Proste projekty i funkcje w języku C++

Źródło: Pixabay, domena publiczna.

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

Z pojęciem funkcji spotkaliśmy się już na lekcjach matematyki. Wiemy, jaką pełnią rolę w informatyce. Funkcje w programach opisują, jak na podstawie danych wejściowych obliczyć pewną wartość – nie wystarczy jednak definicja funkcji, wymagane jest również podanie jej argumentów. Więcej na ten temat znajdziesz w e-materiale Proste projekty i funkcje.

W tym e-materiale przyjrzymy się z wykorzystaniu funkcji w implementacji prostego projektu w języku C++.

Implementację w pozostałych językach programowania znajdziesz w e-materiałach:

- Proste projekty i funkcje w języku Java,
- Proste projekty i funkcje w języku Python.

Więcej zadań? Przejdź do e-materiału Proste projekty i funkcje – ćwiczenia.

> **Twoje cele**
>
> - Przeanalizujesz informacje dotyczące tworzenia i wywoływania funkcji w języku C++.

- Zapiszesz algorytm rozwiązywania równania kwadratowego, wykorzystując język programowania C++.
- Wykonasz kilka ćwiczeń sprawdzających twoją wiedzę na temat funkcji.

Przeczytaj

Rozwiązywanie równania kwadratowego – implementacja w języku C++

Zaimplementujemy w języku C++ wersję stabilną algorytmu rozwiązania równania kwadratowego. Najpierw zdefiniujemy funkcję o nazwie `rozwiazRownanie()`, która będzie przyjmowała parametry `a`, `b`, `c` (współczynniki równania kwadratowego będące liczbami rzeczywistymi), a następnie zwracała rozwiązanie równania kwadratowego, informowała o braku rozwiązania lub wyświetlała komunikat, że dane równanie nie jest równaniem kwadratowym.

Funkcja ma zwracać łańcuch znaków, dlatego przed jej nazwą należy dodać:

```
string
```

Kolejną rzeczą, o której musimy pamiętać w związku z użyciem łańcucha znaków `string`, jest dodanie odpowiedniej biblioteki oraz deklaracja przestrzeni nazw:

```
#include <string>

using namespace std;
```

Utworzone ciało funkcji wygląda tak:

```
#include <iostream>
#include <string>

using namespace std;

string rozwiazRownanie() {

}
```

Kolejnym krokiem jest zadeklarowanie ciała funkcji `rozwiazRownanie()` oraz zmiennej łańcuchowej o nazwie `wynik`. Będzie ona przechowywała rozwiązanie równania kwadratowego lub informację o jego braku.

```
string rozwiazRownanie(double a, double b, double c) {
    string wynik = "";

    return wynik;
}
```

Następnie musimy sprawdzić, czy podane wartości argumentów są poprawne – czy wartość współczynnika a jest różna od 0. W przypadku, gdy wartość współczynnika a jest równa 0, funkcja `rozwiazRownanie()` powinna zakończyć swoje działanie i zwrócić wynik, że podane wartości współczynników definiują równanie liniowe, a nie kwadratowe.

```
string rozwiazRownanie(double a, double b, double c) {
    string wynik = "";

    if (a == 0) {
        wynik = "Nie jest to funkcja kwadratowa, tylko liniowa po
        return wynik;
    }

    return wynik;
}
```

Wykluczyliśmy przypadek, gdy `a = 0`. W kolejnych krokach zajmiemy się szukaniem rozwiązania równania kwadratowego.

```
string rozwiazRownanie(double a, double b, double c) {
    string wynik = "";

    if (a == 0) {
        wynik = "Nie jest to funkcja kwadratowa, tylko liniowa po
        return wynik;
    } else {
        float delta = b * b - 4 * a * c;

        }
    }

    return wynik;
}
```

Utworzyliśmy zmienną `delta`, która przechowuje wartość wyróżnika trójmianu kwadratowego.

```
string rozwiazRownanie(double a, double b, double c) {
    string wynik = "";

    if (a == 0) {
        wynik = "Nie jest to funkcja kwadratowa, tylko liniowa po
        return wynik;
    } else {
        float delta = b * b - 4 * a * c;

        if (delta < 0) { // BRAK ROZWIĄZAŃ
            wynik = "Równanie nie ma rozwiązania";
        } else if (delta == 0) { // JEDNO ROZWIĄZANIE
            double x = -b / (2 * a);
            wynik = "Równanie ma jedno rozwiązanie, x = " + to_st

        }
    }

    return wynik;
}
```

Następnie, w zależności od wartości delty, wykonają się kolejne instrukcje. Od jej wartości zależy, czy równanie kwadratowe będzie miało rozwiązanie.

