

Iteracje

Iteracja (łac. *iteratio* – powtarzanie) to czynność powtarzania (najczęściej wielokrotnego) tej samej instrukcji albo wielu różnych instrukcji w pętli.

W języku C++ istnieją trzy instrukcje iteracyjne:

- *for* (dla),
- *do ... while* (powtarzaj),
- *while* (dopóki).

Pętlę *for* stosujemy w sytuacji, kiedy dokładnie wiemy, ile razy ma ona zostać wykonana. Istnieje wiele wariantów tej pętli, ale zawsze możemy wyróżnić trzy główne części:

1. **Inicjalizacja** to zwykle instrukcja przypisania stosowana do ustawienia początkowej wartości zmiennej sterującej pętlą.
2. **Warunek** jest wyrażeniem relacyjnym określającym moment zakończenia wykonywania pętli.
3. **Inkrementacja** (zwiększenie) lub **dekrementacja** (zmniejszenie) definiuje sposób modyfikacji zmiennej sterującej pętlą po zakończeniu każdego przebiegu (powtórzenia).

Te trzy główne składowe oddzielone są od siebie średnikami (;).

Pętla *for* wykonywana jest dopóty, dopóki wartość warunku wynosi *true*. Gdy warunek osiągnie wartość *false*, działanie programu jest kontynuowane od pierwszej instrukcji znajdującej się za pętlą.

W przeciwieństwie do Turbo Pascala w języku C++ zmienna sterującą pętlą *for* nie musi być typu całkowitego, znakowego lub logicznego. Może być ona na przykład typu *float*.

Pętla ta może być wykonywana tyle razy, ile wartości znajduje się w przedziale

inicjalizacja; warunek; zwiększanie

lub

inicjalizacja; warunek; zmniejszanie

Ogólna postać tej instrukcji jest następująca:

```
for (inicjalizacja; warunek; zwiększenie)
{
    // instrukcje
}
```

lub

```
for (inicjalizacja; warunek; zmniejszanie)
{
    // instrukcje
}
```

W języku C++ jest możliwa zmiana przyrostu zmiennej sterującej pętla.

Kolejną instrukcją iteracyjną jest *do ... while*. Jej ogólna postać jest następująca:

```
do
{
    // instrukcje
}
while (warunek);
```

Cechą charakterystyczną instrukcji iteracyjnej *do ... while* jest to, że bez względu na wartość zmiennej *warunek* pętla musi zostać wykonana **co najmniej jeden raz**. Program po napotkaniu instrukcji *do ... while* wchodzi do pętli i wykonuje instrukcje znajdujące się w nawiasach klamrowych { }, a następnie sprawdza, czy *warunek* jest spełniony. Jeśli tak, wraca na początek pętli, natomiast jeśli *warunek* osiągnie wartość *false* (nieprawda), pętla się zakończy.

Ostatnią instrukcją iteracyjną jest *while*. Jej ogólna postać jest następująca:

while (warunek)

{

// instrukcje

}

Cechą charakterystyczną tej instrukcji jest sprawdzenie warunku jeszcze przed jej wykonaniem. W szczególnym przypadku pętla może nie zostać wcale wykonana. Instrukcja *while* powoduje wykonywanie instrukcji dopóty, dopóki warunek jest prawdziwy.

Przykładowy program napisany w języku C++ wykorzystujący iteracje

Program, który wykorzystuje iteracje do zmiany wartości zmiennej x , zmieniających się od 0 do 10 oraz dla każdej przyjętej wartości oblicza wartość funkcji $y = 3x$.

Przy użyciu instrukcji for:

```
#include <iostream>
#include <conio.h>

using namespace std;

int main()
{
    int x, y;

    cout << "Program oblicza wartość funkcji y = 3x dla x
zmieniającego się od 0 do 10." << endl;

    for (x = 0; x <= 10; x++)
    {
        y = 3 * x;
        cout << "x = " << x << '\t' << "y = " << y << endl;
    }

    getch();

    return 0;
}
```

Opis instrukcji, linii kodu

W pętli:

for (x = 0; x <= 10; x++)

Kolejne wartości x , zmieniające się automatycznie od 0 (inicjacja) do 10 (warunek) z krokiem 1 (zwiększenie), będą podstawiane do wzoru:

*y = 3 * x;*

a następnie zostaną wyświetcone na ekranie.
Znak '\t' oznacza przejście do następnej pozycji w tabulacji linii.

