraz 3.4 przedstawiono odpowiednio operatory: relacyjne, logiczne oraz bitowe.

*Tabela 3.2 Operatory porównania*

| Operator | Opis |
|---|---|
| | równe |
| <> | różne |
| < | mniejsze |
| > | większe |
| <= | mniejsze lub równe |
| >= | większe lub równe |

*Tabela 3.3 Operatory logiczne*

| Operator | Opis |
|---|---|
| OR | suma logicza |
| AND | iloczyn logiczny |
| NOR | negacja |
| XOR | alternatywa wykluczająca |

*Tabela 3.4 Operatory bitowe*

| Operator | Opis |
|---|---|
| SHL | przesunięcie bitowe w lewo |
| SHR | przesunięcie bitowe w prawo |

## 4. Instrukcje warunkowe

W języku Pascal do warunkowego wykonywania danego fragmentu programu służą:

- instrukcja warunkowa **`if`** ( **`if-else`**)
- instrukcja wyboru wielokrotnego **`cas`**

Najczęściej wykorzystywaną instrukcją warunkową jest instrukcja **`if`**. Przyjmuje ona następującą konstrukcję:

```
if WARUNEK then INSTRUKCJA1 (else INSTRUKCJA2);
```

Przy czym użycie słowa kluczowe **else** jest opcjonalne i oznacza, że INSTRUKCJA2 zostanie wykonana, gdy WARUNEK nie zostanie spełniony. Warunek to wyrażenie logiczne zawierające jeden z operatorów logicznych z tabeli 3.3. Warunek może składać się również z kilku wyrażeń logicznych „połączonych" którymś z operatorów logicznych przestawionych w tabeli 3.3. Jeżeli w ramach instrukcji **if** ma zostać wywołane więcej instrukcji niż jedna to należy zastosować konstrukcję:

```
if WARUNEK then
   begin
      INSTRUKCJA1;
      INSTRUKCJA2;
   end
(else)
   begin
      INSTRUKCJA3;
      INSTRUKCJA4;
   end
```

Przykłady wykorzystania instrukcji warunkowej **if** przedstawiono na listingu poniżej.

*Listing 4.1. Przykładowe wyrażenia warunkowe w języku C*

```
program prog41;

const
      SPAM = 10;
var
      foo, bar, foobar : Integer;

begin

      if SPAM = 8 then foo := 5;
      else foo := 2;
      if foo <> 0 then
      begin
            bar := 3;
            foobar := 4;
      end
      if foo > 0 then bar := 1;
      else bar := 0;
end.
```

Do wykonywania instrukcji warunkowych wielokrotnego wyboru, gdzie sprawdza się czy wartość wyrażenia pasuje do jednej z kilku wartości służy instrukcja **case.** Instrukcja ta przyjmuje następująca postać:

```
case WYRAŻENIE of
```

```
        WARTOŚĆ1: INSTRUKCJA1;
        WARTOŚĆ2: INSTRUKCJA2;
        WARTOŚĆ3: INSTRUKCJA3;
    end;
```

Jeżeli `WYRAŻENIE` jest równej której z `WARTOŚĆ-i` to wykonana zostanie stojąca przy niej wartości. Przykład użycia przedstawiono poniżej.

*Listing 4.3. Przykładowe użycie instrukcji warunkowej case*

```
program prog42;

const
      SPAM = 4;
var
      foo : Integer;
      bar : String;

begin
      case SPAM of
            1: bar := 'Jeden';
            2: bar := 'Dwa';
            3: bar := 'Trzy';
            4: bar := 'Cztery';
      end;
end.
```

## 5. Pętle

Pętle wykorzystuje się wszędzie tam gdzie zachodzi potrzeba wykonywania wielokrotnie tego samego ciągu instrukcji.

Do wykonywania tego samego ciągu instrukcji dopóki spełniony jest dany warunek w języku Pascal służy pętla **`while`**. Przyjmuje ona następującą konstrukcję:

```
while WARUNEK do INSTRUKCJA;
```

Jeżeli zamiast jednej instrukcji chcemy aby do chwili spełnienia warunku był wykonywany blok instrukcji to podobnie jak w przypadku instrukcji **`if`** należy go umieścić pomiędzy słowami kluczowymi **`begin`** oraz **`end`**.

Problemem z pętlą **`while`** może okazać się fakt, że może zdarzyć się przypadek gdy nie wykona się ona ani razu – wyrażenie nie jest spełnione. Aby mieć pewność że pętla wykonała się co najmniej raz należy użyć pętli **`repeat`**. Zasadniczą różnica między pętlą **`while`** jest to, że sprawdza ona warunek pod koniec każdego obiegu pętli po słowie kluczowym **`until`**. Pętle można stworzyć według schematu:

```
repeat INSTRUKCJA until WARUNEK;
```

