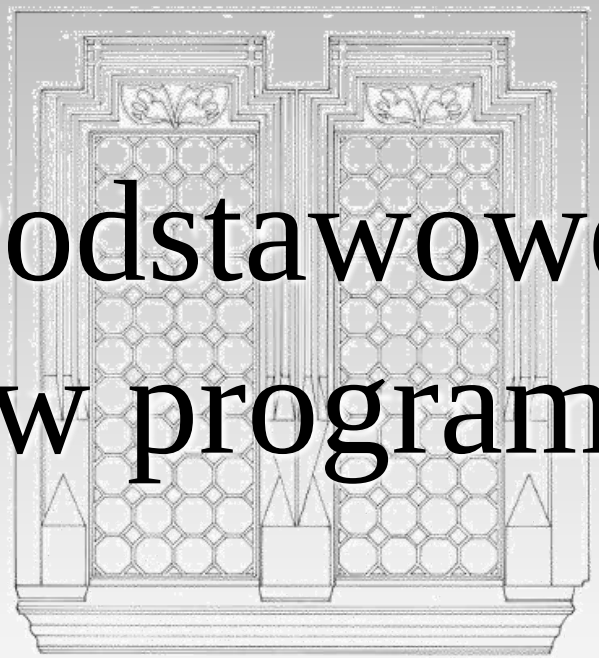
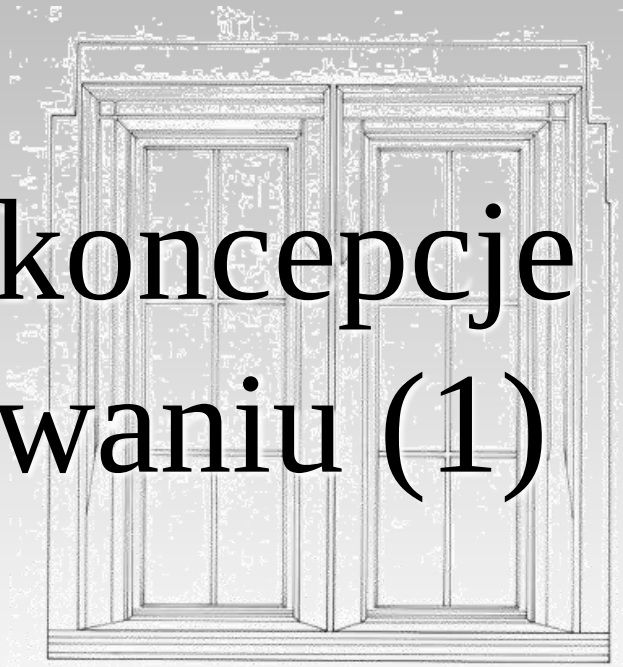


Podstawowe koncepcje w programowaniu (1)



Iteracja oznacza powtarzanie dowolnej instrukcji pod pewnymi warunkami. Instrukcja ta jest w danym miejscu programu zapisana tylko raz.

Jeśli obliczenie y = x^5 („x do potęgi 5”) zapisane sekwencyjnie:
y:= 1; y:= y*x; y:= y*x; y:= y*x; y:= y*x; y:= y*x;
chcemy zapisać "skrótowo”, to powinniśmy zastosować instrukcję iteracyjną.

Iteracja nieograniczona (warunkowa)

Pętla "while … do ..." („dopóki ... wykonuj ...”)
(*najważniejsza, ogólna konstrukcja iteracyjna w językach programowania*)

while warunek **do** dowolna_instrukcja

// zakończenie, gdy warunek nie jest spełniony
// w tej pętli warunek jest testowany przed instrukcją

Iteracja nieograniczona (warunkowa)

PRZYKŁAD

```
a := 24;
b := 6;
while   a>= b  do  a := a – b;

// wartość końcowa a = 0
// liczba iteracji (czyli wykonań instrukcji wewnątrz pętli) : 4
// a jeśli na początku a := 17,  b := 6  to jakie wartości na końcu ?
```

Iteracja nieograniczona (warunkowa)

PRZYKŁAD // oblicz x do potęgi 5

```
read(x);   // wczytaj wartość zmiennej   x
y := 1;
i := 1;
while   i <= 5  do
    begin
    y := y * x;
    i := i + 1
    end;     // nawiasy syntaktyczne konieczne !

// teraz, ponieważ pętla się zakończyła, MUSI zachodzić  i > 5
```

Iteracja nieograniczona (warunkowa)

PRZYKŁAD // przepisywanie danych (wczytaj i drukuj)
// aż do momentu gdy kolejna wczytana liczba ma wartość –1

```
read (x);
while   x ≠ -1 do
begin
    write (x);    // wypisz wartość zmiennej x
            read (x);     // wczytaj kolejną wartość pod zmienną x
end;
writeln ('koniec'); // wypisz komunikat i przejdź do nowego
                    // wiersza
```

// końcowa liczba –1 nie została wypisana.

