

6 Instrukcje

Instrukcja pusta.

Instrukcja przypisania .

`zmienna := wyrażenie ;`

Instrukcja procedury .

`Nazwa ; lub Nazwa (parametry) ;`

6.1 Instrukcje wejścia /wyjścia (Input/ Output)

6.1.1 Wyjście - procedury Write , Writeln

Instrukcja procedury `write` (`writeln`) ma ogólną postać :

`write (f , p1 , p2 , . . . , pn) ;`

`writeln (f , p1 , p2 , . . . , pn) ;`

gdzie `f` -oznacza plik dyskowy (parametr ten może być pominięty wtedy dane wypisywane są na konsolę , czyli ekran monitora)

natomiast każdy z `pi` może występować w postaci

p **p:m** **p:m:n**

p jest typu prostego lub `string` ;

m określa szerokość pola w jakim dane będą wypisywane ;

n określa ilość miejsc po przecinku (tylko typy rzeczywiste)

n, m - wyrażenia całkowite

Program Wypisz ;

`var l : Byte;`

`i : Integer;`

`r : Real;`

`begin`

`i:=2;`

`l:=7;`

`r:= l / i ;`

`write (1:4);`

`write (l : 4);`

`write(r:5:2);`

`end.`

Jaki będzie wynik wykonania programu ?

Co się stanie gdy zmienimy instrukcję na :

```
write(r:1:2);
write(r:4:1);
```

Procedury `write` i `writeln` mogą być wywoływane ze zmienną ilością parametrów . Ponadto parametry mogą mieć różne typy .

6.1.2 Wejście - procedury `Read` , `ReadLn`

Instrukcja procedury `read` (`readln`) ma ogólną postać :

```
read (f , p1 , p2 , . . . , pn);
readln (f , p1 , p2 , . . . , pn);
```

gdzie `f` mają takie same znaczenie jak w przypadku procedury `Write` , natomiast każde `pi` jest typu całkowitego , rzeczywistego , `char` lub `string` ;

Przykład 1:

```
Program Czytaj_1;
var i : Integer;
begin
  Read (i);
  writeln;
  writeln(i);
end.
```

Efektom wykonania powyższego programu będzie

1. wczytanie z konsoli (klawiatura) liczby całkowitej
2. następnie liczba ta zostanie ponownie wyświetlona
- wypisana będzie wartość zmiennej „ i ”

Przykład 2:

```
Program Czytaj_2;
var s1,s2 :string;
```

```
begin
  Readln(s1);
  writeln;
  writeln(s1);
  Readln(s2);
  writeln;
  writeln(s1,s2);
end.
```

```
Program Czytaj_2_b;
var s1,s2 :string;
```

```
begin
  Read(s1);
  writeln;
  writeln(s1);
  Read(s2);
  writeln;
  writeln(s1,s2);
end.
```

Jako ćwiczenie na laboratorium pozostawiam sprawdzenie czy istnieją różnice w wykonaniu tych dwóch wersji programów . Jeśli tak , to co może tłumaczyć takie różnice .

Procedura Readln może również zawierać większą ilość parametrów - może też ich wogóle nie być. Taki przypadek będzie bardzo użyteczny w celu zatrzymania wykonywania programu .

```
Program Stopped;  
begin  
  ...  
  readln;  
end.
```

6.2 Instrukcje strukturalne

Poznane jak dotąd instrukcje były instrukcjami prostymi - nie zawierały w swej składni innych instrukcji . Natomiast instrukcje strukturalne są konstrukcjami językowymi zbudowanymi z ciągów instrukcji na podstawie określonych schematów strukturyzacji .

6.2.1 Instrukcje złożone

Instrukcja złożona ma następującą budowę :

```
BEGIN  
  I1; I2 ; ...In  
END
```

gdzie I1; I2 ; ...In są instrukcjami a słowa kluczowe BEGIN i END reprezentują tzw. nawiasy instrukcyjne .

