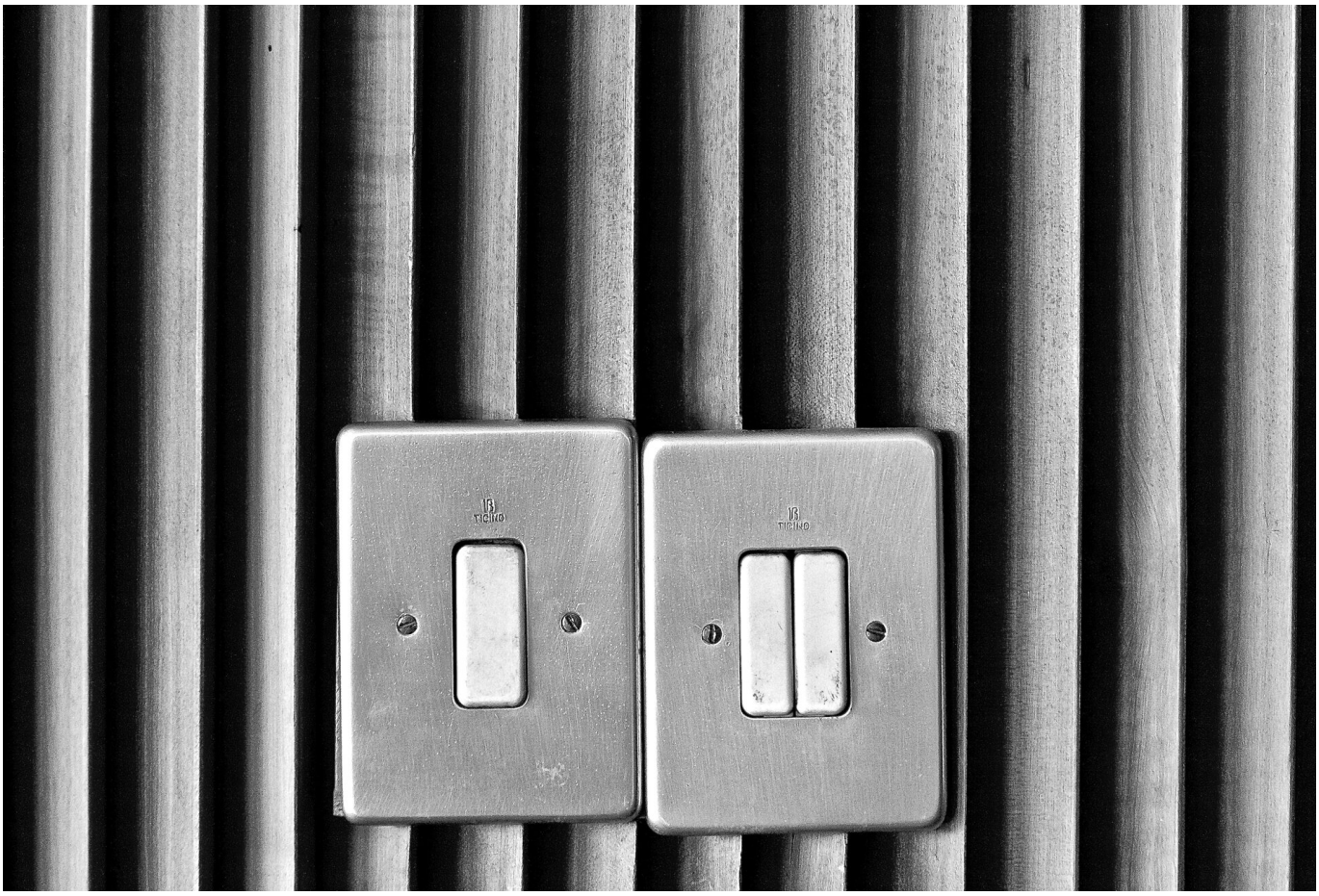
