Pętla **for**

```
#include <iostream>
using namespace std;
int main()
{
   int i;
   for (i=0; i<5; i++)
      cout << "C++ zna pętle." << endl;
   cout << "C++ wie, kiedy przestać." << endl;
   return 0;
}
```

Wynik działania programu:

```
C++ zna pętle.
C++ zna pętle.
C++ zna pętle.
C++ zna pętle.
C++ zna pętle.
C++ wie, kiedy przestać.
```

Pętla **for**

```
#include <iostream>
using namespace std;
int main()
{
   int i;
   for (i = 0; i < 5; i = i + 1)
      cout << "C++ zna pętle." << endl;
   cout << "C++ wie, kiedy przestać." << endl;
   return 0;
}
```

- Inicjalizacja pętli. Pętla zaczyna się od ustawienia liczby całkowitej *i* na 0: **i=0;**. Instrukcja ta wykonuje się tylko raz.
- Warunek pętli. Program sprawdza, czy *i* jest mniejsze od 5: **i<5;**
- Treść pętli. Jeżeli tak, program wykonuje następną instrukcję: **cout << "C++ zna pętle." << endl;**
- Krok pętli. Po zakończeniu treści pętli program zwiększa *i* o 1: **i=i+1**, co kończy pierwszy cykl pętli.
- Następnie pętla zaczyna drugi cykl od porównania nowej wartości zmiennej *i* z 5. Nowa wartość, 1, jest również mniejsza niż 5, więc pokazywany jest kolejny napis i cykl kończy się zwiększeniem *i*. Kolejne etapy cyklu powtarzają się, aż *i* osiągnie wartość 5. Wtedy sprawdzenie warunku daje fałsz, więc program przechodzi do instrukcji znajdującej się za pętlą.

Pętla **for**

Składnia

```
for (inicjalizacja; warunek_pętli; krok_pętli)
{
   treść
}
```

Zwykle pętla **for** obejmuje następujące działania:

1. Ustawienie wartości początkowych (inicjalizacja).
2. Wykonanie testu w celu sprawdzenia, czy pętla ma być dalej wykonywana (warunek pętli).
3. Wykonanie działań objętych pętlą (treść pętli).
4. Aktualizacja wartości używanej (lub używanych) w teście (krok pętli).

Pętla **for**

```
#include <iostream>
using namespace std;
int main()
{
  int limit, i;
  cout << "Podaj wartość początkową odliczania: ";
  cin >> limit;
  for (i = limit; i; i = i - 1)
    cout << "i = " << i << endl;
  cout << "Gotowe, bo już i = " << i << endl;
  return 0;
}
```

Przykładowy wynik działania programu:

```
Podaj wartość początkową odliczania: 4
i = 4
i = 3
i = 2
i = 1
Gotowe, bo już i = 0
```

Pętla **for**

```
#include <iostream>
using namespace std;
int main()
{
   int limit, i;
   cout << "Podaj wartość początkową odliczania: ";
   cin >> limit;
   for (i = limit; i; i = i - 1)
     cout << "i = " << i << endl;
   cout << "Gotowe, bo już i = " << i << endl;
   return 0;
}
```

- Warunkiem pętli jest wyrażenie: **i;**
- W C++ każda wartość zmiennej *i* różna od zera będzie prawdą, a wartość zero - fałszem.
- Pętla zakończy się zatem w chwili, gdy *i* przyjmie wartość 0.

Pętla **for**

```
#include <iostream>
using namespace std;
int main()
{
  int limit, i;
  cout << "Podaj wartość początkową odliczania: ";
  cin >> limit;
  for (i = limit; i; i = i - 1)
    cout << "i = " << i << endl;
  cout << "Gotowe, bo już i = " << i << endl;
  return 0;
}
```

- Pętla **for** to pętla z *uprzednim sprawdzeniem warunku*, co oznacza, że warunek pętli jest sprawdzany przed każdym cyklem pętli. Jeśli warunek już na samym początku nie będzie spełniony, pętla nie wykona się ani razu.
- Wynik działania programu, gdy użytkownik wprowadzi wartość 0:

```
Podaj wartość początkową odliczania: 0
Gotowe, bo już i = 0
```

Pętla **for**

W pętli **for** możliwa jest deklaracja zmiennej.
Zmienna taka będzie jednak istniała tylko w tej pętli. Zatem kiedy program zakończy wykonywanie pętli, zmienna zostanie usunięta.

Poniższy program zakończy się błędem kompilacji.

```
#include <iostream>
using namespace std;
int main()
{
  for (int i = 0; i < 5; i = i + 1)
    cout << "C++ zna pętle." << endl;
  cout << i;
  return 0;
}
```

Instrukcję `i = i + 1` można zastąpić operatorem inkrementacji `i++`.
Instrukcję `i = i - 1` można zastąpić operatorem dekrementacji `i--`.

Pętla **for**

Program umożliwia obliczenie silni liczby n dla $(0 \leq n \leq 7)$.

```
#include <iostream>
using namespace std;
int main()
{
  int n, s, i;
  cout << "Podaj liczbę n (0 <= n <= 7): ";
  cin >> n;
  s = 1;
  for (i = 1; i <= n; i++)
    s = s * i;
  cout << "Silnia liczby " << n << " jest równa: " << s;
  return 0;
}
```

Wynik działania programu dla n = 5.

```
Podaj liczbę n (0 <= n <= 7): 5
Silnia liczby 5 jest równa 120
```

```cpp
#include <iostream>
using namespace std;
int main()
{
  int k, s=0;
  cout << "Podaj liczbę całkowitą: ";
  cin >> k;
  cout << "Zliczanie co " << k << endl;
  for (int i = 0; i < 100; i = i + k)
  {
    cout << i << endl;
    s = s + i;
  }
  cout << "Suma jest równa: " << s;
}
```

```
Podaj liczbę całkowitą 30
Zliczanie co 30:
0
30
60
90
Suma jest równa 180
```

Pętla **while**

Składnia

while (warunek_pętli)
{
  treść
}

- Najpierw program wyznacza wartość wyrażenia *warunek_pętli* ujętego w nawiasy. Jeśli wynikiem jest *true*, wykonywane są instrukcje z treści pętli.
- Po zakończeniu treści pętli, program ponownie wyznacza wartość warunku pętli.
- Proces trwa tak długo, dopóki warunek nie przyjmie wartości *false*.