Przypomnijmy zależność:

Jeśli delta jest **mniejsza od zera**, równanie **nie ma rozwiązań**. Jeśli jest **równa zero**, równanie ma **jedno rozwiązanie**. Jeśli zaś jest **większa od zera**, równanie ma **dwa rozwiązania**.

```
string rozwiazRownanie(double a, double b, double c) {
    string wynik = "";

    if (a == 0) {
        wynik = "Nie jest to funkcja kwadratowa, tylko liniowa po
        return wynik;
    } else {
        float delta = b * b - 4 * a * c;
```

```
9
10          if (delta < 0) { // BRAK ROZWIĄZAŃ
11              wynik = "Równanie nie ma rozwiązania";
12          } else if (delta == 0) { // JEDNO ROZWIĄZANIE
13              double x = -b / (2 * a);
14              wynik = "Równanie ma jedno rozwiązanie, x = " + to_st
15          } else if (delta > 0) { // DWA ROZWIĄZANIA
16          }
17      }
18
19      return wynik;
20  }
```

Przejdźmy do wariantu, gdy `delta` jest większa od 0. Deklarujemy trzy zmienne:

- `double viete = c / a;` – zmienna ta wykorzysta zależności wynikające ze wzorów Viète'a (omówionych w e-materiale Proste projekty i funkcje);
- `double x1 = 0;` – zmienna ta będzie przechowywać pierwsze rozwiązanie równania kwadratowego;
- `double x2 = 0;` – zmienna ta będzie przechowywać drugie rozwiązanie równania kwadratowego.

Następnie sprawdzana jest wartość współczynnika b i w zależności od jego znaku wykonywane są operacje wyliczania wartości pierwiastków.

```
1  string rozwiazRownanie(double a, double b, double c) {
2      string wynik = "";
3
4      if (a == 0) {
5          wynik = "Nie jest to funkcja kwadratowa, tylko liniowa po
6          return wynik;
7      } else {
8          float delta = b * b - 4 * a * c;
9
10         if (delta < 0) { // BRAK ROZWIĄZAŃ
11             wynik = "Równanie nie ma rozwiązania";
12         } else if (delta == 0) { // JEDNO ROZWIĄZANIE
13             double x = -b / (2 * a);
14             wynik = "Równanie ma jedno rozwiązanie, x = " + to_st
15         } else if (delta > 0) { // DWA ROZWIĄZANIA
16             double viete = c / a;
```

```
            double x1 = 0;
            double x2 = 0;

            if (b > 0) {
                x1 = (-b - sqrt(delta)) / (2 * a);
                x2 = viete / x1;
            } else {
                x2 = (-b + sqrt(delta)) / (2 * a);
                x1 = viete / x2;
            }

            wynik = "Równanie ma dwa rozwiązania: \nx1 = " + to_s
        }
    }

    return wynik;
}
```

Używamy funkcji `sqrt()`, dlatego musimy pamiętać o dodaniu biblioteki:

```
#include <cmath>
```

Funkcja jest gotowa. Należy teraz napisać część kodu odpowiedzialną za interakcję z użytkownikiem oraz wywołać funkcję dla zadanych przez użytkownika parametrów.

```
int main() {

    double a, b, c;

    cout << "Podaj wartość współczynnika a:" << endl;
    cin >> a;
    cout << "Podaj wartość współczynnika b:" << endl;
    cin >> b;
    cout << "Podaj wartość współczynnika c:" << endl;
    cin >> c;

    cout << rozwiazRownanie(a, b, c) << endl;

    return 0;
}
```