W przypadku użycia grupy instrukcji wyjątkowo nie stosuje się słów kluczowych **begin** oraz **end**. Instrukcje umieszczane są zwyczajnie pomiędzy słowem **repeat** oraz **until**.
Do wykonywania obiegu ustaloną ilość razy często wygodniejsza w użyciu jest pętla **for**. Ogólna postać pętli **for** wygląda następująco:

```
for ZMIENNA := WYRAŻENIE1 to WYRAŻENIE2 do INSTRUKCJA;
```

gdzie:

- `ZMIENNA` – tzw. zmienna sterująca pętli. Musi być wartością typu liczbowego lub znakowego (typ porządkowy).
- `WYRAŻENIE1, WYRAŻENIE2` – zakres wartości jakie przyjmuje `ZMIENNA` w kolejnych obiegach pętli. Zmienna przyjmuje kolejno każdą wartość z przedziału <`WYRAŻENIE1, WYRAŻENIE2`> dokładnie raz w każdym obiegu pętli.

W powyżej przedstawionej konstrukcji aby pętla wykonała się `WYRAŻENIE2` musi być większe niż `WYRAŻENIE1` - iteracja w górę. W przypadku gdy chcemy aby w kolejnych obiegach pętli wartość zmiennej zmniejszała się stosujemy konstrukcję:

```
for ZMIENNA := WYRAŻENIE1 downto WYRAŻENIE2 do INSTRUKCJA;
```

Przykłady użycia przedstawionych pętli zaprezentowano na listingu 5.1

*Listing 5.1. Przykładowe użycia pętli*

```
program prog5;

uses
      crt; {Import biblioteki zawierającej funkcje Readkey}
var
      i,j,foo : Integer;

begin
      foo := 10;
      for i:= 1 to 50 do Write(i);
      while foo >= 5 do
      begin
            foo := foo - 1;
            Writeln(foo);
      end;
      Readkey; {Oczekiwanie na wciśnięcie dowolnego klawisza}
end.
```

## 6. Operacje we/wy – interakcja z użytkownikiem

W poprzednim punkcie używana była instrukcja `Write()` umożliwiająca wyświetlanie użytkownikowi na ekranie komputera pewnych informacji. Jest to jedna z instrukcji służących

do interakcji programu z użytkownikiem. Kolejną instrukcją która pozwala wyświetlać dane na ekranie jest instrukcja `Writeln()`. Rożnica między tymi dwoma instrukcjami jest taka że w przypadku tej drugiej po wypisaniu danych na ekranie kursor przechodzi do nowej linii. Przykład użycia instrukcji wypisujących dane na ekranie przedstawiono poniżej.

*Listing 6.1. Przykładowe użycia instrukcji Write() oraz Writeln()*

```
program prog61;

uses
      crt;
var
      foo, foobar : Integer;
      bar : String;

begin
      foo := 2;
      foobar := 3;

      Writeln('Wartosc zmiennej foo to ', foo);
      Writeln(foo - 4);
      Writeln('foo = ', foo, '  foobar = ', foobar);  { ** 1 **}
      Writeln('Lorem ipsum');
      Writeln('dolor');
      Write('sit');
      Write(' amet,');
      Readkey;
end.
```

Na listingu powyżej w linii oznaczonej jako `{ ** 1 **}` pokazano przykład formatowania ciągu znakowego z wykorzystaniem zmiennych.

Do wczytywania danych z klawiatury służy w języku Pascal funkcja `Read()`. Instrukcja ta pobiera jako parametr zmienną do której wartość wprowadzona z klawiatury ma zostać przypisana. Funkcja może pobierać większa liczbę argumentów jeżeli ma zostać wczytana większa liczba danych.

*Listing 6.2. Przykładowe użycia instrukcji Write() oraz Writeln()*

```
program prog62;

uses
      crt;
var
      foo : Integer;

begin
      Writeln('Podaj liczbe:');
      Read(foo);
```

```
      Writeln('Potega liczby ', foo, ' to: ', foo * foo);
      Readkey;
end.
```

## Zadania do wykonania

### Zadanie 1

Uruchomić i przeanalizować przykłady z instrukcji zawarte na listingach 5.1, 6.1 oraz 6.2

### Zadanie 2

Napisać program który wyświetli na ekranie tekst jak poniżej.

```
**********************************************
**                                          **
**                HELLO WORLD               **
**                                          **
**********************************************
```

### Zadanie 3

Napisać program który sprawdzi czy podana przez użytkownika liczba jest parzysta.

### Zadanie 4

Napisać program który wyznaczy wartość liczby π według zależności poniżej:

$$\sum_{i=0}^{\infty} \frac{(-1)^i}{2i+1} = \frac{\pi}{4}$$

Dokładność obliczenia (liczba kroków iteracji) powinna być podawana przez użytkownika. Do operacji potęgowania wykorzystać funkcję `Power()` z biblioteki `math`.

### Zadanie 5

Napisać program wyznaczający średnią arytmetyczną liczb wprowadzonych przez użytkownika. Ilość liczb powinna być wprowadzana z klawiatury przez użytkownika.

### Zadanie 6

Rozbudować program z zadania 5 tak, aby dla wprowadzonych liczb wyznaczał:

- wartość maksymalną,
- wartość minimalną.