Pętla **for** i pętla **while**

```
#include <iostream>
using namespace std;
int main()
{
  for (int i = 0; i < 5; i++)
    cout << "C++ zna pętle." << endl;
  cout << "C++ wie, kiedy przestać." << endl;
}
```

```
#include <iostream>
using namespace std;
int main()
{
  int i = 0;
  while (i < 5)
  {
    cout << "C++ zna pętle." << endl;
    i++;
  }
  cout << "C++ wie, kiedy przestać." << endl;
}
```

Pętla **for** i pętla **while**

Pętla **for** w postaci:

```
for (inicjalizacja; warunek_pętli; aktualizacja)
{
   instrukcje
}
```

może zostać zapisana następująco:

```
inicjalizacja;
while (warunek_pętli)
{
   instrukcje
   aktualizacja
}
```

Pętla **for** i pętla **while**

Podobnie pętla **while** w postaci:

```
while (warunek_pętli)
  treść
```

może zostać zapisana następująco:

```
for ( ; warunek_pętli; )
  treść
```

Pętla **for** wymaga trzech wyrażeń, ale mogą być to wyrażenia puste, obowiązkowe są tylko średniki. Brak warunku pętli **for** jest traktowany jako wyrażenie stałe *true*, więc następująca pętla będzie pętlą nieskończoną:

```
for ( ; ; )
  treść
```

Pętle **while** są używane zwykle wtedy, kiedy trudno z góry przewidzieć, ile razy pętla powinna się wykonać.

Program sumuje liczby całkowite do momentu wprowadzenia 0.

```
#include <iostream>
using namespace std;
int main()
{
   int x, s;
   cout << "Podaj liczbę: ";
   cin >> x;
   s = x;
   while (x != 0)
   {
      cout << "Podaj liczbę: ";
      cin >> x;
      s = s + x;
   }
   cout << "Suma jest równa: " << s;
}
```

```
Podaj liczbę: 5
Podaj liczbę: 2
Podaj liczbę: 6
Podaj liczbę: 0
Suma jest równa: 13
```

Pętla **do while**

Pętla **do while** istotnie różni się od pętli **for** i **while**, gdyż warunek jest w niej sprawdzany po wykonaniu treści.
Wobec tego pętla **do while** wykona się zawsze co najmniej raz, niezależnie od spełnienia (bądź niespełnienia) warunku.

Składnia

```
do
{
   treść
} while (warunek_pętli)
```

```
int main()
{
   int x, s=0;
   cout << "Podaj liczbę: ";  cin >> x;
   while (x != 0)
   {
      cout << "Podaj liczbę: ";  cin >> x;
      s = s + x;
   }
   cout << "Suma jest równa: " << s;
}
```

```
int main()
{
   int x, s=0;
   do
   {
      cout << "Podaj liczbę: ";  cin >> x;
      s = s + x;
   } while (x != 0);
   cout << "Suma jest równa: " << s;
}
```

Pętla **do while**

Program wykonuje sumowanie liczb całkowitych. Sumowanie zostaje zakończone, gdy suma składników przekroczy wartość 100.

```
#include <iostream>
using namespace std;
int main()
{
   int x, s, licznik;
   s = 0;
   licznik = 0;
   do
   {
      cout << "Podaj liczbę: ";
      cin >> x;
      s = s + x;
      licznik++;
   }while (s <= 100);
   cout << "Suma jest równa: " << s << endl;
   cout << "Liczba składników: " << licznik;
}
```

```cpp
#include <iostream>
using namespace std;
int main()
{
  int x, s=0, licznik=0;
  do
  {
    cout << "Podaj liczbę: ";
    cin >> x;
    s = s + x;
    licznik++;
  }while (s <= 100);
  cout << "Suma jest równa: " << s << endl;
  cout << "Liczba składników: " << licznik;
}
```

```
Podaj liczbę: 30
Podaj liczbę: 25
Podaj liczbę: 40
Podaj liczbę: 8
Suma jest równa: 103
Liczba składników: 4
```

Instrukcje **break** i **continue**

Instrukcje break i continue umożliwiają programowi pomijanie części kodu.

- 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.
- 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.

Instrukcja **break**

```
#include <iostream>
using namespace std;
int main()
{
   int i = 7;
   while (1)
   {
      cout << "Pętla , i = " << i << endl;
      i = i - 1;
      if (i < 5)
      {
         cout << "Przerwanie pętli.";
         break;
      }
   }
}
```

```
Pętla , i = 7
Pętla , i = 6
Pętla , i = 5
Przerwanie pętli.
```

Instrukcja **continue**

```
#include <iostream>
using namespace std;
int main()
{
   int k;
   for (k = 0; k < 12; k = k+1)
   {
      cout << "A";
      if (k > 1) continue;
      cout << "b" << endl;
   }
}
```

```
Ab
Ab
AAAAAAAAAA
```

Zagnieżdżenie pętli

```
#include <iostream>
using namespace std;
int main()
{
  for (int i = 1; i <= 7; i++)
  {
    for(int j = 1; j <= i; j++)
    {
      cout << "*";
    }
    cout << endl;
  }
}
```

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