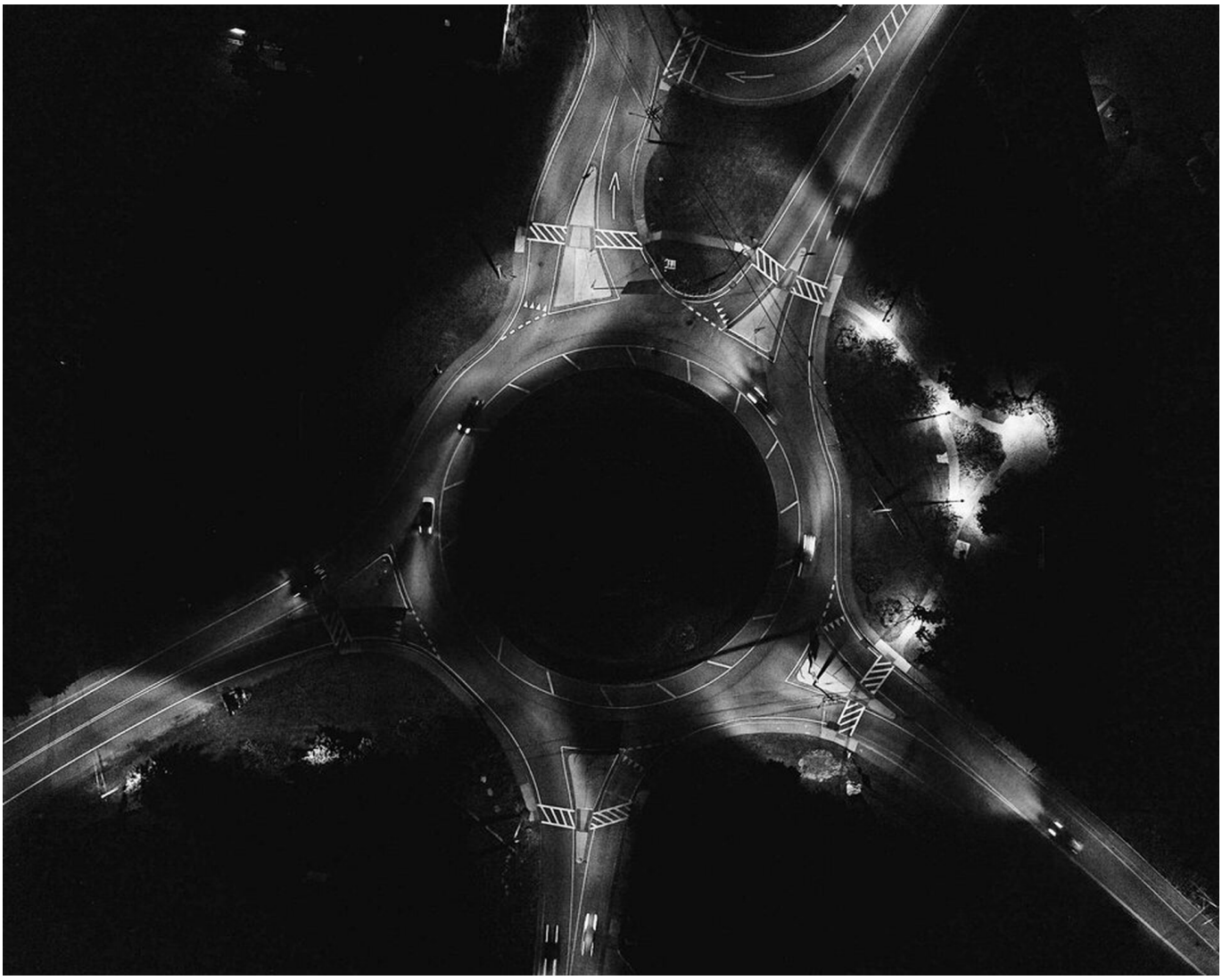


Instrukcja warunkowa w języku C++

- Wprowadzenie
- Przeczytaj
- Schemat interaktywny
- Sprawdź się
- Dla nauczyciela

Źródło: Osman Rana, domena publiczna.