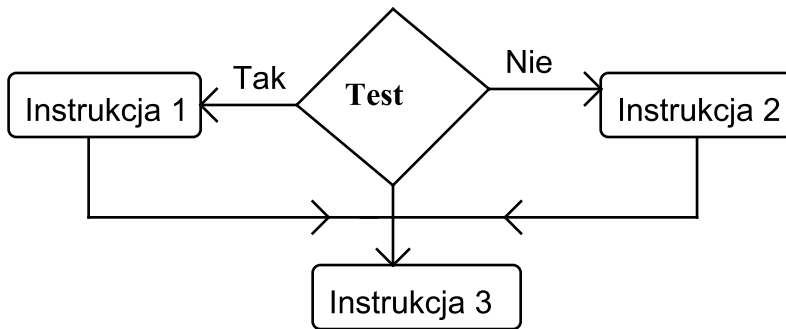
Instrukcja złożona zastępuje więc ciąg instrukcji , a właściwie to zamienia ciąg instrukcji w jedną instrukcję .Takie podejście jest bardzo użyteczne , gdy musimy użyć wielu instrukcji natomiast składnia dopuszcza tylko jedną .

Separatorem instrukcji jest średnik „;” . występuje on pomiędzy każdymi dwoma instrukcjami . Możemy go również postawić po ostatniej instrukcji , wtedy oddzieli on tą instrukcję od następnej - tym przypadku będzie to instrukcja pusta .

6.2.2 Instrukcja warunkowa

Instrukcja wyboru warunkowego :

```
IF warunek THEN instrukcja_1
  ELSE instrukcja_2
```



Instrukcja ta jest równoważna schematowi . W schemacie może być opuszczona instrukcja 2 , a w składni może być opuszczona fraza ELSE .

```
IF warunek THEN instrukcja_1
```

Warunek musi być wyrażeniem logicznym(Boolean), czyli dającym w wyniku wartość boolowską: True lub FALSE.

Natomiast *Instrukcja1* i *Instrukcja2* muszą być pojedynczymi instrukcjami.

Przykłady :

- (a) IF X<>0 THEN y := 1/x ELSE write('Dzilenie przez 0');
- (b) IF (X>0) AND(X=1) THEN ELSE y:= 1/x;
- (c) IF B < 0 THEN y:= B - 3 ELSE ;

ELSE w przykładzie (c) może być pominięte , w przykładzie (b) można go pominąć zmieniając warunek na przeciwny czyli

- (b`) IF NOT ((X>0) AND(X=1)) THEN
y:= 1/x;

```
(d) IF delta > 0 then
  begin
    x1:= (-b - Sqrt(delta)) / (2*a);
    x2:= (-b - Sqrt(delta)) / (2*a);
  end
ELSE
  begin
    x1:= -b / (2*a);
    x2:=x1;
  end;
```

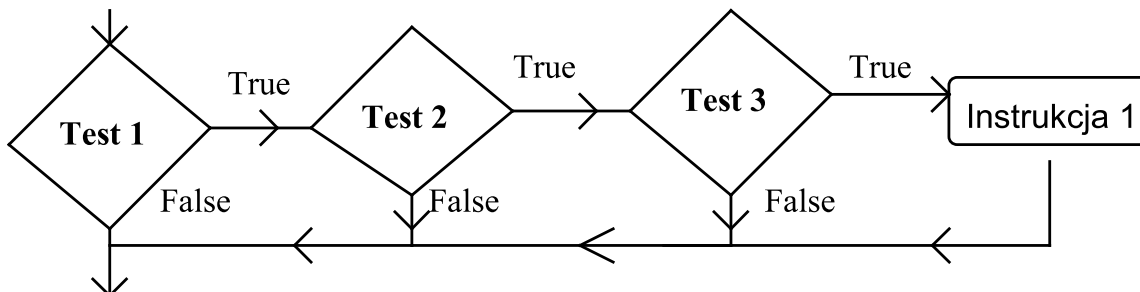
Ponieważ w składni instrukcji warunkowej występują pojedyncze instrukcje , a my musieliśmy użyć ich ciągu zastosowaliśmy instrukcje złożone , które zamieniło nam ciągu instrukcji w pojedyncze instrukcje .

