Operacje wejścia i wyjścia w języku C++

Źródło: Christina Morillo, domena publiczna.

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

Jak komunikować się z napisanym programem? Służą do tego poznane już przez nas operacje wejścia i wyjścia. Pozwalają one na dwustronną komunikację użytkownika z programem. Obejmują wszystkie czynności, od odczytywania przycisków z klawiatury do wyświetlania filmów na ekranie.

W tym e-materiale zapoznamy się z operacjami wejścia i wyjścia w języku C++.

Charakterystykę operacji wejścia i wyjścia w innych językach programowania znajdziesz w e-materiałach:

- Operacje wejścia i wyjścia w języku Java,
- Operacje wejścia i wyjścia w języku Python.

> **Twoje cele**
>
> - Zbadasz zastosowania operacji wejścia i wyjścia w C++.
> - Przeanalizujesz, jak poprawnie i efektywnie używać operacji wejścia i wyjścia.
> - Rozwiążesz podstawowe problemy związane z operacjami wejścia i wyjścia.

Przeczytaj

Załóżmy, że tworzymy program do obliczania pola figur geometrycznych. Napisaliśmy już większość kodu, jednak wciąż nie możemy zaimplementować rozwiązania, które pozwalałoby na komunikację z użytkownikiem.

Strumienie

Chcąc rozwiązać ten problem, musimy poznać strumienie. Można je sobie wyobrazić jako dane płynące od źródła do celu. Istnieje wiele celów i źródeł strumieni. Może to być ekran, klawiatura, plik lub zewnętrzne urządzenie.

W języku C++ wyróżniamy trzy typy strumieni:

- strumienie wejściowe (wczytują dane),
- strumienie wyjściowe (wypisują dane),
- strumienie uniwersalne (wczytują i wypisują dane).

Strumienie predefiniowane

Aby uniknąć sytuacji, w której za każdym razem musimy odpowiednio programować strumień, aby wyświetlał podane wartości w odpowiednim formacie, w C++ mamy możliwość użycia strumieni predefiniowanych. W tym celu dołączamy bibliotekę `iostream`. Jest to biblioteka standardowa dla języka programowania C++. Obsługuje operacje wejścia i wyjścia oraz ma kilka innych zastosowań. Bibliotekę dołączymy do kodu, dodając na jego początku linijkę `#include <iostream>.`

W języku C++ mamy do dyspozycji cztery strumienie predefiniowane. Nas najbardziej interesują dwa z nich, czyli:

- `std::cout` – strumień wyjścia,
- `std::cin` – strumień wejścia.

Warto jednak również wspomnieć o dwóch pozostałych:

- `std::cerr` – strumień błędów,
- `std::clog` – strumień błędów, szczególnie wydajny w przypadku dużej ilości danych.

Strumienie te są rzadziej używane. Często wiązane są z plikiem na dysku, do którego przesyłają dane o błędach, które wystąpiły.

Operatory << i >>

W zależności od strumienia, którego chcemy użyć, zastosujemy odpowiednio `<<` lub `>>`. Jak się za chwilę przekonamy, wskazują one – podobnie jak strzałki – kierunek przepływu strumienia: jeśli chcemy pobrać dane, stosujemy `>>`, a w przypadku, gdy chcemy je wypisać – używamy operatora `<<`.

Strumień wyjścia – `std::cout`

Stosujemy za każdym razem, gdy chcemy coś wyświetlić, np. wynik programu obliczającego wartość wyrażenia.

Załóżmy, że chcemy, aby nasz program wyświetlił wynik obliczenia wzoru na pole trójkąta przy danych, które wcześniej zdefiniowaliśmy w programie. Kod wyglądałby następująco:

```
int podstawa = 5;
int wysokosc = 10;
double poleTrojkata = podstawa * wysokosc / 2;

std::cout << poleTrojkata;
```

Program ten wypisałby `25`, czyli pole trójkąta o podstawie `5` i wysokości `10`.

Strumień wejścia – `std::cin`

Strumień wejścia jest stosowany w przypadku, gdy chcemy wczytać dane. Możemy za jego pomocą wprowadzić dane, które mają zostać użyte we wzorze.

Załóżmy, że chcemy, aby nasz program policzył pole prostokąta, korzystając z danych wprowadzonych przez użytkownika. Kod wyglądałby następująco:

```
int bok1;
int bok2;
int poleProstokata;

std::cin >> bok1;
std::cin >> bok2;

poleProstokata = bok1 * bok2;
```

Teraz połączymy operacje wejścia i wyjścia, aby stworzyć program, który poprosi użytkownika o wprowadzenie danych, a następnie obliczy pole trapezu i wypisze wynik.

```
int podstawa1;
int podstawa2;
int wysokosc;
std::cout << "Podaj pierwszą podstawę trapezu: ";
std::cin >> podstawa1;

std::cout << "Podaj drugą podstawę trapezu: ";
std::cin >> podstawa2;

std::cout << "Podaj wysokość trapezu: ";
std::cin >> wysokosc;

double poleTrapezu = (bok1 + bok2) * wysokosc / 2;

std::cout << "Pole trapezu wynosi: ";
std::cout << poleTrapezu;
```

W ten sposób zastosowaliśmy oba strumienie w jednym kodzie. Tekst wprowadziliśmy, poprzedzając i kończąc go cudzysłowem – jest to łańcuch znaków.

