Pętle warunkowe w języku C++

- Wprowadzenie
- Przeczytaj
- Prezentacja multimedialna
- Sprawdź się
- Dla nauczyciela

Źródło: Heni Noviyanti, domena publiczna.

> W tym e-materiale powtarzamy wiadomości ze szkoły podstawowej.

Pętle należą do grupy instrukcji sterujących działaniem programu. Pozwalają one na cykliczne wykonanie instrukcji. Poznaliśmy już pętlę `for` oraz działanie pętli warunkowej `while`.

W tym e-materiale dowiesz się, jak dokonać implementacji pętli `while` w języku C++.

Ciekawi cię, jak wyglądają implementacje w innych językach programowania? Możesz się z nimi zapoznać w dwóch pozostałych e-materiałach z tej serii:

- Pętle warunkowe w języku Java,
- Pętle warunkowe w języku Python.

Więcej zadań? Sięgnij do Pętle warunkowe – zadania maturalne.

> **Twoje cele**
>
> - Zidentyfikujesz sytuacje, w których należy zastosować pętle warunkowe.
> - Przeanalizujesz składnie pętli warunkowej w języku C++.
> - Zaimplementujesz programy, w którym zastosujesz pętle warunkowe.

Przeczytaj

Pętle są jednym z najważniejszych narzędzi używanych w programowaniu. Dzięki nim nasz kod staje się o wiele krótszy, a tym samym bardziej przejrzysty. Pętle, którymi się teraz zajmiemy, to pętle `while` oraz `do .. while`. Zanim jednak przejdziemy do implementacji tych pętli w języku C++, przedstawmy kilka najważniejszych informacji.

Pętla `while` składa się z kluczowego słowa `while`, występującego na początku pętli.

Instrukcje zawarte w pętli będą wykonywane tylko w przypadku spełnienia warunku zawartego w pętli `while.` Jeżeli warunek nie będzie spełniony na początku, żadne operacje zawarte w pętli nie zostaną wykonane.

Pętla `do .. while` składa się ze słowa kluczowego `do`, które znajduje się na początku pętli, a kończy się słowem kluczowym `while`.

Warunek pętli sprawdzany jest na końcu, dlatego instrukcje zawarte w pętli wykonają się co najmniej jeden raz.

Pętla while

Zajmijmy się pętlą `while` i jej składnią w języku C++.

```
while (warunek) {
    // Instrukcje
}
```

Przeanalizujmy poniższą pętlę `while`. Za co jest odpowiedzialna?

```
int i = 4

while (i > 0) {
    cout << "INFORMATYKA" << endl;

    i--;
}
```

Na początek deklarujemy zmienną `i`, której wartość będzie równa 4. Następnie, dopóki zmienna `i` jest większa od zera, wypisujemy słowo „INFORMATYKA”. Z każdą iteracją pętli,

zmienna `i` będzie dekrementowana. W związku z tym program wyświetli na ekranie cztery razy napis „INFORMATYKA”. W przypadku, gdy `i` stanie się mniejsze lub równe zero, warunek nie będzie już spełniony i pętla nie zostanie wykonana więcej razy.

Spróbujmy zmodyfikować powyższą pętlę tak, aby jej warunek był zawsze spełniony.

```
int i = 4

while (i > 0) {
    cout << "INFORMATYKA" << endl;
}
```

Wystarczyło z naszej pętli usunąć dekrementację zmiennej `i`. Dzięki temu, ta zmienna przez cały czas działania pętli będzie przechowywała wartość 4. Liczba 4 jest większa od 0, tym samym warunek będzie zawsze spełniony i powstanie pętla nieskończona.

Aby uzyskać pętlę nieskończoną, możemy użyć następującej składni:

```
while (true) {
    cout << "PĘTLA NIESKOŃCZONA" << endl;
}
```

Na wyjściu programu otrzymamy ciągłe, nieskończone wyświetlanie tekstu „PĘTLA NIESKOŃCZONA”. W tym celu wystarczyło umieścić w warunku wykonania pętli `true`.