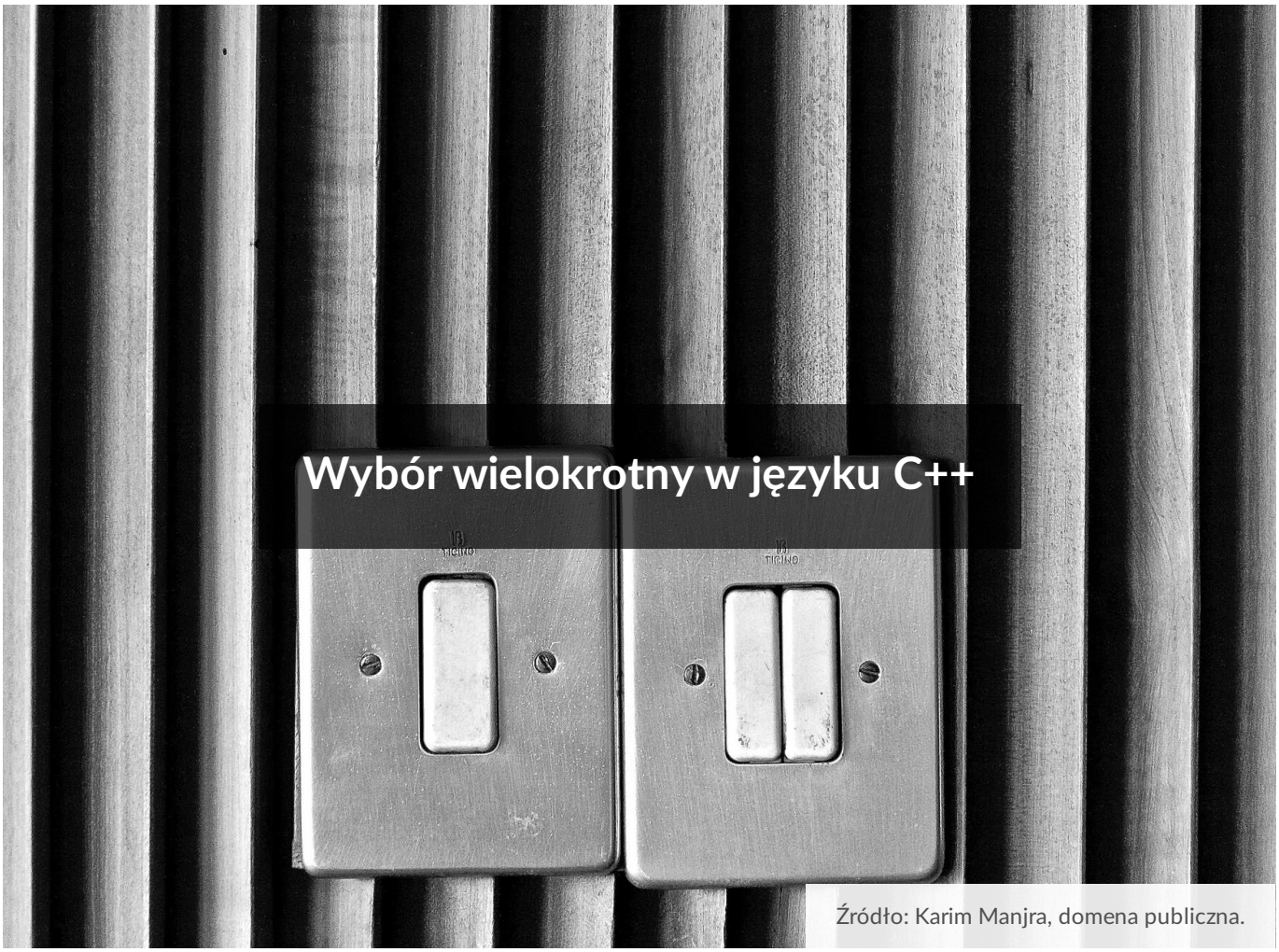


