INSTRUKCJE ITERACYJNE

FOR – do działania potrzebuje licznik, który będzie zwiększany do zadanej wielkości o jeden.

For *licznik:=wart_pocz* ***to*** *wart_koncowa* ***do*** *lista_polecen*

Przykład:

```
Var
        Licznik:integer;
Begin
        For Licznik:=0 to 10 do
                Writeln('Wartosc licznika: ',Licznik);
End.
```

Na ekranie pojawi się:
Wartosc licznika: 0
Wartosc licznika: 1
…
Wartosc licznika: 10

FAKT – jak widać pętla For zwiększa licznik. Czasem może być jednak potrzeba by w pętli licznik był zmniejszany. Rozwiązaniem jest pętla For w nieco innej „odmianie":

For *licznik:=wart_pocz* ***downto*** *wart_kon* ***do*** *lista_polecen*

Różnica polega na zastosowaniu słowa kluczowego **DOWNTO** zamiast **TO**. W tym wypadku wartość początkowa musi być większa od wartości końcowej, ponieważ licznik będzie w trakcie iteracji zmniejszany, a nie tak jak poprzednio zwiększany.

Przykład:

```
Var
        Licznik:integer;
Begin
        For Licznik:=10 downto 0 do
                Writeln('Wartosc licznika: ',Licznik);
End.
```

Na ekranie pojawi się:
Wartosc licznika: 10
Wartosc licznika: 9
…
Wartosc licznika: 0

DO DYSKUSJI: Zastanów się co było by gdyby w przypadku pętli FOR..TO..DO wartość początkowa była na starcie większa od wartości końcowej, oraz w przypadku pętli FOR..DOWNTO..DO wartość początkowa jest startowo mniejsza od wartości końcowej.

PODSUMOWANIE:
1.Mamy do dyspozycji dwie odmiany instrukcji FOR: iterująca w górę i w dół.
2.Do działania pętli FOR konieczny jest licznik – zmienna typu całkowitego.
3.Licznik zwiększa/zmniejsza się automatycznie o 1 – zatem nie trzeba już samemu o to dbać.
4.Pętla FOR sprawdzi się wszędzie tam, gdzie wiemy, że pewną czynność chcemy wykonać X-razy.

Zadania dla FOR:

Zad1.Napisz program, który z pomocą pętli for pobiera od użytkownika N-liczb typu całkowitego i sumuje je. Po pobraniu wszystkich liczb drukuje wartość sumy. N użytkownik wprowadza na klawiaturze.

Zad2.Napisz program, który pobiera od użytkownika N-liczb typu rzeczywistego i wylicza ich średnią arytmetyczną.

Zad3.Napisz program, który pobiera od użytkownika dwie liczby całkowite A i B, a następnie wylicza sumę liczb z przedziału <A,B>.

Zad4.Napisz program, który pobiera od użytkownika wartość liczby A oraz całkowitą liczbę N (N>=0). Dalej program podnosi liczbę A do potęgi N. Na końcu program ma wyświetlać wynik potęgowania.

Pętla z warunkiem sprawdzanym po wykonaniu poleceń (repeat..until)

Pętla ta jest wykonywana tak długo aż nie zostanie spełniony warunek**.**

Bywa, że chcemy powtarzać pewną czynność, ale nie jesteśmy w stanie określić ile razy.
W przypadku pętli FOR należało zdefiniować, że pewną czynność powtarzamy X razy zaczynając od Z a kończąc na Y (Y – Z = X). Jeśli jednak pewną czynność będziemy powtarzać do czasu, kiedy spełnimy pewne kryterium (warunek), jednak nie da się przewidzieć, czy do jego osiągnięcia wystarczy 10, 20, a może 1000 iteracji – wówczas z pomocą przychodzi nam pętla REPEAT..UNTIL.

repeat *lista_polecen* ***until*** *warunek*

Przykład:

```
Var
        i,X:integer;
Begin
        Writeln('Podaj x:');
        Readln(X);
        i:=0;
        Repeat
                X:=X+X;
                i:=i+1;
        Until X>100;
        Writeln('Pęla kręsiła się ',I,' razy');
End.
```

Na ekranie pojawi się:
Podaj x: 10 {wpradzam na klawiaturze wartość 10}
Pętla kręciła się 4 razy

JAK WIDAĆ PĘTLA WYKONAŁA 4 ITERACJE. W Momocie kiedy warunek został spełniony pętla zakończyła iteracje. Gdybym na podatku za X wprowadził inną wartość, pętla wykonała by Iną liczbę iteracji.