Co w przypadku, gdy chcielibyśmy, aby nasza pętla nie wykonała się ani razu?

Jest na to sposób:

```
int i = 4

while (i > 5) {
    cout << "TEN NAPIS SIĘ NIE WYPISZE" << endl;
}
```

Wystarczyło umieścić w warunku wykonania pętli wyrażenie, które jest nieprawdziwe. Liczba przechowywana w zmiennej `i`, to znaczy liczba 4, nie jest większa od 5, dlatego pętla nie zostanie wykonana ani razu.

Pętla do .. while

Istnieje inna pętla – `do .. while`. Jej składnia w języku C++ wygląda następująco:

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

Jaka jest różnica między omawianymi dwiema pętlami?

Przeanalizujmy poniższą pętlę w celu uzyskania odpowiedzi na to pytanie.

```
do {
    cout << "KOCHAM INFORMATYKĘ" << endl;
} while (1 > 2);
```

Mimo że warunek nie jest spełniony (1 nie jest większe od 2), na ekranie zostanie wyświetlony napis „KOCHAM INFORMATYKĘ". Dlaczego? W pętli `do .. while` warunek sprawdzany jest na jej końcu. Możemy zatem być pewni, że używając właśnie tej pętli, instrukcje w niej zawarte zostaną wykonane co najmniej jeden raz, tak jak w powyższym przypadku.

Upewnijmy się, że pętla `do .. while` wykona się co najmniej jeden raz, a pętla `while` może nie wykonać się ani razu.

```
do {
    cout << "Jestem do .. while" << endl;
} while (1 > 2);

while (1 > 2) {
    cout << "Jestem while" << endl;
}
```

Na wyjściu otrzymamy „Jestem do .. while". Zatem instrukcje zawarte w pętli `while` nie wykonały się ani razu, w przeciwieństwie do instrukcji w pętli `do .. while`.

Słownik

> **dekrementacja**

zmniejszenie wartości argumentu o jeden

pętla nieskończona

pętla, której warunek wykonania jest zawsze spełniony

Prezentacja multimedialna

Polecenie 1

Napisz program, który będzie wypisywał liczby parzyste z przedziału `<m, n>`. Przetestuj działanie programu dla przedziału `<1, 20>`.

Specyfikacja problemu:

Dane:

- `<m, n>` – przedział, w skład którego wchodzą liczby całkowite dodatnie

Wynik:

Program wyświetla liczby parzyste z przedziału `<m, n>`.

```
#include <iostream>

using namespace std;

int main() {
    int m = 1;
    int n = 20;

    // tutaj dopisz kod
}
```

```
1
```

Polecenie 2

Porównaj swoje rozwiązanie z prezentacją, która analizuje i rozwiązuje podobny problem.

1

Pętla jest jedną z podstawowych konstrukcji programowania strukturalnego. Umożliwia cykliczne wykonywanie ciągu instrukcji określoną liczbę razy, do momentu zajścia pewnych warunków, dla każdego elementu kolekcji lub w nieskończoność.
Źródło: pixabay.com, domena publiczna.

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PczU5nNBT

Naszym zadaniem jest napisanie programu, który będzie wypisywał liczby parzyste z przedziału <1, limit>, gdzie limit będzie podawany przez użytkownika.

2

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PczU5nNBT

Zacznijmy od zbudowania szkieletu programu, czyli zadeklarowania głównej funkcji programu – `main`. To w niej będzie się znajdować reszta naszego programu.

```
#include <iostream>

using namespace std;

int main() {
    return 0;
}
```

3

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PczU5nNBT

Następnie zadeklarujmy dwie zmienne typu całkowitego:

- `int liczba` – będzie to zmienna sterująca naszej pętli. Będziemy tę zmienną inkrementować w pętli i sprawdzać, czy jest parzysta.
- `int limit` – górna granica przedziału, do której mają być wypisywane liczby parzyste. Będzie ustalana przez użytkownika.

```
#include <iostream>

using namespace std;

int main() {
```

```
6     int liczba = 1;
7     int limit;
8
9     return 0;
10 }
```

4

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PczU5nNBT

Następnym krokiem będzie napisanie kodu odpowiadającego za komunikację programu z użytkownikiem.