# Wybór wielokrotny w języku C++




Źródło: Karim Manjra, domena publiczna.

Instrukcja warunkowa sprawdza się bardzo dobrze, gdy trzeba wybrać do zrealizowania jeden z dwóch scenariuszy opisanych w kodzie programu. Sprawy komplikują się wraz ze zwiększaniem liczby scenariuszy. Lepszym rozwiązaniem okazuje się wówczas instrukcja wielokrotnego wyboru – `switch`.

W tym e-materiale poznasz instrukcję wielokrotnego wyboru w języku C++.

Ciekawi cię, jak wygląda omawiana kwestia w innych językach programowania? Możesz się z nimi zapoznać w dwóch pozostałych lekcjach z tej serii:

- Wybór wielokrotny w języku Java,
- Wybór wielokrotny w języku Python.

**Twoje cele**

- Zastosujesz wyrażenia trójargumentowe.
- Poznasz składnię instrukcji wielokrotnego wyboru.
- Przekonasz się, że instrukcja `switch` w pewnych przypadkach okazuje się bardziej efektywna niż instrukcja warunkowa.
- Wykorzystasz instrukcję `switch` do rozwiązania prostych zadań.

# Przeczytaj

## Wybór wielokrotny w języku C++

### Trójargumentowe wyrażenie wyboru

Używane jest w sytuacjach, gdy wystarczy zwrócić prostą informację zależną od jednego warunku. W takim przypadku rozbudowywanie kodu jest niecelowe.

```
warunek_logiczny ? wartość_gdy_prawda : wartość_gdy_fałsz;
wartość_gdy_prawda if warunek_logiczny else wartość_gdy_fałsz;
```

Instrukcja taka odpowiada poleceniom:

```
if warunek_logiczny:
    wartość_gdy_prawda
else:
    wartość_gdy_fałsz
```

Porównajmy kody dwóch funkcji. Funkcje przyjmują dwa argumenty. Każda z nich ma sprawdzić czy pierwsza liczba przekazana jako argument jest podzielna przez drugą.

Pierwsza funkcja wykorzystuje zwykłą konstrukcję `if / else`:

```
bool czy_podzielna_przez(int liczba, int dzielnik) {
    if (liczba % dzielnik == 0)
        return true;
    else
        return false;
}
```

Druga funkcja wykorzystuje wyrażenie trójargumentowe:

```
bool czy_podzielna_przez(int liczba, int dzielnik) {
    return liczba % dzielnik == 0 ? true : false;
}
```

Obie powyższe funkcje mogą być zapisane również następująco:

```
bool czy_podzielna_przez(int liczba, int dzielnik) {
    return liczba % dzielnik == 0
}
```

Innym przykładem zastosowania wyrażenia trójargumentowego jest podanie jednego z dwóch komunikatów, zależnych od spełnienia pewnego warunku.

```
int wartosc = 15;
cout << (wartosc > 20 ? "Dużo" : "Mało");
// Mało

wartosc = 23;
cout << (wartosc > 20 ? "Dużo" : "Mało");
// Dużo
```

## Instrukcja `switch` w języku C++

Instrukcja wielokrotnego wyboru w języku C++ rozpoczyna się od słowa kluczowego `switch`:

```
int numer_zawodnika = 2;

switch (numer_zawodnika) {
    case 0:
        // wypisz informacje o zawodniku 0.
        break;
    case 1:
        // wypisz informacje o zawodniku 1.
        break;
    case 2:
        // wypisz informacje o zawodniku 2.
        break;
    case 3:
        // wypisz informacje o zawodniku 3.
        break;
    case 4:
        // wypisz informacje o zawodniku 4.

```

```
        break;
}
```

Przedstawiliśmy sposób użycia instrukcji `switch` ze zmienną typu całkowitego. Nie ma jednak przeszkód, aby wykorzystać zmienne typu znakowego (`char`):

```
char litera = 'c';

switch (litera) {
    case 'a':
        // wartość zmiennej litera to 'a'
        break;
    case 'b':
        // wartość zmiennej litera to 'b'
        break;
    case 'c':
        // wartość zmiennej litera to 'c'
        break;
    case 'd':
        // wartość zmiennej litera to 'd'
        break;
    case 'e':
        // wartość zmiennej litera to 'd'
        break;
}
```

> **Ważne!**
>
> Definiując zmienne znakowe użyliśmy znaków apostrofu (' ), a nie cudzysłowów. W języku C++ stosujemy apostrofy, kiedy chcemy przypisać zmiennej typu `char` konkretny znak. Użycie cudzysłowu oznacza, że chodzi o ciąg znaków. Jest on zawsze zakończony tzw. znakiem pustym, mającym w kodzie ASCII wartość 0.
>
> Zmienna typu `char` może przechowywać tylko pojedyncze znaki. Zastosowanie cudzysłowów spowodowałoby dodanie znaku pustego.