Iteracja nieograniczona (warunkowa)

PRZYKŁAD // zsumuj dane w pliku (algorytm zapisany słownie w poprzednim wykładzie)

```
suma := 0;
//ustaw pozycję czytania na początku pliku
while nie-koniec-pliku do
begin
    read(dana);
    suma := suma + dana;
end;
write (suma);
```

Algorytm Euklidesa
obliczania największego wspólnego dzielnika liczb a i b (NWP(a,b))

Poszerzamy zbiór operatorów arytmetycznych o 2 działania:
-operator **mod** : obliczanie reszty z dzielenia
np. 17 **mod** 5 = 2, 17 **mod** 17= 0, 16 **mod** 17 = 16
-operator **div** : dzielenie całkowite (obcięcie reszty)
np. 17 **div** 6 =2, 17 **div** 16 = 1, 16 **div** 17 = 0

Algorytm Euklidesa
obliczania największego wspólnego dzielnika liczb a i b (NWP(a,b))

```
 read(a , b);     // zakładamy, że a, b są liczbami naturalnymi
 x := a;
 y := b;
while   x * y > 0 do
if  x > y  then  x := x mod y
            else  y := y mod x ;
nwp := x + y ;
```

Przeprowadź test dla : a=432, b=126

Jeśli operator **mod** nie jest dostępny, to można zastosować jedno z dwu rozwiązań:

1. Zakładając, że istnieje operator **div** wykorzystać go do obliczania reszty z dzielenia.
2. W ogóle nie wykonywać dzielenia, posłużyć się odejmowaniem.

Iteracja nieograniczona (warunkowa)

Pętla **”repeat ... until ...”** („powtarzaj ... aż do ...”)

repeat instrukcja **until** warunek

// zakończenie gdy warunek jest spełniony
// w tej pętli warunek jest testowany po instrukcji

Iteracja nieograniczona (warunkowa)

PRZYKŁAD // test podzielności przez 13, bez operacji dzielenia

```
  n := 222001;
  s := n;
  repeat
s := s - 13
until  s <= 0;

if  s = 0 then wynik := ‘liczba podzielna przez 13’
          else  wynik := ‘liczba niepodzielna przez 13’
```

Iteracja nieograniczona (warunkowa)

PYTANIE

Jak posługując się pętlą **while** uzyskać efekt identyczny jak przy użyciu pętli **repeat** ? A na odwrót ?

Iteracja ograniczona

Pętla „for” („dla ”)

postać ogólna (wersja „rosnąco-do” / „**to”/)**:

for zmienna_sterująca **:=** wartość_początkowa
to wartość_końcowa
do instrukcja

Zakładamy, że wartości zmiennej są całkowite.

Iteracja ograniczona

Znaczenie pętli **for** wyrażone przy użyciu pętli **while**:

```
zmienna_sterująca := wartość_początkowa;
while   zmienna_sterująca <= wartość_końcowa do
     begin
          dowolna_instrukcja;
          zmienna_sterująca := zmienna_sterująca + 1
    end;
```

Iteracja ograniczona

UWAGI

Jeśli wartość początkowa zmiennej sterującej jest większa od wartości końcowej to iteracja przebiegnie 0 razy, w przeciwnym razie przebiegnie (wartość końcowa – wartość początkowa + 1) razy.

Konstrukcja **for** jest wygodna, gdy w momencie rozpoczęcia iteracji wiadomo z góry ile będzie powtórzeń.

Analogicznie zachowuje się wersja pętli **for** „malejąco-do” / „**downto**”/.

Iteracja ograniczona

PRZYKŁAD // oblicz x do potęgi 5

```
read( x); y := 1;
for  i := 1  to 5 do
begin
     y := y * x;
  write ( y)
end;
```

Iteracja ograniczona

PRZYKŁAD // wypisz 100 liczb z zakresu od 151 do 250

```
for  i := 151 to  250  do writeln (i);

// wewnątrz pętli można wykorzystywać zmienną sterującą !
```

ZASADA
Dobry programista nie zmienia wartości zmiennej sterującej wewnątrz iteracji.