Jednak gdybym od razu wprowadził za X wartość spełniającą warunek końca pętla i tak wykonała by jedną iterację co wynika z faktu, że najpierw wykonywana jest iteracja a dopiero potem sprawdzany jest warunek.

Przykład:
Podaj x: 101 {wpradzam na klawiaturze wartość 101}
Pętla kręciła się 1 razy

PODSUMOWANIE:
1.Pętla REPEAT..UNTIL wykona się przynajmniej raz, bez względu na to czy warunek końca jest spełniano od samego początku czy nie.
2.Pętla ta jest przydatna wszędzie tam, gdzie nie da się z góry przewidzieć ilości iteracji.
3.Spełnienie warunku zdefiniowanego po słowie kluczowym UNTIL powoduje przerwanie działania pętli.

Pętla z warunkiem na początku.

Pętla ta jest wykonywana w przypadku, gdy zostanie spełniony warunek.

Jest to pętla, która działa podobnie jak pętla REPEAT..UNTIL. Nie wymaga ona określania ilości iteracji jak miało to miejsce w przypadku pętli FOR.
Jednak różni się ona od pętli REPEAT..UNTIL tym, że tamta pętla działała do momentu, gdy warunek określony pętli nie został spełniony. Tutaj warunek musi być spełniony aby pętla działała – jeśli warunek nie będzie spełniony pętla przestaje się wykonywać.. Dodatkowa różnica polega na tym, że w przypadku pętli REPEAT..UNTIL jedna iteracja zawsze miała miejsce (bez względu na warunek końca) – tutaj, jeśli warunek konieczny do wykonywania się pętli nie będzie spełniony od samego początku pętla nie wykona się ani razu.

while *warunek* ***do*** *lista-polecen*

Przykład:

```
Var
        I:integer;
Begin
        Writeln('Podaj i');
        Readln(I);
        While i > 0 do
                I:=I – 5;
        Readln;
End.
```

<u>Na ekranie pojawi się:</u>
Podaj i: -1   {wpradzam na klawiaturze wartość -1}

!!!!!!!!!!!!!!! PĘTLA NIE WYKONA ŻADNEJ ITERACJI PONIEWAŻ NA STARCIE I NIE SPEŁNIA WARUNKU KONIECZNEGO DODZIAŁANIA PĘTLI!!!

Podsumowanie:
1.Pętla ma podobne działanie i zastosowanie jak pętla REPEAT..UNTIL
2.Spełnienie warunku w przypadku tej pętli konieczne jest by mogła ona działać – w wypadku REPEAT..UNTIL spełnienie warunku powodowało jej zakończenie.
3.W przypadku tej pętli jeśli warunek nie będzie spełniony od początku pętla nie wykona się ani razu.

Zadania dla pozostałych pętli:

Zad1.Napisz program, który pobiera od użytkownika dwie wartości całkowite P i K. Następnie generuje kolejne liczby całkowite zaczynając od P, do momentu, aż ich suma nie przekroczy wartości K.

Zad2.Napisz program, który pobiera od użytkownika znak (CHAR), do momentu, aż użytkownik nie poda znaku kończącego wykonanie pętli (Znak końca zdefiniowany w programie jako stała).

NP.:
ZNAK_KOŃCA='x'

Użytkownik jest proszony o podanie znaku tyle razy aż nie poda 'x' – wówczas pojawia się informacja, że wybrał właściwy znak i program Si kończy.

Zad3.Napisz dwa dowolne programy – w jednym wykorzystaj pętle REPEAT..UNTIL, a w drugim WHILE..DO.

C.D.N.

Zadanie dodatkowe (będzie realizowane na zajęciach, jednak warto się nad nim zastanowić już w domu).

Napisz program realizujący algorytm CHOINKA: użytkownik podaje wartość N, a następnie na ekranie pojawia się choinka o N poziomach.

N = 3;

```
*
**
***
```

WERSJA BARDZIEJ ZAAWANSOWANA (z wcięciami)
N = 3;

```
   *
  * *
* * *
```