Przy użyciu instrukcji do ... while:

```
#include <iostream>
```

```
#include <conio.h>
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
    auto x = 0, y = 0;
```

cout << "Program oblicza wartość funkcji $y = 3x$ dla x zmieniającego się od 0 do 10." << endl;

```
do
```

```
{
```

```
    y = 3 * x;
```

```
    cout << "x = " << x << '\t' << "y = " << y << endl;
```

```
    x++;
```

```
}
```

```
while (x <= 10);
```

```
_getch();
```

```
return 0;
```

```
}
```

Opis instrukcji, linii kodu

Pętla do ... while:

```
do
{
    y = 3 * x;
    cout << "x = " << x << '\t' << "y = " << y << endl;
    x++;
}
while (x <= 10);
```

nie ma wbudowanego mechanizmu zmiany sterującej nią zmiennej, dlatego musimy do niej ten mechanizm dobudować. Rolę zmiennej sterującej pełni tutaj x . Zmienną tę powinniśmy przed pętlą wyzerować, stąd zapis:

```
auto x = 0;
```

Następnie x należy zwiększać o krok, który w naszym przypadku wynosi 1. Ilustruje to następująca linijka kodu:

```
x++;
```

Pętla będzie powtarzana tak długo, aż zostanie spełniona zależność $x \leq 10$. Zwróćmy uwagę, że warunek sprawdzający zakończenie działania pętli, tzn. *while ($x \leq 10$)*, znajduje się na jej końcu.

Przy użyciu instrukcji while:

```
#include <iostream>
#include <conio.h>

using namespace std;

int main()
{
    auto x = 0, y = 0;

    cout << "Program oblicza wartość funkcji y = 3x dla x
zmieniającego się od 0 do 10." << endl;

    while (x <= 10)
    {
        y = 3 * x;
        cout << "x = " << x << '\t' << "y = " << y << endl;
        x++;
    }

    getch();

    return 0;
}
```

Opis instrukcji, linii kodu

Pętla *while*, podobnie jak *do .. while*, nie ma wbudowanego mechanizmu sterującej nią zmiennej, musimy więc do niej ten mechanizm dobudować. Rolę zmiennej sterującej pełni tutaj zmienna *x*:

```
while (x <= 10)
{
    y = 3 * x;
    cout << "x = " << x << '\t' << "y = " << y << endl;
    x++;
}
```

Zmienną *x* powinniśmy przed pętlą wyzerować, zatem:

```
auto x = 0;
```

Następnie należy ją zwiększać o krok, który w naszym przypadku wynosi 1. Ilustruje to następująca linijka kodu:

```
x++;
```

Pętla będzie powtarzana tak długo, aż stanie się prawdziwa zależność $x \leq 10$. Zwróćmy uwagę, że warunek sprawdzający zakończenie działania pętli, tzn. *while ($x \leq 10$)*, znajduje się na jej początku.

Zadania

3.1 Napisz program, który za pomocą instrukcji **for** wyświetla liczby całkowite od **1** do **20**.

3.2 Napisz program, który za pomocą instrukcji **do .. while** sumuje liczby całkowite od **1** do **100**.

3.3 Napisz program, który za pomocą instrukcji **while** sumuje liczby parzyste z przedziału od **1** do **100**.

wskazówki do zadania

Należy skorzystać z właściwości operatora modulo (**%**).

3.4 Napisz program, który za pomocą instrukcji ***do .. while*** sumuje liczby nieparzyste z przedziału od **1** do **100**.

wskazówki do zadania

Należy skorzystać z właściwości operatora modulo (**%**) i operatora negacji (**!**).

3.5 Napisz program, który za pomocą instrukcji ***for*** znajduje największą i najmniejszą liczbę ze zbioru **n** liczb losowych z przedziału od **0** do **99** oraz oblicza ich średnią (w zadaniu **n = 5**).

3.6 Napisz program wyświetlający tabliczkę mnożenia dla liczb od **1** do **100** z wykorzystaniem podwójnej pętli ***while***.

3.7 Napisz program, który wyświetla duże litery alfabetu od **A** do **Z** i od **Z** do **A** z wykorzystaniem pętli **do .. while**.

wskazówki do zadania

Litery alfabetu oddzielone są od siebie znakiem (,), każdy ciąg liter alfabetu wyświetlany jest w nowej linii zakończony znakiem (.).

3.8 Zmodyfikuj program z poprzedniego działu z zadania **2.4**, używając wybranej przez siebie iteracji, tak aby użytkownik zgadując liczbę dostawał od programu odpowiedzi czy zgadywana liczba jest mniejsza czy większa od wylosowanej. Gdy użytkownik już zgadnie wylosowaną liczbę program zakończy iterację wyświetlając obok gratulacji, informację po ilu trafieniach zgadł daną liczbę. Zmodyfikuj

również przedział losowanej liczby na liczby z przedziału od **0** do **100**.

3.9 Wykonuj wszystkie zadania rzetelnie i bez pomocy kolegi/koleżanki. Kolega/Koleżanka nie napisze za Ciebie egzaminu, nawet gdy trafisz z nim/nią do tej samej grupy egzaminacyjnej, to każda próba jakiejkolwiek pomocy kończy się wyproszeniem Ciebie oraz osoby, która próbowała Tobie pomóc z egzaminu przez Egzaminatora oraz niezaliczeniem danej kwalifikacji zawodowej dla was dwóch lub nawet całej grupy egzaminacyjnej, w zależności od decyzji Egzaminatora.

Powodzenia ;)