Oto cały kod programu:

```
#include <iostream>
#include <string>
#include <cmath>

using namespace std;

string rozwiazRownanie(double a, double b, double c) {
    string wynik = "";

    if (a == 0) {
        wynik = "Nie jest to funkcja kwadratowa, tylko liniowa po
        return wynik;
    } else {
        float delta = b * b - 4 * a * c;

        if (delta < 0) { // BRAK ROZWIĄZAŃ
            wynik = "Równanie nie ma rozwiązania";
        } else if (delta == 0) { // JEDNO ROZWIĄZANIE
            double x = -b / (2 * a);
            wynik = "Równanie ma jedno rozwiązanie, x = " + to_st
        } else if (delta > 0) { // DWA ROZWIĄZANIA
            double viete = c / a;
            double x1 = 0;
            double x2 = 0;

            if (b > 0) {
                x1 = (-b - sqrt(delta)) / (2 * a);
                x2 = viete / x1;
            } else {
                x2 = (-b + sqrt(delta)) / (2 * a);
                x1 = viete / x2;
            }

            wynik = "Równanie ma dwa rozwiązania: \nx1 = " + to_s
        }
    }

    return wynik;
```

```
}

int main() {

    double a, b, c;

    cout << "Podaj wartość współczynnika a:" << endl;
    cin >> a;
    cout << "Podaj wartość współczynnika b:" << endl;
    cin >> b;
    cout << "Podaj wartość współczynnika c:" << endl;
    cin >> c;

    cout << rozwiazRownanie(a, b, c) << endl;

    return 0;
}
```

Wyjście konsoli:

```
Podaj wartość współczynnika a:
2
Podaj wartość współczynnika b:
1
Podaj wartość współczynnika c:
-3
Równanie ma dwa rozwiązania:
x1 = -1.500000
oraz
x2 = 1.000000
Program ended with exit code: 0
```

Słownik

> **implementacja**
>
> realizacja algorytmu w konkretnym języku programowania
> **parametr**
>
> zmienna zdefiniowana przy deklaracji metody lub funkcji

Schemat interaktywny

Polecenie 1

Zapoznaj się ze schematem interaktywnym. Napisz program za pomocą języka C++ i przetestuj jego działanie dla funkcji $2x^2 + 8x - 10 = 0$.

Specyfikacja problemu:

Dane:

- `a`, `b`, `c` – liczby całkowite

Wynik:

Program na wyjściu standardowym zwróci rozwiązanie równania kwadratowego lub jego brak.

Polecenie 2

Dodaj do swojego programu komentarze tak, żeby był zrozumiały dla osoby, która nie potrafi programować.

Sprawdź się

Pokaż ćwiczenia: 

Ćwiczenie 1

Zmodyfikuj kod programu rozwiązującego równanie kwadratowe tak, aby wartość delta była wyliczana w osobnej funkcji o nazwie `obliczDelta()`. Funkcja ma zwracać wartość typu `double` i jako argumenty przyjmować `a`, `b` oraz `c`.

Przetestuj jego działanie dla następujących wartości:

```
a = 3
b = 2
c = -5
```

Specyfikacja problemu:

Dane:

- `a`, `b`, `c` – liczby rzeczywiste

Wynik:

Program wyświetla wynik pierwiastki kwadratowego.

Przykładowe wyjście:

```
Równanie ma dwa rozwiązania:
x1 = -1.666667
oraz
x2 = 1.000000
```

Twoje zadania

1. Program ma rozwiązywać równanie kwadratowe o zadanych parametrach a, b, c, wynoszących odpowiednio: 3, 2, -5. Wartość delty ma być uzyskiwana za pomocą funkcji `obliczDelta`.

Ćwiczenie 2

Napisz program, który sprawdzi, czy z odcinków o danych długościach można utworzyć trójkąt. Przetestuj jego działanie dla odcinków o długości 9, 10, 12.

Specyfikacja problemu:

Dane:

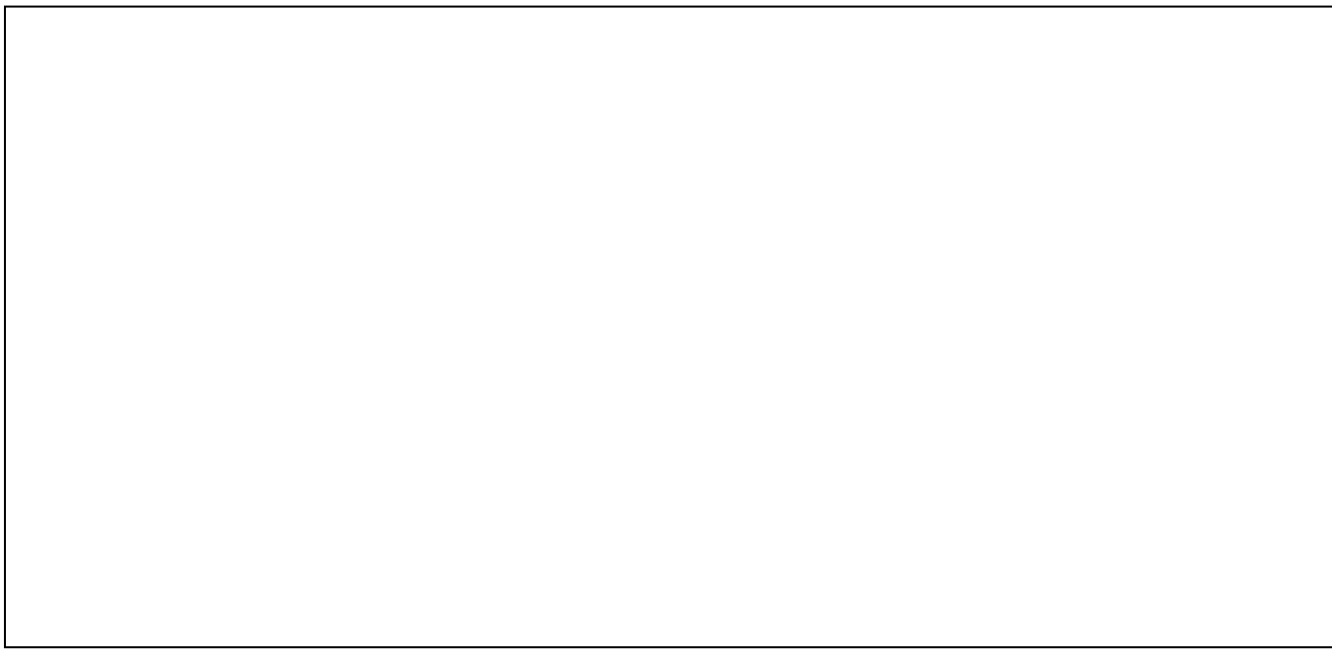
- `a`, `b`, `c` – liczby naturalne; długości odcinków

Wynik:

Program na wyjściu standardowym
wypisze: `Z PODANYCH ODCINKÓW MOŻNA UTWORZYĆ TRÓJKĄT` lub
`Z PODANYCH ODCINKÓW NIE MOŻNA UTWORZYĆ TRÓJKĄTA`.

Twoje zadania

1. Program sprawdza, czy z odcinków o podanych długościach można utworzyć trójkąt, a następnie wypisuje odpowiedni komunikat.



Ćwiczenie 3

Program realizuje stabilny algorytm rozwiązywania równania kwadratowego. Zmodyfikuj go o wyliczenie rozwiązania i zapisz go w zmiennej `wynik`, w przypadku gdy równanie jest liniowe (`a == 0`).

Przetestuj jego działanie dla wartości:

```
a = 0
b = 2
c = -5
```

Specyfikacja problemu:

Dane:

- `a, b, c` – zmienne typu *double*

Wynik:

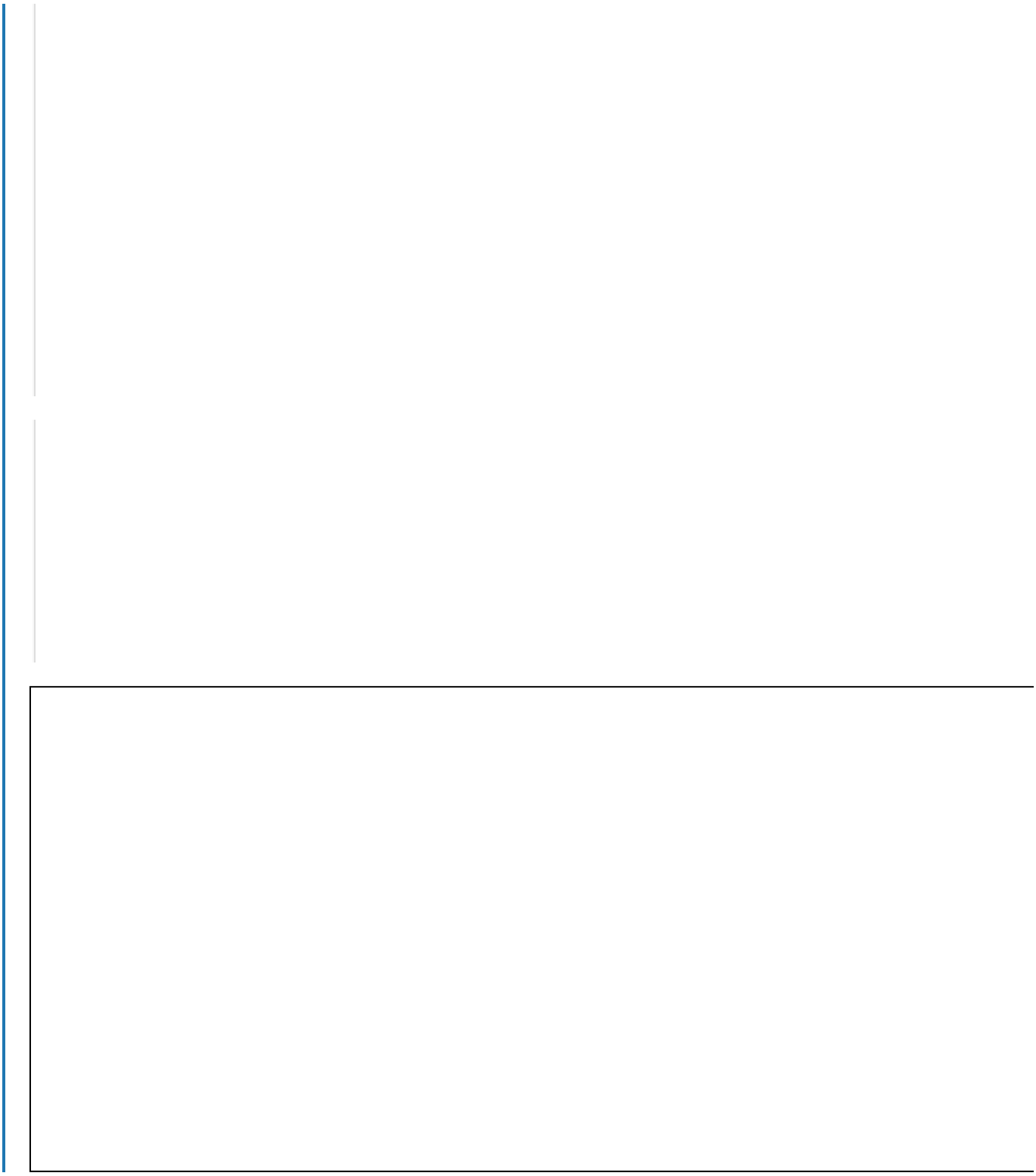
Program na wyjściu standardowym wypisze rozwiązanie wraz z komunikatem: „Nie jest to funkcja kwadratowa, tylko liniowa, ponieważ `a = 0`, rozwiązanie funkcji liniowej = [wyliczone rozwiązanie]”.

Przykładowe wyjście:

```
Nie jest to funkcja kwadratowa, tylko liniowa ponieważ a = 0, r
```

Twoje zadania

1. Program, w przypadku gdy wartość współczynnika a będzie równa 0, ma wyliczyć rozwiązanie oraz wypisać komunikat: "Nie jest to funkcja kwadratowa, tylko liniowa, ponieważ `a = 0`, rozwiązanie funkcji liniowej = [wyliczone rozwiązanie]".



Dla nauczyciela

Autor: Maurycy Gast
Przedmiot: Informatyka

Temat: Proste projekty i funkcje w języku C++

Grupa docelowa:

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

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);

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 informacje dotyczące tworzenia i wywoływania funkcji w języku C++.
- Zapiszesz algorytm rozwiązywania równania kwadratowego, wykorzystując język programowania C++.

- Wykonasz kilka ćwiczeń sprawdzających twoją wiedzę na temat funkcji.

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: „Proste projekty i funkcje w języku C++". Uczniowie zapoznają się z treściami w sekcji „Przeczytaj".

Faza wstępna:

1. Przedstawienie tematu i celów zajęć.

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.** Nauczyciel czyta polecenie nr 1 z sekcji „Schemat interaktywny". Prosi uczniów, aby wykonali je w parach. Następnie wybrana osoba

prezentuje propozycję odpowiedzi, a pozostali uczniowie ustosunkowują się do niej. Nauczyciel w razie potrzeby uzupełnia ją, udziela też uczniom informacji zwrotnej.
3. **Ćwiczenie umiejętności.** Uczniowie wykonują indywidualnie ćwiczenie nr 1 z sekcji „Sprawdź się", a następnie porównują swoje odpowiedzi z kolegą lub koleżanką.

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 wykonują ćwiczenie 2 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 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:

- Uczniowie mogą wykorzystać multimedium w sekcji „Schemat interaktywny" do przygotowania się do lekcji powtórkowej.