Instrukcja występująca po THEN czy też po ELSE może być również instrukcją warunkową IF

Np

```
IF test1 THEN
  IF test2 THEN
    IF test3 THEN
      Instrukcja1
    ELSE
  ELSE
ELSE
```

co uwidacznia następujący schemat blokowy



```
(e)IF delta > 0 then
  begin
    x1:= (-b - Sqrt(delta))/(2*a);
    x2:= (-b + Sqrt(delta))/(2*a);
  end
ELSE IF delta = 0
  begin
    x1:= -b / (2*a);
    x2:=x1;
  end
ELSE
  Write('Nie ma pierwiastków rzeczywistych');
```

Przykład:

```
program oceny;
var ocena:real;
    wynik:integer;
Begin
wynik:=90;
IF wynik >=30 then
  IF wynik >=35 then
    IF wynik >=40 then
      IF wynik >=50 then
        IF wynik >=60 then
          IF wynik >=70 then
            IF wynik >=80 then
              IF wynik >=90 then
                Ocena:=5.0
              ELSE Ocena:=4.5
            ELSE Ocena:=4.0
          ELSE Ocena:=3.5
        ELSE Ocena:=3.0
      ELSE Ocena:=2.5
    ELSE Ocena:=2.0
  ELSE Ocena:=1.5
ELSE Ocena:=1.0;
writeln(ocena:4:1);
End.
```

Program ten przyporządkuje wynikowi z zakresu 1 - 100 punktów odpowiednią ocenę. Punkty określają procent odpowiedzi poprawnych. Układ konstrukcji IF...THEN ...ELSE pokazuje nam, że program ten składa się z wielu zagnieżdżonych instrukcji warunkowych.

```

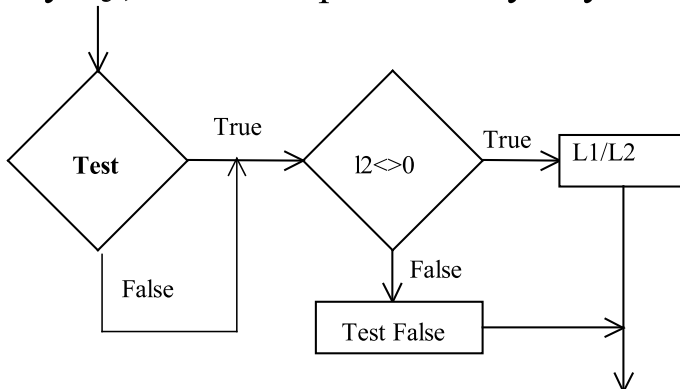
program ifelse;
var test,test2:boolean;
    l1,l2:integer;
Begin
    test:=25*26<27*24;
    IF test then
        IF l2<>0 then
            writeln(l1/l2)
        ELSE writeln('Test
False');
End.

```

Poddamy dogłębnej analizie przytoczony przykład programu. Wyrażenie widzimy tu dwie instrukcje warunkowe . Jednak odnajdujemy tylko jedno słowo ELSE , co oznacza , że jedna instrukcja jest postaci IF...THEN...ELSE natomiast druga IF...THEN...

Należy postawić sobie pytanie , do którego IF odnosi się to ELSE ?

Myślę , że bardzo pomocne byłoby tu stworzenie schematu blokowego



Na jego podstawie widzimy wyraźnie , sposób wykonywania naszej instrukcji warunkowej .

6.2.3 Instrukcja wyboru CASE

Składnia instrukcji CASE jest następująca

CASE selektor **OF**

pierwsza lista stałych wyboru : I1;

druga lista stałych wyboru : I2;

n-ta lista stałych wyboru : In

ELSE Ie;

END ;

gdzie $I1..In$, Ie są instrukcjami prostymi lub złożonymi, selektor jest wyrażeniem typu porządkowego (*Integer*, *Char*, *Boolean* ale NIE *Real*).