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

Wiesz już, czym charakteryzują się instrukcje warunkowe. Informacje na ten temat znajdziesz w e-materiale Instrukcja warunkowa.

W tym e-materiale zapoznasz się z ich implementacją w języku C++.

Implementacje w innych językach programowania:

- Instrukcja warunkowa w języku Java,
- Instrukcja warunkowa w języku Python.

Więcej zadań? Sięgnij do: Instrukcja warunkowa – ćwiczenia.

> **Twoje cele**
>
> - Przeanalizujesz składnię instrukcji warunkowej w języku C++.
> - Wyjaśnisz, jak zapisać warunki wielokrotnie złożone.
> - Wykorzystasz instrukcję warunkową do rozwiązywania prostych problemów.

Przeczytaj

Definiowanie warunków

Warunek może przyjąć dwie wartości logiczne: prawda (`true`) lub fałsz (`false`). Często oznaczane także 1 (prawda) i 0 (fałsz).

Operacje zawarte w instrukcji warunkowej wykonają się tylko wtedy, kiedy postawiony warunek przyjmie wartość `true`.

Jedną z podstawowych operacji wykorzystywanych do tworzenia warunków, podobnie jak w przypadku naszego programu obliczającego cenę, jest operacja porównania. W zależności od wyniku działania zwracana jest odpowiednia wartość logiczna, np.

```
int x = 5;

std::cout << (x == 5) << std::endl; // wyświetli 1 (true)
std::cout << (x != 5) << std::endl; // wyświetli 0 (false)
std::cout << (x == 7) << std::endl; // wyświetli 0 (false)
std::cout << (x > 2) << std::endl;  // wyświetli 1 (true)
std::cout << (x < 5) << std::endl;  // wyświetli 0 (false)
std::cout << (x <= 5) << std::endl; // wyświetli 1 (true)
std::cout << (x >= 5) << std::endl; // wyświetli 1 (true)
```

> **Ważne!**
>
> Do przypisania wartości do zmiennej używa się operatora `=`, natomiast do porównania ze sobą dwóch elementów służą **operatory relacyjne**. Podwójny znak równości `==` wskaże, czy dwa elementy są sobie równe. Operator „różne od” zapisujemy jako `!=`. Do silnych nierówności wykorzystamy `>` i `<`, natomiast do słabych `>=` i `<=`.

Warunki złożone

Choć warunki można zagnieżdżać, to istnieją metody, które ułatwiają zapisywanie złożonych warunków.

Warunki zagnieżdżone:

```
#include <iostream>

```

```
using namespace std;

int main()
{
    int x = 5;

    if (x < 10) {
        if (x > 3) {
            std::cout << "Liczba jest większa niż 3 i mniejsza ni
         }
    }
}
```

W języku C++ mamy do dyspozycji zestaw trzech operatorów logicznych.

Koniunkcję wyrażamy za pomocą operatora `&&`. Zwróci on wartość `true` wtedy i tylko wtedy, gdy wszystkie argumenty mają wartość `true`.

Operator `||` reprezentuje alternatywę. Zwróci on wartość `true`, gdy przynajmniej jeden argument ma wartość `true`.

Istnieje również jednoargumentowy operator negacji `!`. Zwróci on wartość `true` dla argumentu o wartości `false` i `false` dla argumentu o wartości `true`.

```
bool t = true;
bool f = false;
bool wynik;

wynik = t && t;     // wynik = true
wynik = t && f;     // wynik = false
wynik = f && t;     // wynik = false
wynik = f && f;     // wynik = false

wynik = t || t;     // wynik = true
wynik = t || f;     // wynik = true
wynik = f || t;     // wynik = true
wynik = f || f;     // wynik = false

wynik = !t;         // wynik = false
wynik = !f;         // wynik = true
```

Warunki połączone operatorami:

```
#include <iostream>

using namespace std;

int main()
{
    int x = 5;

    if (x < 10 && x > 3) {
            std::cout << "Liczba jest większa niż 3 i mniejsza ni
    }
}
```