Wartość górnej granicy pobieramy za pomocą strumienia `cin` i zapisujemy w zmiennej `limit`.

```
1  #include <iostream>
2
3  using namespace std;
4
5  int main() {
6      int liczba = 1;
7      int limit;
8
9      cout << "Wprowadź granicę, do której chcesz aby zostały wypisane liczby parzyste" << endl;
10     cin >> limit;
11
12     return 0;
13 }
```

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PczU5nNBT

5

Napiszmy teraz szkielet pętli `while`, której liczba iteracji zależy od wartości podanej przez użytkownika.

```
1  #include <iostream>
2
3  using namespace std;
4
5  int main() {
6      int liczba = 1;
7      int limit;
8
9      cout << "Wprowadź granicę, do której chcesz aby zostały wypisane liczby parzyste" << endl;
10     cin >> limit;
11
12     while (liczba <= limit) {
13
14
15     }
16
17     return 0;
18 }
```

6

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PczU5nNBT

Kolejnym krokiem będzie napisanie instrukcji warunkowej `if`, w której będziemy sprawdzać, czy liczba jest parzysta.

```
1  #include <iostream>
2
3  using namespace std;
4
5  int main() {
6      int liczba = 1;
```

```
7     int limit;
8
9     cout << "Wprowadź granicę, do której chcesz aby zostały wypisane liczby parzyste" << endl;
10    cin >> limit;
11
12    while (liczba <= limit) {
13        if (liczba % 2 == 0) {
14            cout << liczba << endl;
15        }
16    }
17
18    return 0;
19 }
```

7

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PczU5nNBT

To, czy liczba jest parzysta, sprawdzamy przy pomocy operatora modulo – %. Jeżeli reszta z dzielenia `liczba` przez 2 jest równa 0, znaczy to że jest ona wielokrotnością dwójki, czyli jest parzysta.

8

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PczU5nNBT

Nie zapomnijmy o inkrementacji zmiennej sterującej `liczba`, aby pętla nie wykonywała się w nieskończoność.

```
1  #include <iostream>
2
3  using namespace std;
4
5  int main() {
6      int liczba = 1;
7      int limit;
8
9      cout << "Wprowadź granicę, do której chcesz aby zostały wypisane liczby parzyste" << endl;
10     cin >> limit;
11
12     while (liczba <= limit) {
13         if (liczba % 2 == 0) {
14             cout << liczba << endl;
15         }
16
17         liczba++;
18     }
19
20     return 0;
21 }
```


Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PczU5nNBT

Program jest gotowy. Zastanów się jednak, co musielibyśmy zmienić, jeżeli zamiast pętli `while` użylibyśmy `do .. while`.

Wykonaj to zadanie samodzielnie, a następnie sprawdź kolejny slajd.

Pamiętajmy o regule: pętla wykonuje się, dopóki warunek w środku jest spełniony, czyli zwraca wartość *true*. Jeśli choć raz ten warunek się nie spełni, pętla zostaje zerwana – zakończona.
Źródło: pixabay.com, domena publiczna.

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PczU5nNBT

Tak wygląda kod programu po zmianie typu pętli z `while` na `do .. while`.

```
1  #include <iostream>
2
3  using namespace std;
4
5  int main() {
6      int liczba = 1;
7      int limit;
8
9      cout << "Wprowadź
   granicę, do której chcesz
   aby zostały wypisane
   liczby parzyste" << endl;
10     cin >> limit;
11
12     do {
13         if (liczba % 2 ==
   0) {
14             cout << liczba
   << endl;
15         }
16
17         liczba++;
```

```
18     } while (liczba <=
limit);
19
20     return 0;
21 }
```

Polecenie 3

Zapisz notatkę, w której podsumujesz najważniejsze informacje przedstawione w prezentacji multimedialnej.

