Funkcje w języku C++

- Wprowadzenie
- Film samouczek
- Przeczytaj
- Sprawdź się
- Dla nauczyciela

Źródło: Pixabay, domena publiczna.

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

Funkcje pozwalają podzielić aplikację na bloki logiczne. Można je później wielokrotnie wykorzystywać, bez konieczności wpisywania w różnych częściach programu tych samych sekwencji poleceń.

Umiejętność posługiwania się funkcjami znacząco upraszcza pisanie programów. Kod, w którym wykorzystano funkcje, staje się bardziej czytelny i łatwiejszy do przeanalizowania. W tym e-materiale przyjrzymy się tworzeniu funkcji w języku C++.

Więcej informacji o funkcjach znajdziesz w e-materiale Funkcje. Chcesz wiedzieć, jak wygląda omawiane zagadnienie w innych językach programowania? Zapoznaj się z e-materiałami:

- Funkcje w języku Java,
- Funkcje w języku Python.

> **Twoje cele**
>
> - Scharakteryzujesz budowę funkcji w języku C++.
> - Przeanalizujesz kilka przykładów, dzięki którym zrozumiesz, czym jest i jak działa funkcja.

- Wykonasz kilka zadań związanych z funkcjami w języku C++.

Film samouczek

Polecenie 1

Napisz program, którego zadaniem będzie policzenie sumy dwóch liczb.

Specyfikacja problemu:

Dane:

- `liczbaA, liczbaB` – liczby całkowite
- `suma` – funkcja obliczająca sumę dwóch liczb

Wynik:

Program na wyjściu standardowym zwróci sumę dwóch liczb.

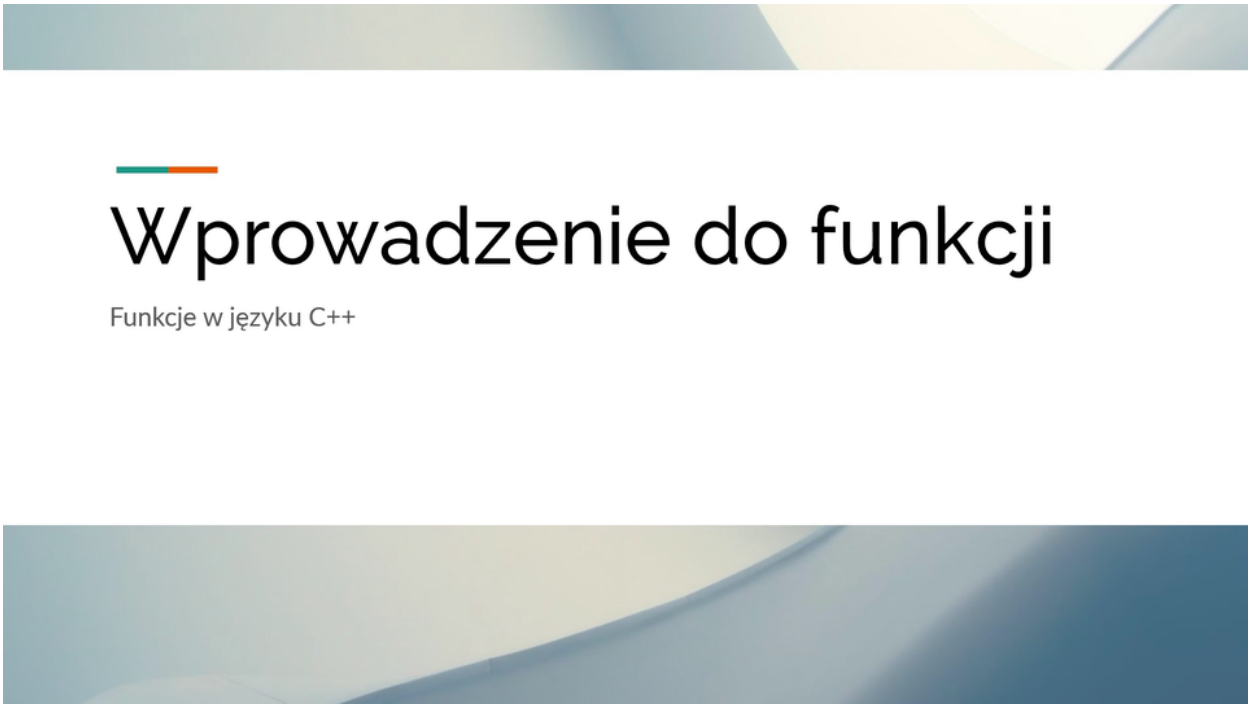
Lista kroków:

```
#include <iostream>

using namespace std;

// tutaj zdefiniuj funkcję

int main()
{
   int liczbaA = 5;
   int liczbaB = 12; // dla testów zmień liczby

   // tutaj dopisz kod funkcji main
}
```

```
1
```

Polecenie 2

Porównaj swoje rozwiązanie z filmem.



