Pętle (ang. loop) to konstrukcje językowe służące do zdefiniowania szeregu instrukcji, które będą powtarzane wielokrotnie. Poznamy trzy podstawowe rodzaje pętli: for, while oraz do .. while.

Każda z pętli nadaje się do stworzenia dowolnej iteracji, niemniej jednak niektóre pętle sprawdzają się lepiej przy określonych problemach.

Pętla for

Przeważnie używana w sytuacji, gdy przed uruchomieniem pętli znamy ilość iteracji (powtórzeń)

Składnia

Pętla for składa się z 3 pól:

```
for(pole1; pole2; pole3){
…kod…
}
```

czyli

```
for (inicjalizacja; warunek_pętli; krok_pętli){
…kod_do_powtórzenia…
}
```

pole1 – inicjalizacja pętli, w tym miejscu tworzymy zmienną odpowiadającą za odliczanie ilości przebiegów pętli i nadajemy jej wartość początkową. Ta instrukcja wykonuje się tylko raz.

Przykłady:

```
for (int i=0;
for (int j=10;
for (int a=b;
```

pole2 – warunek pętli. Warunek jest sprawdzany przed uruchomieniem pętli. Jeżeli warunek jest **true** – pętla się uruchomi. Pole to uruchamiane jest (sprawdzany jest warunek) na początku każdej pętli. Odpowiedź **false** na warunek oznacza, że pętla się nie uruchomi.

Przykłady:

```
for (int i=0;i<10;
for (int j=10;j>=0;
for (int a=b;a>0 && a<10;
```

pole3 – krok pętli. Po zakończeniu każdego przebiegu pętli komenda w tym polu jest uruchamiana. Przeważnie w tym polu zmieniana jest wartość zmiennej odpowiadającej za warunek pętli. Pole wykonuje się jako ostatnie.

Przykłady:

```
for (int i=0;i<10;i=i+1)
for (int j=10;j>=0;j--)
for (int a=b;a>0 && a<10;a=a+b)
```

Każde z pól pętli for jest opcjonalne. Pętla for może wyglądać w następujący sposób:

```
for(int i=0; ; ){          // for tylko z pole1
 cout<<i<<endl;
 i=i+1;                    // to zastępuje pole3
 if(i==10) { break;}       // to zastępuje pole2
}


int i=0;                   // to zastępuje pole1
for(;i<10; ){              // for tylko z pole2
 cout<<i<<endl;
 i=i+1;                    // to zastępuje pole3
}


int i=10;                  // to zastępuje pole1
for( ; ; ){                // for bez pole1, pole2 i pole3
 cout<<i<<endl;
 if(i<=0) { break;}// to zastępuje pole2
 i=i-1;                    // to zastępuje pole3
}
```

Pętla for może być pętlą rosnącą lub pętlą malejącą.

Przykład pętli rosnącej:

```
for(int i=0;i<11;i++) {cout<<i<<endl;}
for(int i=10;i<=20;i=i+2) {cout<<i<<endl;}
```

Przykład pętli malejącej:

```
for(int i=10;i>=0;i--) {cout<<i<<endl;}
for(int i=100;i>10;i=i-5) {cout<<i<<endl;}
```

Pętla while

Przeważnie używana w sytuacji, gdy nie wiemy ile razy pętla ma się wykonać. Podobnie jak w pętli **for**, na początku działania pętli **while** sprawdzany jest warunek. Jeżeli warunek jest **true** – pętla się uruchomi, jeżeli jest **false** – kod pętli zostanie pominięty. Należy zadbać o taką konstrukcję pętli, aby w ciele pętli istniała możliwość ustawienia zmiennych tak, aby pętla mogła się w którymś momencie zakończyć.

Składnia

```
while(warunek){
…kod…
}
```

ciekawostka – w ten sam sposób zadziała pętla for(; warunek;)

Przykłady:

```
int a=10;
int b;
cout<<”dzielenie 10 przez podaną liczbę – podaj liczbę”<< endl;
cin>>b;
while(b==0){
cout<<”nie można dzielić przez 0, podaj inne b”<<endl;
cin>>b;
}
```

```
int i=1;
while(i<10){
cout<<”pętla nr “<<i<<endl;
i++;
}
```

Pętla do while

Jedyna pętla, która musi się wykonać przynajmniej raz. Stosowana podobnie jak pętla while. Warunek w pętli sprawdzany jest dopiero na końcu kodu.

Składnia

```
do{
…kod…
}while(warunek);
```

Przykłady:

```
int a;
int suma=0;
do{
cout<<”podaj a”<<endl;
cin>>a;
suma=suma+a;
}while(suma<10);
cout<<”suma podanych liczb przekroczyła 10 i wynosi=”<<suma;

float a,b;
cout<<”podaj 2 liczby większe od 10”<<endl;
do{
cout<<”podaj 1 liczbę:”<<endl;
cin>>a;
cout<<”podaj 2 liczbę:”<<endl;
cin>>b;
}while(a<=10 || b<=10);
```

Instrukcję sterujące przebiegiem pętli

Instrukcja break

Oprócz instrukcji switch może być użyta wewnątrz dowolnej pętli.

- Powoduje natychmiastowe przerwanie wykonywania pętli.
- Jeśli mamy do czynienia z pętlami zagnieżdżonymi, instrukcja break powoduje przerwanie tylko tej pętli, w której została bezpośrednio użyta. Jest to więc przerwanie z wyjściem o jeden poziom wyżej.

Przykład:

```
#include<iostream>
using namespace std;

int main()
{setlocale(LC_CTYPE, "Polish");

for(int i=0;i<11;i++){  // pętla wykonuje się od 0 do 10
 cout<<i<<endl;
 if(i==3){break;}       // gdy i==3 następuje przerwanie działania pętli
}

return 0;}
```

Wynik:

```
0
1
2
3


--------------------------------
Process exited after 0.02899 seconds with return value 0
Press any key to continue . . .
```

Instrukcja continue

- znajduje zastosowanie w pętlach.
- powoduje, że program pomija resztę treści pętli i zaczyna nowy cykl pętli.
- w odróżnieniu od instrukcji break sama pętla nie zostaje przerwana. Przerwany jest tylko bieżący obieg pętli

Przykład:

```
1  #include<iostream>
2  using namespace std;
3
4  int main()
5  {setlocale(LC_CTYPE, "Polish");
6
7  for(int i=10;i>=1;i--){      // pętla wykonuje się od 10 do 1
8    if(i%2==0){continue;}      // gdy zmienna i będzie liczbą parzystą
9    cout<<i<<endl;             // następuje przerwanie i pętla zaczyna się od
10 }                            // początku z kolejną wartości i
11 return 0;}
```

Wynik:

```
9
7
5
3
1


--------------------------------
Process exited after 0.07957 seconds with return value 0
Press any key to continue . . .
```