Sprawdź się

Pokaż ćwiczenia:

Ćwiczenie 1

Napisz program, który wypisze x razy dany łańcuch znaków `tekst`. Użyj pętli `while`. Przetestuj działanie programu dla łańcucha znaków `panta rhei`, który ma zostać wypisany 19 razy.

Specyfikacja problemu:

Dane:

- `tekst` – łańcuch znaków do wypisania
- x – liczba naturalna

Wynik:

Program wypisuje w nowych liniach x razy łańcuch znaków `tekst`.

Twoje zadania

1. Program ma wypisać 19 razy napis `panta rhei`.

Ćwiczenie 2

Napisz program, który wypisze wszystkie liczby całkowite z przedziału `<m, n>`. Użyj pętli `while`. Przetestuj działanie programu dla przedziału <0, 19>.

Specyfikacja problemu:

Dane:

- `m` – liczba naturalna dodatnia
- `n` – liczba naturalna dodatnia

Wynik:

Program wypisuje wszystkie liczby całkowite z przedziału `<m, n>`.

Twoje zadania

1. Program ma wypisywać kolejne liczby z przedziału `<m, n>`. Użyj pętli `while`. Liczby wypisuj w nowych liniach.

Ćwiczenie 3

Napisz program, który za pomocą pętli oblicza sumę liczb nieparzystych w przedziale `<m, n>`. Przetestuj działanie programu dla przedziału <1, 20>.

Specyfikacja problemu:

Dane:

- `<m, n>` – przedział zawierający liczby całkowite

Wynik:

- `suma` – suma liczb nieparzystych w przedziale `<m, n>`

Twoje zadania

1. Program wypisuje sumę liczb nieparzystych w przedziale `<m, n>`.

Dla nauczyciela

Autor: Maurycy Gast
Przedmiot: Informatyka

Temat: Pętle warunkowe w języku C++

Grupa docelowa:

Liceum ogólnokształcące i technikum, liceum ogólnokształcące, technikum, zakres podstawowy i rozszerzony

Podstawa programowa:

Cele kształcenia – wymagania ogólne

> II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera oraz innych urządzeń cyfrowych: układanie i programowanie algorytmów, organizowanie, wyszukiwanie i udostępnianie informacji, posługiwanie się aplikacjami komputerowymi.

Treści nauczania – wymagania szczegółowe

> II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych.
>
> > Zakres podstawowy. Uczeń:
> >
> > > 1) projektuje i programuje rozwiązania problemów z różnych dziedzin, stosuje przy tym: instrukcje wejścia/wyjścia, wyrażenia arytmetyczne i logiczne, instrukcje warunkowe, instrukcje iteracyjne, funkcje z parametrami i bez parametrów, testuje poprawność programów dla różnych danych; w szczególności programuje algorytmy z punktu I.2);
> >
> > Zakres rozszerzony. Uczeń spełnia wymagania określone dla zakresu podstawowego, a ponadto:
> >
> > > 2) stosuje zasady programowania strukturalnego i obiektowego w rozwiązywaniu problemów;
> > >
> > > 3) sprawnie posługuje się zintegrowanym środowiskiem programistycznym przy pisaniu, uruchamianiu i testowaniu programów;

Kształtowane kompetencje kluczowe:

- kompetencje cyfrowe;

- kompetencje osobiste, społeczne i w zakresie umiejętności uczenia się;
- kompetencje matematyczne oraz kompetencje w zakresie nauk przyrodniczych, technologii i inżynierii.