Film dostępny pod adresem /preview/resource/R1D3C7sXhyBlc

Źródło: Contentplus.pl Sp. z o.o., licencja: CC BY-SA 3.0.

Film nawiązujący do treści materiału: Wprowadzenie do funkcji.

Kod programu zaprezentowanego w filmie:

Źródło: Contentplus.pl Sp. z o.o., licencja: CC BY-SA 3.0.
Plik o rozmiarze 565.00 B w języku polskim

Polecenie 3

Zapisz notatkę, w której podsumujesz najważniejsze informacje przedstawione w filmie samouczku.

Przeczytaj

Funkcje bywają często nazywane podprogramami – i w istocie, są one „programami w programie”. Za pomocą funkcji wykonuje się zazwyczaj jedno konkretne zadanie wewnątrz większej aplikacji.

O przydatności funkcji najłatwiej jest przekonać się wtedy, gdy w różnych częściach programu wykonujemy tę samą operację (przykładowo, dodajemy do siebie kilka liczb). W takiej sytuacji możemy oczywiście powielać w odpowiednich miejscach kodu identyczne sekwencje poleceń. Rozsądniej jednak będzie napisać funkcję realizującą operację dodawania, a później wywoływać ją za każdym razem, gdy okaże się potrzebna.

Samo wywołanie gotowej funkcji jest bardzo proste. Wynika z tego, że raz napisana funkcja pozwala skrócić kod programu. Czyni go zarówno bardziej czytelnym, jak i optymalnym.

Co istotne, funkcję możemy wywoływać, podając różne argumenty (przykładowo: zestaw liczb, na których chcemy wykonać jakieś działanie). Lista argumentów jest ustalana podczas pisania programu. Jeżeli wydaje ci się to trochę zagmatwane, nie martw się. Wszystko zrozumiesz po przeanalizowaniu przykładów z dalszej części lekcji.

Składnia funkcji w języku C++

Zacznijmy od wyjaśnienia, gdzie należy umieścić deklarację funkcji. Dotychczas cały kod (pomijając dodawanie bibliotek i deklarowanie zmiennych globalnych) pisaliśmy wewnątrz funkcji głównej `main()`. W przypadku funkcji jest inaczej.

W języku C++ deklarację własnej funkcji umieszczamy przed blokiem funkcji głównej `main()`, tak jak w poniższym fragmencie kodu:

```
1  #include <iostream>
2
3  using namespace std;
4
5  double sumaDwochLiczb(double liczba1, double liczba2) {
6
7  }
8
9  int main {
10
11     return 0;
12 }
```

Przeanalizujmy ten wycinek programu. Podczas deklarowania funkcji obowiązują następujące zasady:

- Nazwa funkcji musi być ciągła (nie może zawierać spacji).
- Nazwa funkcji może zawierać litery, cyfry i znaki podkreślenia.
- Nazwa funkcji nie może rozpoczynać się od cyfry.
- Mamy swobodę w wyborze nazwy funkcji, ale dobrze jest, gdy opisuje ona wykonywane zadania. Przykładowo, funkcja obliczająca silnię mogłaby się nazywać `silnia()` albo `obliczSilnie()`.
- Przed nazwą funkcji trzeba określić typ zwracanej przez nią wartości (co konkretnie funkcja powinna zwracać) - w przypadku, gdy funkcja nic nie zwraca jako typ wskazujemy `void`.
- Funkcja musi zwracać wartość tego typu, który wskazano przed jej nazwą.
- W nawiasach (tuż po nazwie funkcji) możemy – ale nie musimy – umieścić listę argumentów. Poszczególne argumenty rozdzielamy przecinkiem, wskazujemy też osobno typ każdego z nich. W naszym przykładzie funkcja ma dwa argumenty typu `double`: `liczba1` i `liczba2`.

Na razie nasza funkcja nie potrafi zrobić nic: między lewym a prawym nawiasem klamrowym nie znalazło się żadne polecenie. Z nazwy funkcji wynika, że ma ona obliczać sumę dwóch liczb.