Listy stałych są albo wymienionymi wartościami

np 1,2,57

albo przedziałami np 1..9

albo połączeniem np 0..9,11,12,17..35

Przykład 1:

```
Case wynik DIV 10 of
  10,9      : Ocena:=5.0
  8         : Ocena:=4.5
  7         : Ocena:=4.0
  6         : Ocena:=3.5
  5         : Ocena:=3.0
  4         : Ocena:=2.5
  3         : Ocena:=2.0
  0,1,2     : Ocena:=1.0
End;
```

Przykład 2

```
program Wybor;
var droga : char;
    czas : real;
begin
  Read(Droga);
  Case droga of
    'A','a' : czas:=3.0;
    'B','b' : czas
:=4.0;
    'C','c' : czas
:=4.5;
    ELSE czas :=0;
  end;
end.
```

Instrukcja *CASE* w wielu przypadkach jest bardziej czytelna niż połączenie wielu zagnieżdżonych instrukcji *IF*. Dobrym przykładem może tu być wyliczanie ocen.

Zarówno w każdej z instrukcji *IF* jak i w instrukcji *CASE* mogą być opuszczone fragmenty zawierające instrukcje wykonywane w przypadku niespełnienia warunku (w *CASE* żadnego z warunków).

W instrukcji *CASE* *ELSE* może nie występować w niektórych implementacjach Pascala, standard Pascala jej nie zawiera - jest to jednak poprawna konstrukcja Turbo Pascala

```
Program realCase;
var x:real; a:integer;s:string;
begin
```

```
read(a);  
x:=sqrt(a)/a;  
CASE x OF  
  1: s:='Liczba dodatnia';  
 -1: s:='Liczba ujemna';  
  0: s:='Liczba 0'  
end;  
writeln(s);  
end.
```

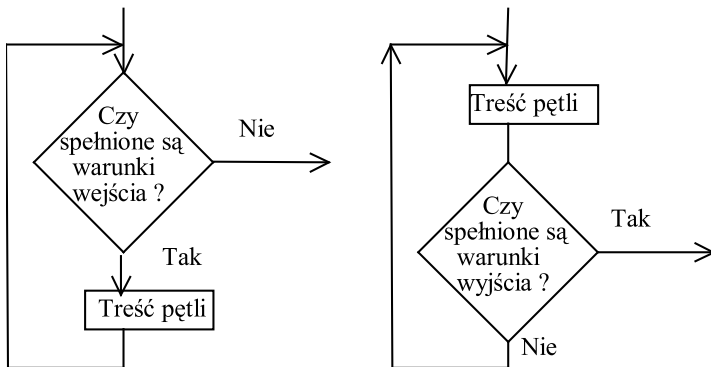
Czy ten program jest poprawnie zbudowany . Widzimy tu błędne użycie instrukcji CASE . Nie jest dozwolone , aby selektor był wyrażeniem (zmienna czy też stałą) rzeczywistym .

Należy w instrukcji CASE wyrażenie x zastąpić $\text{Trunc}(x)$. Wynikiem tej funkcji jest część całkowita z liczby x .

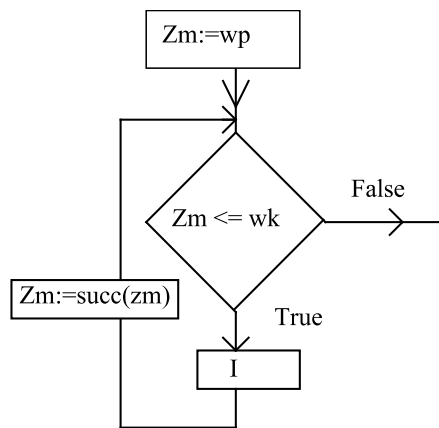
Program nie będzie działał poprawnie , gdy podamy liczbę 0.