## Polecenie `break`

Przedstawiając składnię komendy `switch` użyliśmy instrukcji `break`. Kończy ona zestaw poleceń towarzyszących konkretnemu przypadkowi (`case`). Gdy program napotka komendę `break`, przerywane jest wykonywanie całej instrukcji wyboru wielokrotnego.

Przyjrzyjmy się ostatniemu przykładowi. Gdy zacznie być przetwarzana linia 10, program sprawdzi, czy wartość zmiennej `litera` jest równa przypadkowi `case 'c'`. Ponieważ okaże się, że tak właśnie jest, rozpocznie się wykonywanie fragmentu kodu zapisanego w linii numer 11. Później pojawia się komenda `break`. Oznacza to zakończenie instrukcji `switch` i przejście za linię numer 19.

## Łączenie warunków

Wiadomo już, dlaczego stosujemy instrukcję `break`. Pomyśl, jak zachowa się poniższy program:

```
1  char litera = 'B';
2
3  switch(litera) {
4      case 'a':
5      case 'A':
6          // wartość zmiennej 'litera' to 'a' lub 'A'
7          break;
8      case 'b':
9      case 'B':
10         // wartość zmiennej 'litera' to 'b' lub 'B'
11         break;
12 }
```

Użycie dwóch słów kluczowych `case` spowoduje, że możemy dokonać wyboru na podstawie większej liczby wyrażeń. Warto pamiętać, że po wykonaniu instrukcji program będzie kontynuował sprawdzanie następnych warunków, chyba że natrafi na instrukcję `break`.
Skoro wartość zmiennej `litera` to 'B', wykona się linijka 10. Efekt byłby taki sam, gdyby `litera` była równa 'b'.

### Polecenie `default`

W instrukcji wyboru wielokrotnego mamy do dyspozycji polecenie `default`. Oznacza ono: „w każdym przypadku”.

Innymi słowy `default` towarzyszy instrukcjom wykonywanym w sytuacji, gdy wartość zmiennej przekazanej poleceniu `switch` nie odpowiada żadnemu przypadkowi. Mamy dzięki temu pewność, że program wykona fragment kodu przypisany poleceniu `default`. Oto przykład:

```
int numer_zawodnika = 40;

switch (numer_zawodnika) {
    case 0:
        // wypisz informacje o zawodniku z numerem 0.
        break;
    case 1:
        // wypisz informacje o zawodniku z numerem 1.
        break;
    case 2:
        // wypisz informacje o zawodniku z numerem 2.
        break;
    default:
        // nie ma zawodnika o podanym numerze
}
```

Wartość zmiennej `numer_zawodnika` nie pojawiła się w opisie żadnego przypadku, a zatem wykonane zostaną instrukcje towarzyszące słowu kluczowemu `default`. Jeżeli ten fragment kodu jest zapisany na końcu instrukcji wielokrotnego wyboru, nie ma potrzeby używania komendy `break`. W innym przypadku jest to niezbędne.

## Słownik

> **break**
>
> używane w języku C++ polecenie służące do opuszczenia instrukcji `switch` po wykonaniu sekwencji komend odpowiadającej konkretnemu przypadkowi (`case`)

# Schemat interaktywny

> **Polecenie 1**
>
> Napisz program, który wyświetla informację zwrotną w zależności od oceny otrzymanej ze sprawdzianu.
>
> **Specyfikacja problemu:**

**Dane:**

- ocena - liczba naturalna z przedziału <2, 6>

**Wynik:**

Komunikat dopasowany do danej oceny:

- Ocena 6: „Wspaniale!”
- Oceny 5 i 4: „Bardzo dobrze”
- Ocena 3: „Spróbuj powtórzyć najważniejsze informacje”
- Ocena 2: „Podejdź do sprawdzianu jeszcze raz”
- Ocena 1: „Ten materiał sprawił ci trudność, może porozmawiaj na ten temat z osobą prowadzącą przedmiot?”