Instrukcja `else` oraz `else if`

Język C++ ma również dwie dodatkowe instrukcje: `else` oraz `else if`. Pierwsza pozwala wykonać alternatywny blok kodu, kiedy kryterium instrukcji warunkowej nie zostanie spełnione. Druga jest połączeniem obu wspomnianych instrukcji, a więc wykona blok kodu, jeśli instrukcja warunkowa nie zostanie spełniona, ale jeśli zostanie spełniony nowy warunek.

```
#include <iostream>

using namespace std;

int main()
{
    int x = 11;

    if (x < 10 && x > 3) {
        std::cout << "Liczba jest większa niż 3 i mniejsza niż 10
    } else {
        std::cout << "Liczba jest mniejsza lub równa 3 lub większ
    }
}
```

Przykładowy problem

Przyjrzyjmy się bardziej rozbudowanemu przykładowi. Wyobraźmy sobie taką sytuację: pewna uczelnia prowadzi roczne kursy przygotowawcze dla przyszłych studentów. Nie są one obowiązkowe, jednak w rekrutacji dają pewną przewagę nad osobami, które w nich nie uczestniczyły. Uczelnia oblicza punkty rekrutacyjne w taki sposób, że wynik egzaminu maturalnego na poziomie rozszerzonym mnoży przez 3, a podstawowym przez 2. Finaliści olimpiad zdobywają dodatkowe 100 punktów, a laureaci 150.

Osoby uczestniczące w kursie niezależnie uczestnictwa w olimpiadach oraz od poziomu, na jakim zdają egzamin maturalny ze wskazanego przedmiotu, mnożą jego wynik przez 3 oraz otrzymują dodatkowo 150 punktów.

Korzystając z samej instrukcji warunkowej, musielibyśmy stworzyć szereg warunków:

```
int main() {
   wynik = punkty;

    if (kurs == false) {
        if (rozszerzenie == true) {
            wynik = punkty * 3;
        }

        if (rozszerzenie == false) {
            wynik = punkty * 2;
        }

        if (olimpiada == true) {
            wynik += 100;

            if (olimpiadaLaureat == true) {
                wynik += 50;
            }
        }
    }

    if (kurs == true) {
        wynik = punkty * 3 + 150;
    }
}
```

Wykorzystując operatory logiczne, możemy znacznie uprościć kod obliczający wynik rekrutacji:

```
int main() {
    wynik = punkty;

    if (kurs == false && rozszerzenie == true) {
        wynik = punkty * 3;
    }

    if (kurs == false && rozszerzenie == false) {
        wynik = punkty * 3;
    }

    if (kurs == false && olimpiada == true && olimpiadaLaureat ==
        wynik += 100;
    }

    if (kurs == false && olimpiada == true && olimpiadaLaureat ==
        wynik += 150;
    }

    if (kurs == true) {
        wynik = punkty * 3 + 150;
    }
}
```

Wykorzystując instrukcje `else` oraz `else if`, otrzymamy taki kod:

```
int main() {
    wynik = punkty;

    if (kurs == false && rozszerzenie == false && olimpiada == tr
        wynik = punkty * 2 + 100;
        if (olimpiadaLaureat == true) {
            wynik += 50;
        }
    } else if (kurs == false && rozszerzenie == false && olimpiad
        wynik = punkty * 2;
    } else if (kurs == false && rozszerzenie == true && olimpiada
        wynik = punkty * 3 + 100;
        if (olimpiadaLaureat == true) {
            wynik += 50;
```

```
        }
    } else if (kurs == false && rozszerzenie == true && olimpiada
        wynik = punkty * 3;
    } else {
        wynik = punkty * 3 + 150;
    }
}
```

Słownik

> **operator logiczny**
>
> operator pozwalający na wykonywanie podstawowych operacji algebry Boole'a, takich jak koniunkcja, alternatywa czy negacja
>
> **zagnieżdżenie**
>
> umieszczenie jednej operacji wewnątrz drugiej