6.3 Instrukcje iteracyjne

Iteracje (pętle) pozwalają na opisaniu w prosty sposób czynności które są wykonywane wielokrotnie . W języku Pascal istnieją dwa rodzaje pętli



6.3.1 Instrukcja FOR



Składnia instrukcji :

FOR $zm:=wp$ TO wk DO I ;

gdzie

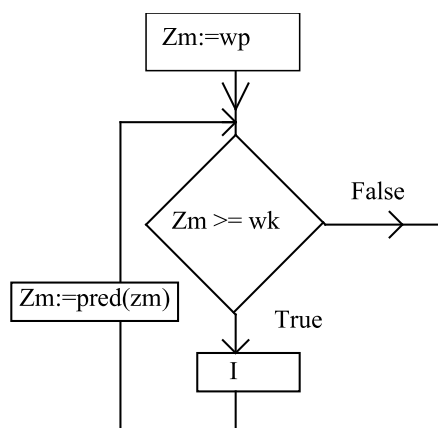
zm - Zmienna sterująca pętlą

wp - wartość początkowa

wk - wartość końcowa

I - pojedyncza instrukcja

Instrukcja OR jest realizacją pętli z warunkiem wejścia . Jest ona pewnym wariantem pętli WHILE.



Składnia instrukcji :

FOR $zm:=wp$ DOWNTO wk DO I ;

gdzie

zm - Zmienna sterująca pętlą

wp - wartość początkowa

wk - wartość końcowa

I - pojedyncza instrukcja

Pętla FOR to przykład iteracji ograniczonej – z góry wiadomo ile razy wykona się dana pętla.

UWAGA!

W treści pętli nie wolno zmieniać wartości zmiennej sterującej pętlą.

Przykład:

```
var zn:char;k:Integer;
begin
  zn:='*';
  for k:=1 to 8 do
    write(zn:k);
end.
```

PROGRAM Reprezentacja;

```
Uses Crt;
VAR   znak : Char;
BEGIN
  Writeln('Program drukuje kody ASCII znakow');
  FOR znak:='0' TO '9' DO
    Writeln('cyfra ',znak,' ma kod ',Ord(znak));
  FOR znak:='A' TO 'Z' DO
    Writeln('litera ',znak,' ma kod ',Ord(znak));
  REPEAT UNTIL Keypressed;
END.
```

program srednia;

```
uses crt;
var
i,ilosc,suma,a:integer;
wynik:real;
begin
  clrscr;
  writeln('Program oblicza srednia arytmetyczna');
  write('Podaj ilosc liczb z ktorych oblicze
srednia: ');
  suma:=0;{przypisanie wartosci poczatkowej sumie}
  readln(ilosc);
  for i:=1 to ilosc do
    begin
      write('Podaj liczbe ',i,' a',i,'= ');
      readln(a);
      suma:=suma+a;
    end;
  wynik:=suma/ilosc;
  writeln('Srednia z podanych liczb wynosi ' ,
  wynik:2:2);
  writeln('Nacisnij klawisz ENTER');
  readln;
end.
```

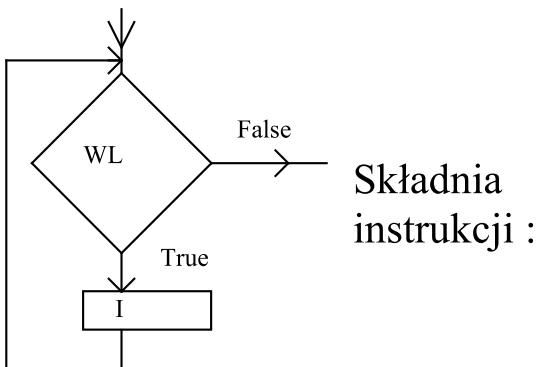
```
PROGRAM Suma_liczb_parzystych;
USES  Crt;
CONST  n=100 ;
VAR
    suma,licznik : Integer;

BEGIN
    ClrScr;
    Writeln('Program sumuje liczby parzyste od 1
do ',n);
    suma:=0;

    FOR licznik:=1 to n DO
        IF NOT Odd(licznik) THEN suma:=suma+licznik;

    Writeln;
    Writeln('Suma liczb przystych od 1 do ', n, '
        wynosi ', suma);
    ReadLn
END.
```