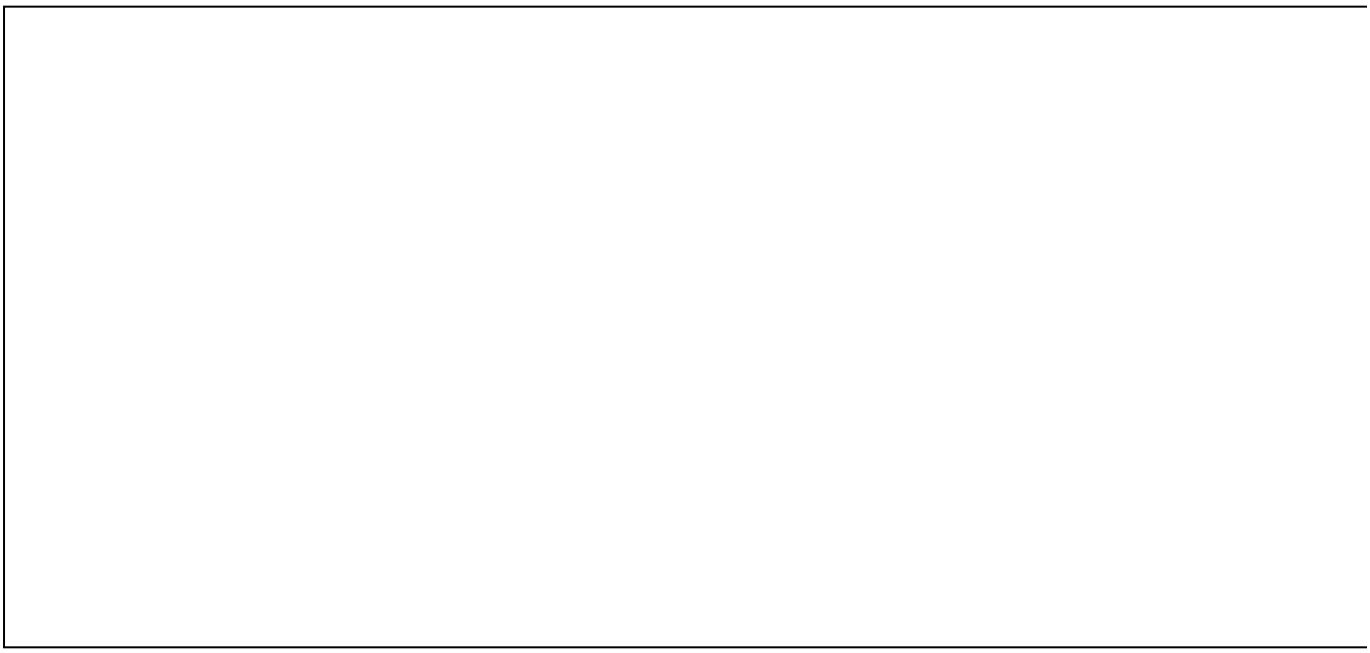
```
#include <iostream>

int main () {

    // Tutaj dodaj kod. Żeby coś wypisać, użyj polecenia: std::cout
}
```

```

```



**Polecenie 2**

Porównaj swoje rozwiązanie z filmem.


Film dostępny pod adresem /preview/resource/RrasWvxnypz30



Film nawiązujący do treści lekcji dotyczący wyboru wielokrotnego w instrukcji switch w C++.

**Polecenie 3**

Uczniowie napisali prosty kalkulator, który wykonuje cztery podstawowe działania na dwóch liczbach. Zapoznaj się ze schematem i napisz ten program, używając języka C++. Przetestuj jego działanie dla różnicy 72 i 48.



```
#include <iostream>

int main () {

    // Tutaj dodaj kod. Żeby coś wypisać, użyj polecenia: std::cout
}
```

```
1
```

1

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P8FnaYa5C

Napiszemy program, który będzie obliczał wyniki różnych działań wykonywanych na dwóch liczbach całkowitych.

2

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P8FnaYa5C

Zacznijmy od zbudowania szkieletu programu, czyli zadeklarowania funkcji głównej (`main`). W jej ciele zapiszemy wszystkie polecenia:

```
#include <iostream>

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

3

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P8FnaYa5C

Następnie zadeklarujemy dwie zmienne całkowite: `liczba1` oraz `liczba2`. Właśnie na nich będziemy wykonywać działania matematyczne:

```
#include <iostream>

```

```
int main() {
    int liczba1 = 0;
    int liczba2 = 0;

    return 0;
}
```

4

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P8FnaYa5C

Kolejnym etapem jest napisanie kodu odpowiedzialnego za komunikację programu z użytkownikiem. Powinien on podać dwie liczby. Odczytujemy je, korzystając ze strumienia `cin`:

```
#include <iostream>

int main() {
    int liczba1 = 0;
    int liczba2 = 0;

    std::cout << "Podaj pierwszą liczbę" << std::endl;
    std::cin >> liczba1;
    std::cout << "Podaj drugą liczbę" << std::endl;
    std::cin >> liczba2;


    return 0;
}
```

Materiał audio dostępny pod adresem:

5

https://zpe.gov.pl/b/P8FnaYa5C

Zdefiniujemy też zmienną typu `char` o nazwie `dzialanie`. Będzie ona przechowywać znak opisujący działanie matematyczne:

```
#include <iostream>

int main() {
    int liczba1 = 0;
    int liczba2 = 0;

    std::cout << "Podaj pierwszą liczbę" << std::endl;
    std::cin >> liczba1;
    std::cout << "Podaj drugą liczbę" << std::endl;
    std::cin >> liczba2;

    char dzialanie;

    return 0;
}
```

6

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P8FnaYa5C

Poinformujemy użytkownika, jaki znak przypisany jest konkretnemu działaniu matematycznemu i poprosimy go o wybranie operacji do wykonania.

Następnie pobierzemy z klawiatury znak wpisany przez użytkownika:

```
#include <iostream>

int main() {
    int liczba1 = 0;
```

```
    int liczba2 = 0;

    std::cout << "Podaj pierwszą liczbę" << std::endl;
    std::cin >> liczba1;
    std::cout << "Podaj drugą liczbę" << std::endl;
    std::cin >> liczba2;

    char dzialanie;

    std::cout << "Podaj znak działania, które chcesz wykonać na tych dwóch liczbach \n"
        "+ - dodawanie \n"
        "- - odejmowanie \n"
        "* - mnożenie \n"
        "/ - dzielenie \n"
    << std::endl;

    std::cin >> dzialanie;

    return 0;
}
```

7

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P8FnaYa5C

Deklarujemy zmienną `wynik`, która będzie przechowywać rezultat obliczeń matematycznych:

```
#include <iostream>

int main() {
    int liczba1 = 0;
```

```
    int liczba2 = 0;

    std::cout << "Podaj pierwszą liczbę" << std::endl;
    std::cin >> liczba1;
    std::cout << "Podaj drugą liczbę" << std::endl;
    std::cin >> liczba2;

    char dzialanie;

    std::cout << "Podaj znak działania, które chcesz wykonać na tych dwóch liczbach \n"
        "+ - dodawanie \n"
        "- - odejmowanie \n"
        "* - mnożenie \n"
        "/ - dzielenie \n" << std::endl;

    std::cin >> dzialanie;

    double wynik = 0;

    return 0;
}
```

8

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P8FnaYa5C

Przejdźmy do napisania szkieletu polecenia `switch`. Będzie ono wykorzystywać zmienną `dzialanie` jako parametr:

```
#include <iostream>

```

```cpp
int main() {
    int liczba1 = 0;
    int liczba2 = 0;

    std::cout << "Podaj pierwszą liczbę" << std::endl;
    std::cin >> liczba1;
    std::cout << "Podaj drugą liczbę" << std::endl;
    std::cin >> liczba2;

    char dzialanie;

    std::cout << "Podaj znak działania, które chcesz wykonać na tych dwóch liczbach \n"
        "+ - dodawanie \n"
        "- - odejmowanie \n"
        "* - mnożenie \n"
        "/ - dzielenie \n" << std::endl;

    std::cin >> dzialanie;

    double wynik = 0;

    switch (dzialanie) {

    }

    return 0;
}
```

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P8FnaYa5C

W ciele instrukcji `switch` zapisujemy sekwencje poleceń odpowiadających poszczególnym operacjom matematycznym. Bloki kodu do wykonania są wybierane na podstawie wartości zapisanej w zmiennej `dzialanie`. Wynik obliczeń przechowa zmienna `wynik`:

```
#include <iostream>

int main() {
    int liczba1 = 0;
    int liczba2 = 0;

    std::cout << "Podaj pierwszą liczbę" << std::endl;
    std::cin >> liczba1;
    std::cout << "Podaj drugą liczbę" << std::endl;
    std::cin >> liczba2;

    char dzialanie;

    std::cout << "Podaj znak działania, które chcesz wykonać na tych dwóch liczbach \n"
        "+ - dodawanie \n"
        "- - odejmowanie \n"
        "* - mnożenie \n"
        "/ - dzielenie \n" << std::endl;

    std::cin >> dzialanie;

    double wynik = 0;

    switch (dzialanie) {
        case '+': wynik = liczba1 + liczba2;
            break;
```

```
27          case '-': wynik =
liczba1 - liczba2;
28              break;
29          case '*': wynik =
liczba1 * liczba2;
30              break;
31          case '/': wynik =
liczba1 / liczba2;
32              break;
33      }
34
35      return 0;
36  }
```

10

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P8FnaYa5C

Ostatnią czynnością jest wyświetlenie wyniku.

Oto pełny kod programu:

```
1  #include <iostream>
2
3  int main() {
4      int liczba1 = 0;
5      int liczba2 = 0;
6
7      std::cout << "Podaj
pierwszą liczbę" <<
std::endl;
8      std::cin >> liczba1;
9      std::cout << "Podaj
drugą liczbę" <<
std::endl;
10     std::cin >> liczba2;
11
12     char dzialanie;
13
14     std::cout << "Podaj
znak działania, które
```

```
    chcesz wykonać na tych
    dwóch liczbach \n"
15          "+ - dodawanie \n"
16          "- - odejmowanie
    \n"
17          "* - mnożenie \n"
18          "/ - dzielenie \n"
    << std::endl;
19
20      std::cin >> dzialanie;
21
22      double wynik = 0;
23
24      switch (dzialanie) {
25          case '+': wynik =
    liczba1 + liczba2;
26              break;
27          case '-': wynik =
    liczba1 - liczba2;
28              break;
29          case '*': wynik =
    liczba1 * liczba2;
30              break;
31          case '/': wynik =
    liczba1 / liczba2;
32              break;
33      }
34
35      std::cout << wynik;


36
37      return 0;
38  }
```

# Sprawdź się

Pokaż ćwiczenia: 

**Ćwiczenie 1**

Napisz program, który sprawdzi, czy pewna liczba należy do pięcioelementowego zbioru {5, 10, 15, 20, 25}. Jeżeli tak jest, należy wyświetlić komunikat „tak”. W przeciwnym wypadku powinien zostać wypisany komunikat „nie”.

## Twoje zadania

1. Program sprawdza, czy liczba 20 należy do zbioru {5, 10, 15, 20, 25}.

```cpp
#include <iostream>

int main() {
    int x = 20;

    // Tutaj dodaj własny kod.
    // Aby wypisać komunikat użyj polecenia std::cout
}
```

```
1
```

### Ćwiczenie 2

Napisz program, który sprawdzi, czy różnica między wartościami zmiennych `wiek_osoby_b` i `wiek_osoby_a` wynosi 1, 2 lub 3, czy też jest inna. W każdym z wymienionych przypadków należy wyświetlić odpowiedni komunikat.

## Twoje zadania

1. Program sprawdza czy różnica między wartościami zmiennych `wiek_osoby_b` i `wiek_osoby_a` wynosi 1, 2 lub 3 (czy też jest inna), a następnie wyświetla stosowny komunikat.

```
#include <iostream>

int main() {
    int wiek_osoby_a = 13;
    int wiek_osoby_b = 15;

    // Tutaj dodaj własny kod.
    // Wypisz "Osoba b jest starsza o 1 rok", jeżeli różnica między wiek_osoby_b a wiek_osoby_a jest równa 1
    // Wypisz "Osoba b jest starsza o 2 lata", jeżeli różnica między wiek_osoby_b a wiek_osoby_a jest równa 2
    // Wypisz "Osoba b jest starsza o 3 lata", jeżeli różnica między wiek_osoby_b a wiek_osoby_a jest równa 3
    // Wypisz "Ta osoba jest starsza", jeżeli różnica między
```

```

```

### Ćwiczenie 3

Małgosia otrzymuje wynagrodzenie za wykonywanie prac domowych. Każda czynność jest opłacana według określonej stawki. Napisz program, który na podstawie wysokości otrzymanej kwoty poinformuje, czym zajmowała się Małgosia.

## Twoje zadania

1. Program ma za zadanie sprawdzić, jaką czynność wykonała Małgosia, jeżeli dostała 150 zł i wyświetlić odpowiedni komunikat.