Schemat interaktywny

Polecenie 1

Matka i syn planują zakup biletów lotniczych. Podstawowa cena biletu oferowanego przez linie lotnicze wynosi `cena` zł. Dopłata za przewożenie bagażu cięższego niż `limitMasy` kg wynosi `doplataNadbagaz` zł. Ubezpieczenie bagażu to dodatkowa kwota w wysokości `ubezpieczenieBagaz` zł – jeśli jednak pasażerowie dopłacają już za nadbagaż, to ubezpieczenie kosztuje ich tylko `ubezpieczenieNadbagazu` zł. Osoby podróżujące klasą biznesową mają cenę zarówno nadbagażu jak i ubezpieczenia wliczoną w koszt biletu, którego cena jest wyższa o `doplataBiznes` zł od podstawowej. Rodzina nie leci klasą biznes. Matka bierze ze sobą `wagaBagazuMatka` kg bagażu. Syn bierze ze sobą jedynie `wagaBagazuSyn` kg bagażu. Oboje decydują się na ubezpieczenie.

Ile zapłacą za bilety?

Przetestuj swój program dla następujących danych:

```
cena = 250
limitMasy = 6
doplataNadbagaz = 70
ubezpieczenieBagaz = 40
ubezpieczenieNadbagazu = 25
doplataBiznes = 150
wagaBagazuMatka = 9
wagaBagazuSyn = 4
klasaBiznes = fałsz
ubezpieczenie = prawda
```

Specyfikacja:

Dane:

- *cena* – liczba naturalna
- *limitMasy* – liczba naturalna
- *doplataNadbagaz* – liczba naturalna
- *ubezpieczenieBagaz* – liczba naturalna

- *ubezpieczenieNadbagazu* – liczba naturalna
- *doplataBiznes* – liczba naturalna
- *wagaBagazuMatka* – liczba naturalna
- *wagaBagazuSyn* – liczba naturalna
- *klasaBiznes* – zmienna logiczna
- *ubezpieczenie* – zmienna logiczna

Wynik:

Program wyświetla informację o sumarycznej cenie podróży zapisanej w zmiennej *doZaplaty*.

```
#include <iostream>

using namespace std;

int main() {
    int cena = 250;
    bool klasaBiznes = false;
    bool ubezpieczenie = true;
    int wagaBagazu = 9;
    int doZaplaty = 0;

    // tutaj dopisz kod

    cout << "Do zaplaty: " << doZaplaty;
```

```

```

Polecenie 2

Zapoznaj się z prezentacją i porównaj z nią swoje rozwiązanie.

1

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/Punf1NbMT

Matka i syn planują zakup biletów lotniczych. Podstawowa cena biletu oferowanego przez linie lotnicze to 250 zł. Dopłata za przewożenie bagażu cięższego niż 6 kg wynosi 70 zł. Ubezpieczenie bagażu to dodatkowa kwota w wysokości 40 zł – jeśli jednak pasażerowie dopłacają już za nadbagaż, to ubezpieczenie kosztuje ich tylko 25 zł. Osoby podróżujące klasą biznesową mają cenę zarówno nadbagażu, jak i ubezpieczenia wliczoną w koszt biletu, którego cena jest wyższa o 150 zł od podstawowej.

2

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/Punf1NbMT

Rodzina nie zdecydowała się na podróż klasą biznes, zatem cenę ich biletów możemy obliczyć przy pomocy następującego warunku:

```
if (klasaBiznes == false &&
    ubezpieczenie == true &&
    wagaBagazu > limitMasy) {
    doZaplaty = cena + 95;
} else if (klasaBiznes == false &&
    wagaBagazu > limitMasy) {
    doZaplaty = cena + 70;
} else if (klasaBiznes == false &&
    ubezpieczenie == true) {
    doZaplaty = cena + 40;
} else {
    doZaplaty = cena;
}
```

3

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/Punf1NbMT

Kobieta zabierze ze sobą 9 kg bagażu i wykupiła ubezpieczenie, a zatem:

```
cena = 250;
klasaBiznes = false;
ubezpieczenie = true;
wagaBagazu = 9;
```

4

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/Punf1NbMT

Podstawmy te wartości do zmiennych w naszym warunku:

```
if (false == false &&
    true == true &&
    9 > 6) {
    doZaplaty = 250 + 95;
} else if (false == false
&&
    9 > 6) {
    doZaplaty = 250 + 70;
} else if (false == false
&&
    true == true) {
    doZaplaty = 250 + 40;
} else {
    doZaplaty = 250;
}
```