Kod ten można jednak uprościć. Zamiast wielokrotnie powtarzać instrukcje `std::cout` i `std::cin`, możemy raz jeszcze użyć odpowiednio operatora `<<` lub `>>`.

Wprowadzanie danych do programu może zostać zapisane w następujący sposób:

```
std::cout << "Podaj boki oraz wysokość trapezu: ";
std::cin >> bok1 >> bok2 >> wysokosc;
```

Po wpisaniu instrukcji `std::cin` możemy w tym samym wierszu wielokrotnie użyć operatora `>>` do wprowadzenia kolejnych zmiennych.

Taką samą metodę można wykorzystać, używając `cout`.

```
std::cout << "Pole trapezu wynosi: " << poleTrapezu;
```

Słownik

strumień wejścia

strumień używany do wprowadzania danych do programu

strumień wyjścia

strumień używany do wyprowadzania danych z programu

Prezentacja multimedialna

Polecenie 1
Napisz program, który obliczy pole trapezu. Za wysokość trapezu przyjmij wartość 4.

Specyfikacja

Dane:

- `podstawa1` – liczba zmiennoprzecinkowa dodatnia; długość pierwszej podstawy
- `podstawa2` – liczba zmiennoprzecinkowa dodatnia; długość drugiej podstawy
- `wysokosc` – liczba zmiennoprzecinkowa dodatnia; wysokość trapezu

Wynik:

Program prezentuje na standardowym wyjściu pole trapezu.

Wskazówka:

Wzór na pole trapezu: $P = \frac{(a+b)\cdot h}{2}$, gdzie a i b to podstawy trapezu, h to jego wysokość.

Polecenie 2
Porównaj swoje rozwiązanie z prezentacją.

Polecenie 3

Sprawdź się

Pokaż ćwiczenia:

Ćwiczenie 1

Ćwiczenie 2

Ćwiczenie 3

Ćwiczenie 4

Zapoznaj się z kodem i wykonaj ćwiczenie.

```
cout << 12 cout << 14;
```

Ćwiczenie 5

Zapoznaj się z kodem i wykonaj ćwiczenie.

```
cin << a << b;
```

Ćwiczenie 6

Zapoznaj się z kodem i wykonaj ćwiczenie.

```
cout << 2; cout >> 7;
```

Ćwiczenie 7

Opisz krótko, za co odpowiada strumień wyjścia.

Ćwiczenie 8

Dla nauczyciela

Autor: Zespół autorski Contentplus.pl sp. z o.o.
Przedmiot: Informatyka

Temat: Operacje wejścia i wyjścia 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:
>
> 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):

- Zbadasz zastosowania operacji wejścia i wyjścia w C++.
- Przeanalizujesz, jak poprawnie i efektywnie używać operacji wejścia i wyjścia.
- Rozwiążesz podstawowe problemy związane z operacjami wejścia i wyjścia.

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: „Operacje wejścia i wyjścia w języku C++”. Uczniowie zapoznają się z treściami w sekcji „Przeczytaj”.

Faza wstępna:

1. Nauczyciel wyświetla i odczytuje temat lekcji oraz cele zajęć. Prosi uczniów o sformułowanie kryteriów sukcesu.

2. **Rozpoznanie wiedzy uczniów.** Uczniowie tworzą pytania dotyczące tematu zajęć, na które odpowiedzą w trakcie lekcji.

Faza realizacyjna:

1. **Praca z tekstem.** Uczniowie przystępują do cichego czytania tekstu e-materiału. Indywidualnie zapoznają się z treścią w sekcji „Przeczytaj".
2. Nauczyciel prosi, aby wybrany uczeń przeczytał polecenie nr 1 z sekcji „Prezentacja multimedialna". Następnie prosi uczniów, aby podzielili się na grupy i opracowali odpowiedzi. Po ustalonym wcześniej czasie przedstawiciel wskazanej (lub zgłaszającej się na ochotnika) grupy prezentuje propozycję rozwiązania, a pozostali uczniowie ustosunkowują się do niego. Nauczyciel w razie potrzeby uzupełnia je, udzielając także uczniom informacji zwrotnej.
3. **Ćwiczenie umiejętności.** Uczniowie wykonują indywidualnie ćwiczenie nr 1-5, a następnie porównują swoje odpowiedzi z kolegą lub koleżanką.

Faza podsumowująca:

1. Nauczyciel wyświetla na tablicy temat lekcji i cele zawarte w sekcji „Wprowadzenie". W kontekście ich realizacji podsumowuje przebieg zajęć, a także wskazuje mocne i słabe strony pracy uczniów.
2. 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".
2. Uczniowie wykonują ćwiczenie 6-8 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 „Prezentacja multimedialna" można wykorzystać jako materiał służący powtórzeniu materiału.