6.3.2 Instrukcja WHILE



WHILE *wl* DO *I*;

gdzie

wl - wyrażenie logiczne

I - pojedyncza instrukcja

Instrukcja WHILE jest realizacją pętli z warunkiem wejścia. Instrukcja oznacza: *Dopóki prawdziwe jest wyrażenie WL wykonuj instrukcję I.*

Przykłady :

```
Program Gwazdka;
var zn:char;k:Integer;
begin
  zn:='*';
  k:=1;
  while k<=8 do
  begin
    write(zn:k);
    k:=k+1;
  end;
```

```
Program Liczby;
uses crt;
begin
  i:byte;
  i:=0;
  while i<10
  do
  begin
    i:=i+1;
    writeln(i);
  end;
end.
```

xxx

```

PROGRAM Petla2;
USES Crt;
VAR i, x, y, Nwd : Integer;
  BEGIN
  ClrScr;
  Write('Podaj 1-sza liczbe : ');
  ReadLn(x);
  Write('Podaj 2-ga liczbe : ');
  ReadLn(y);
  i:=1;
  WHILE (x>i) Or (y>i) DO
    Begin
      If (x Mod i=0) And (y Mod i=0) Then Nwd:=i;
      i:=i+1;
    End;
  Write('Najwiekszy wspolny dzielnik = ', Nwd);
  ReadKey;
END.

```

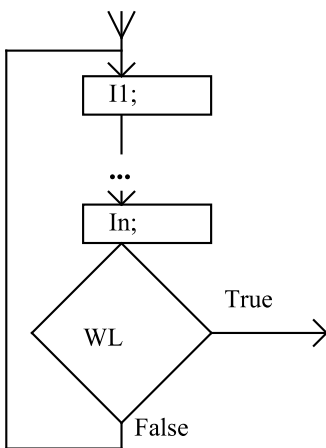
6.3.3 Instrukcja Repeat

Składnia instrukcji :

```

REPEAT I1; ... In UNTIL wl;
gdzie
wl   - wyrażenie logiczne
I1, ..., In   - instrukcje

```



Instrukcja REPEAT jest realizacją pętli z warunkiem wyjścia. Instrukcja oznacza: *Wykonuj instrukcję I1,...,In aż prawdziwe będzie wyrażenie WL*. Dowolna pętla zrealizowana za pomocą instrukcji WHILE może być zbudowana za pomocą instrukcji REPEAT. Twierdzenie odwrotne nie jest prawdziwe .

Przykład :

```
Read(zn);
WHILE zn <> '*' do
  begin
    write(zn);
    read(zn);
  end;
writeln;
REPEAT zn <> '*'
  read(zn);
  if zn <> '*' then
    write(zn);
UNTIL zn = '*'
writeln

Uses CRT;
Var
  licz:byte;

Begin
  ClrScr;
  REPEAT
    Write('Wpisz jakąś
  liczbę (0-wyjście) :
  ');
    ReadLn(licz);
    WriteLn('Potega
  liczby ',licz,' to
  ',licz*licz);
    WriteLn;
  UNTIL licz=0;
End.
```

Przykład:

```
PROGRAM Petla3;
USES Crt;
VAR n, x : Integer;
    y     : Real;
    Zn    : Char;
BEGIN
  ClrScr;
  Write('Podaj X : ');
  ReadLn(x);
  y:=1;
  n:=1;
  REPEAT
    Write('N = ', n:3);
    y:=x/n;
    WriteLn('  X/N = ', y:7:2);
    n:=n+1;
  UNTIL y<0.2;
Repeat Until KeyPressed
END.
```