5

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/Punf1NbMT

Wykonując dalsze obliczenia, otrzymamy:

```
if (true && true && true)
{
    doZaplaty = 345;
} else if (true && true) {
    doZaplaty = 320;
} else if (true && true) {
    doZaplaty = 290;
} else {
    doZaplaty = 250;
}
```

6

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/Punf1NbMT

Po wykonaniu koniunkcji każdy z warunków zostanie spełniony. Wobec tego program wykona następnie instrukcje zamieszczone wewnątrz pierwszego warunku, który zostanie spełniony, i pominie kolejne. To ważne, aby pamiętać o tej zasadzie, układając warunki.

```
if (true) {
    doZaplaty = 345;
} else if (true) {
    doZaplaty = 320;
} else if (true) {
    doZaplaty = 290;
} else {
    doZaplaty = 250;
}
```

7

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/Punf1NbMT

Obliczmy też cenę biletu syna. Zabierze on ze sobą jedyne 4 kg bagażu, ale także go ubezpieczy:

```
cena = 250;
klasaBiznes = false;
ubezpieczenie = true;
wagaBagazu = 4;
```

8

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/Punf1NbMT

Podstawmy wartości w warunku:

```
if (false == false &&
    true == true &&
    4 > 6) {
    doZaplaty = 345;
} else if (false == false
&&
    4 > 6) {
    doZaplaty = 320;
} else if (false == false
&&
    true == true) {
    doZaplaty = 290;
} else {
    doZaplaty = 250;
}
```

9

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/Punf1NbMT

Wykonawszy obliczenia, otrzymamy:

```
if (true && true && false)
{
    doZaplaty = 345;
} else if (true && false)
{
    doZaplaty = 320;
} else if (true && true) {
    doZaplaty = 290;
} else {
    doZaplaty = 250;
}
```

10

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/Punf1NbMT

Tym razem dopiero trzeci warunek jest prawdą – kobieta zapłaci za swój bilet 345 zł, a jej syn 290 zł.

```
if (false) {
    doZaplaty = 345;
} else if (false) {
    doZaplaty = 320;
} else if (true) {
    doZaplaty = 290;
} else {
    doZaplaty = 250;
}
```

Problem 1

Uczniowie przygotowali program, który temperaturę podaną w Fahrenheitach przelicza na stopnie Celsjusza i na tej podstawie ocenia stan zdrowia.

Spróbuj napisać program przedstawiony w schemacie blokowym za pomocą języka C++. Przetestuj jego działanie dla temperatury 247 F.

Specyfikacja:

Dane:

temperatura – temperatura ciała podana w stopniach Fahrenheita; liczba rzeczywista

Wynik:

Na standardowym wyjściu pojawia się jedno z określeń opisujących stan zdrowia: *hipotermia*, *temperatura w normie*, *gorączka*, *hipertermia*, *hiperpireksja*

```
#include <iostream>

using namespace std;

int main() {

    // tutaj dopisz kod

    cout<<"Ocena temperatury ciała: "<<twojaT;
}
```

```
1
```

Sprawdź się

Pokaż ćwiczenia: 

Ćwiczenie 1

Napisz program, który sprawdzi, czy podana liczba x jest mniejsza od 10. Przetestuj jego działanie dla liczby `x = 7`.

Specyfikacja:

Dane:

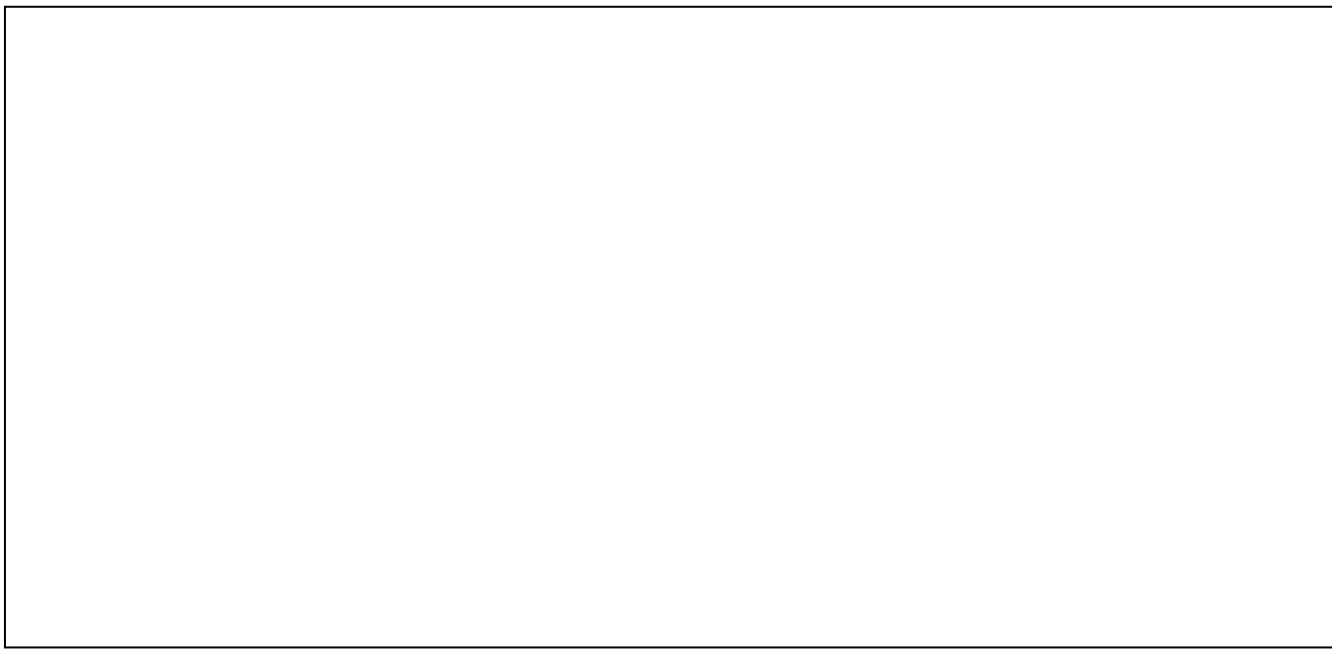
x - liczba naturalna

Wynik:

Program wyświetla komunikat: *Liczba x jest mniejsza od 10*; w przeciwnym przypadku program nie wyświetla niczego.

Twoje zadania

1. Program sprawdza, czy liczba 7 jest mniejsza od 10.



Ćwiczenie 2

Napisz program, który sprawdzi, czy podana liczba x jest parzysta, większa od 50, ale mniejsza od 100. Sprawdź jego działanie dla x = 73.

Specyfikacja:

Dane:

x - liczba naturalna

Wynik:

Gdy liczba *x* jest parzysta, większa od 50 i mniejsza od 100, program wyświetla komunikat: *Liczba spełnia warunki*; w przeciwnym przypadku program wyświetla komunikat: *Liczba nie spełnia warunków*.

Twoje zadania

1. Program sprawdza, czy liczba 73 spełnia podane warunki.

Ćwiczenie 3

Uczniowie pisali sprawdzian z informatyki, z którego mogli uzyskać maksymalnie 100 punktów. Punktacja przedstawiała się następująco:

- 0–39 pkt – ndst
- 40–54 pkt – dop
- 55–69 pkt – dst
- 70–84 pkt – db
- 85–98 pkt – bdb
- 99–100 pkt – cel

Napisz program, który przypisze ocenę do podanej liczby punktów.

Przetestuj jego działanie dla ucznia, który zdobył 22 punkty.

Specyfikacja:

Dane:

p - liczba naturalna z zakresu od 0 do 100

Wynik:

Program wyświetla, w zależności od liczby uzyskanych punków, jedno ze słów: *ndst*, *dop*, *dst*, *db*, *bdb* lub *cel*.