```cpp
#include <iostream>

int main() {
    int wynagrodzenie = 150;

    // Tutaj dodaj własny kod.
    // Wypisz "Małgosia podlała kwiatki", jeżeli wynagrodzenie jest równe 20
    // Wypisz „Małgosia sprzątała kuchnię", jeżeli wynagrodzenie jest równe 50
    // Wypisz "Małgosia myła okna", jeżeli wynagrodzenie jest równe 70
    // Wypisz "Małgosia posprzątała piwnicę", jeżeli wynagrodzenie jest równe 120
```

```cpp

```

# Dla nauczyciela

**Autor:** Maurycy Gast
**Przedmiot:** Informatyka

**Temat: Wybór wielokrotny w języku C++**

**Grupa docelowa:**

Liceum ogólnokształcące i technikum, liceum ogólnokształcące, technikum, zakres 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 rozszerzony. Uczeń spełnia wymagania określone dla zakresu podstawowego, a ponadto:
> >
> > > 2) stosuje zasady programowania strukturalnego i obiektowego w rozwiązywaniu problemów;

**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):**

- Zastosujesz wyrażenia trójargumentowe.
- Poznasz składnię instrukcji wielokrotnego wyboru.
- Przekonasz się, że instrukcja `switch` w pewnych przypadkach okazuje się bardziej efektywna niż instrukcja warunkowa.
- Wykorzystasz instrukcję `switch` do rozwiązania prostych zadań.

**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: „Wybór wielokrotny w języku C++". Nauczyciel prosi uczniów o zapoznanie się z treściami w sekcji „Przeczytaj".

**Faza wstępna:**

1. Nauczyciel wyświetla uczniom temat zajęć oraz cele. Prosi, by na ich podstawie uczniowie sformułowali kryteria sukcesu.
2. **Rozpoznanie wiedzy uczniów.** Uczniowie tworzą pytania dotyczące tematu zajęć, na które odpowiedzą w trakcie lekcji.

**Faza realizacyjna:**

1. Uczniowie rozwiązują Polecenie 1 z sekcji „Schemat interaktywny".

2. **Ćwiczenie umiejętności.** Uczniowie, pracując w parach, wykonują ćwiczenie nr 1 z sekcji „Sprawdź się". Nauczyciel sprawdza poprawność pisanych kodów, porównuje je i omawia wraz z uczniami. Wskazuje najbardziej efektywne rozwiązanie.
3. Nauczyciel dzieli uczniów na grupy czteroosobowe. Uczniowie wykonują ćwiczenie nr 2: „Napisz program, który sprawdzi, czy różnica między wartościami zmiennych `wiek_osoby_b` i `wiek_osoby_a` wynosi 1, 2 lub 3, czy też jest inna. W każdym z wymienionych przypadków należy wyświetlić odpowiedni komunikat." a następnie każda grupa wyznacza jedną osobę, którą wymienia się z inną grupą. W nowych zespołach uczniowie porównują swój kod i wybierają najbardziej efektywne rozwiązanie.

**Faza podsumowująca:**

1. Na koniec zajęć nauczyciel raz jeszcze wyświetla na tablicy temat lekcji i cele zawarte w sekcji „Wprowadzenie". W odniesieniu do ich realizacji dokonuje szczegółowej oceny rozwiązania zastosowanego przez wybranego ucznia.
2. Wybrany uczeń podsumowuje zajęcia, zwracając uwagę na nabyte umiejętności, omawia ewentualne problemy podczas rozwiązania ćwiczeń z programowania w języku C++.

**Praca domowa:**

1. Uczniowie wykonują ćwiczenie nr 3: „Małgosia otrzymuje wynagrodzenie za wykonywanie prac domowych. Każda czynność jest opłacana według określonej stawki. Napisz program, który na podstawie wysokości otrzymanej kwoty poinformuje, czym zajmowała się Małgosia." 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:**

- Treści w sekcji „Schemat interaktywny" można wykorzystać na lekcji jako podsumowanie i utrwalenie wiedzy uczniów.