Cele operacyjne (językiem ucznia):

- Zidentyfikujesz sytuacje, w których należy zastosować pętle warunkowe.
- Przeanalizujesz składnie pętli warunkowej w języku C++.
- Zaimplementujesz programy, w którym zastosujesz pętle warunkowe.

Strategie nauczania:

- konstruktywizm;
- konektywizm.

Metody i techniki nauczania:

- dyskusja;
- rozmowa nauczająca z wykorzystaniem multimedium i ćwiczeń interaktywnych;
- ćwiczenia praktyczne.

Formy pracy:

- praca indywidualna;
- praca w parach;
- praca w grupach;
- praca całego zespołu klasowego.

Środki dydaktyczne:

- komputery z głośnikami, słuchawkami i dostępem do internetu;
- zasoby multimedialne zawarte w e-materiale;
- tablica interaktywna/tablica, pisak/kreda;
- oprogramowanie dla języka C++, w tym kompilator GCC/G++ 4.5 (lub nowszej wersji) i Code::Blocks 16.01 (lub nowszej wersji), Orwell Dev-C++ 5.11 (lub nowszej wersji) lub Microsoft Visual Studio.

Przebieg lekcji

Przed lekcją:

1. **Przygotowanie do zajęć.** Nauczyciel loguje się na platformie i udostępnia e-materiał: „Pętle warunkowe w języku C++”. Uczniowie zapoznają się z treściami w sekcji „Przeczytaj”.

Faza wstępna:

1. Nauczyciel wyświetla uczniom temat, wskazuje cele zajęć oraz ustala z uczestnikami zajęć kryteria sukcesu.
2. **Rozpoznanie wiedzy uczniów.** Nauczyciel prosi wybranego ucznia lub uczniów o przedstawienie sytuacji problemowej związanej z tematem lekcji.

Faza realizacyjna:

1. **Praca z tekstem.** Uczniowie przystępują do cichego czytania tekstu e-materiału. Indywidualnie zapoznają się z treścią w sekcji „Przeczytaj".
2. **Praca z multimedium.** Uczniowie pracują w parach. Analizują treść problemu 1 z sekcji „Prezentacja multimedialna" i opracowują jego rozwiązanie. Następnie porównują je z omówionym w prezentacji.
3. **Ćwiczenie umiejętności.** Uczniowie wykonują ćwiczenia nr 1-2 z sekcji „Sprawdź się". Nauczyciel sprawdza poprawność wykonanych zadań, omawiając je wraz z uczniami. Następnie uczniowie indywidualnie rozwiązują ćwiczenie nr 3 na czas. Osoba, która poprawnie rozwiąże zadania jako pierwsza, wygrywa, a nauczyciel może nagrodzić ją oceną za aktywność.

Faza podsumowująca:

1. Na koniec zajęć z programowania w C++ nauczyciel prosi uczniów o rozwinięcie zdania: „Na dzisiejszych zajęciach nauczyłam/łem się jak…".

Praca domowa:

1. Uczniowie proponują alternatywny sposób rozwiązania problemu postawionego w sekcji „Prezentacja multimedialna".

Materiały pomocnicze:

- Oficjalna dokumentacja techniczna dla kompilatora GCC/G++ 4.5 (lub nowszej wersji).
- Oficjalna dokumentacja techniczna dla oprogramowania Code::Blocks 16.01 (lub nowszej wersji), Orwell Dev-C++ 5.11 (lub nowszej wersji) lub Microsoft Visual Studio.
- Oficjalna dokumentacja techniczna dla oprogramowania Code::Blocks 16.01 (lub nowszej wersji), Orwell Dev-C++ 5.11 (lub nowszej wersji) lub Microsoft Visual Studio.

Wskazówki metodyczne:

- Uczniowie mogą wykorzystać treści w sekcjach: „Przeczytaj", „Prezentacja multimedialna", „Sprawdź się" jako materiał do lekcji powtórkowej.