Twoje zadania

1. Program sprawdza, czy uczeń, zdobywając 22 punkty, uzyskał ocenę niedostateczną.

Dla nauczyciela

Autor: Maurycy Gast
Przedmiot: Informatyka

Temat: Instrukcja warunkowa w języku C++

Grupa docelowa:

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

Podstawa programowa:

Cele kształcenia – wymagania ogólne

I. Rozumienie, analizowanie i rozwiązywanie problemów na bazie logicznego i abstrakcyjnego myślenia, myślenia algorytmicznego i sposobów reprezentowania informacji.

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.

III. Posługiwanie się komputerem, urządzeniami cyfrowymi i sieciami komputerowymi, w tym: znajomość zasad działania urządzeń cyfrowych i sieci komputerowych oraz wykonywania obliczeń i programów.

Treści nauczania – wymagania szczegółowe

I. Rozumienie, analizowanie i rozwiązywanie problemów.

Zakres podstawowy. Uczeń:

1) planuje kolejne kroki rozwiązywania problemu, z uwzględnieniem podstawowych etapów myślenia komputacyjnego (określenie problemu, definicja modeli i pojęć, znalezienie rozwiązania, zaprogramowanie i testowanie rozwiązania).

3) wyróżnia w problemie podproblemy i charakteryzuje: metodę połowienia, stosuje podejście zachłanne i rekurencję;

4) porównuje działanie różnych algorytmów dla wybranego problemu, analizuje algorytmy na podstawie ich gotowych implementacji;

5) sprawdza poprawność działania algorytmów dla przykładowych danych.

Zakres rozszerzony. Uczeń spełnia wymagania określone dla zakresu podstawowego, a ponadto:

2) do realizacji rozwiązania problemu dobiera odpowiednią metodę lub technikę algorytmiczną i struktury danych;

3) objaśnia dobrany algorytm, uzasadnia poprawność rozwiązania na wybranych przykładach danych i ocenia jego efektywność;

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);

2) do realizacji rozwiązań problemów prawidłowo dobiera środowiska informatyczne, aplikacje oraz zasoby, wykorzystuje również elementy robotyki;

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):

- Przeanalizujesz składnię instrukcji warunkowej w języku C++.
- Wyjaśnisz, jak zapisać warunki wielokrotnie złożone.
- Wykorzystasz instrukcję warunkową do rozwiązywania prostych problemów.

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.

Przebieg lekcji

Przed lekcją:

1. **Przygotowanie do zajęć.** Nauczyciel loguje się na platformie i udostępnia e-materiał: „Instrukcja warunkowa w języku C++". Nauczyciel prosi uczniów o zapoznanie się z treściami w sekcji „Przeczytaj".

Faza wstępna:

1. Nauczyciel inicjuje rozmowę wprowadzającą w temat lekcji. Przedstawia cele zajęć oraz kryteria sukcesu.

Faza realizacyjna:

1. **Praca z multimedium.** Nauczyciel wyświetla zawartość sekcji „Schemat interaktywny". Uczniowie w parach próbują rozwiązać polecenia. Na forum klasy omawiają swoje propozycje. Następnie zapoznają się z Prezentacją multimedialną.
2. **Ćwiczenie umiejętności.** Uczniowie wykonują indywidualnie ćwiczenia nr 1–2. Po wykonaniu każdego z nich następuje omówienie rozwiązania przez nauczyciela.

Faza podsumowująca:

1. Nauczyciel zadaje pytania podsumowujące, np.
    - w jakich sytuacjach instrukcja warunkowa nie jest potrzebna?
    - w jakich sytuacjach instrukcja warunkowa jest konieczna?
    - w jakiej sytuacji kolejność występowania warunków ma znaczenie?

Praca domowa:

1. Uczniowie wykonują ćwiczenie 3 z sekcji „Sprawdź się".

Materiały pomocnicze:

- Oficjalna dokumentacja techniczna dla języka C++.
- 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.

Wskazówki metodyczne:

- Nauczyciel może wykorzystać multimedium w sekcji „Schemat interaktywny" do pracy przed lekcją. Uczniowie zapoznają się z jego treścią i przygotowują do pracy na zajęciach w ten sposób, żeby móc samodzielnie rozwiązać zadania dołączone do e-materiału „Instrukcja warunkowa w języku C++".