Napiszmy zatem ciało funkcji, czyli sprawmy, by dodawała ona do siebie dwie liczby, które podamy jako argumenty:

```
1 double sumaDwochLiczb(double liczba1, double liczba2) {
2
3     return liczba1 + liczba2;
4 }
```

Program jest gotowy. Wywołajmy naszą funkcję, używając konkretnych argumentów:

```
1 #include <iostream>
2
3 using namespace std;
4
5 double sumaDwochLiczb(double liczba1, double liczba2) {
6
7     return liczba1 + liczba2;
8 }
```

```
9
10 int main() {
11
12     cout << sumaDwochLiczb(5.45, 3.35);
13
14     return 0;
15 }
```

Program wyświetli wynik dodawania liczb: `8.8`.

Wywołaliśmy funkcję `sumaDwochLiczb(double liczba1, double liczba2)` z poziomu funkcji głównej`main()`. W nawiasach wpisaliśmy wprost wartości liczb, które chcieliśmy zsumować. Jako argumenty możemy także podawać zmienne:

```
1 #include <iostream>
2
3 using namespace std;
4
5 double sumaDwochLiczb(double liczba1, double liczba2) {
6
7     return liczba1 + liczba2;
8 }
9
10 int main() {
11
12     double zmienna1 = 5.45;
13     double zmienna2 = 3.35;
14
15     cout << sumaDwochLiczb(zmienna1, zmienna2);
16
17     return 0;
18 }
```

Warto zaznaczyć, że zmienne nie muszą nosić takich samych nazw jak argumenty w deklaracji funkcji. Po wywołaniu funkcji `sumaDwochLiczb()` zmienne o nazwach `zmienna1` i `zmienna2` zastąpiły odpowiednio zadeklarowane argumenty `liczba1` i `liczba2`.

Istnieją także funkcje wywoływane bez argumentów oraz takie, które nie zwracają żadnej wartości. Oto przykład:

```
1 #include <iostream>
2
3 using namespace std;
4
5 void wypiszCytat() {
6
7     cout << "To, że milczę, nie znaczy, że nie mam nic do powiedz
8 }
9
10 int main() {
11
12     wypiszCytat();
13
14     return 0;
15 }
```

W tym przypadku zamiast konkretnego typu zwracanej wartości (`bool, int, double...`) przed nazwą funkcji umieściliśmy słowo kluczowe `void`.

Wywołanie funkcji z ostatniego przykładu spowoduje wypisanie cytatu.

Jest to funkcja niezwracająca wartości i zarazem tzw. funkcja bezargumentowa. Możemy jednak dodać do niej argumenty. Niech użytkownik programu sam wprowadzi tekst, jaki chce wyświetlić:

```
1 #include <iostream>
2
3 using namespace std;
4
5 void wypiszCytat(char cytat[100]) {
6
7     cout << cytat;
8 }
9
10 int main() {
11
12     char cytat[100];
13
14     cin.getline(cytat, sizeof(cytat));
15
16     wypiszCytat(cytat);
```

```
17
18     return 0;
19 }
```

Musimy posłużyć się złożoną ze stu elementów tablicą typu `char`, ponieważ do odebrania więcej niż jednego słowa z klawiatury używamy funkcji wbudowanej `getline()`. Jako argumenty przyjmuje ona tablicę typu `char` oraz rozmiar obiektu (długość cytatu).

Słownik

> **funkcja bezargumentowa**
>
> funkcja, która nie przyjmuje żadnych argumentów
> **funkcja wbudowana**
>
> gotowa funkcja, wchodząca w skład jednej z bibliotek języka C++ (zgodnych ze standardem ISO/IEC 14882:2017); przykładowymi funkcjami wbudowanymi są: `length()`, `begin()`, `end()`, `sizeof()`

Sprawdź się

Pokaż ćwiczenia:

Ćwiczenie 1

Do kodu funkcji dopisz parametr, który spowoduje wyświetlenie łańcucha znaków. Przetestuj działanie programu dla napisu *INFORMATYKA*.

Specyfikacja:

Dane:

- `wypisz_zdanie` – funkcja do uzupełnienia
- `slowo` – łańcuch znaków, który ma zostać wypisany

Wynik:

Program ma wyświetlić zadany tekst.

Twoje zadania

1. W odpowiednim miejscu wpisz argument funkcji. Nadaj mu właściwą nazwę.

Ćwiczenie 2

Dopisz do istniejącego kodu funkcję służącą do obliczania kwadratu liczby całkowitej.

Specyfikacja:

Dane:

- `kwadrat` – funkcja do uzupełnienia
- `liczba_testowa` – liczba całkowita, której kwadrat ma zostać obliczony

Wynik:

Program ma obliczyć i wypisać kwadrat danej liczby całkowitej.

Twoje zadania

1. Program oblicza drugą potęgę liczby całkowitej. Jako liczbę testową podaj 5.

Ćwiczenie 3

Samoloty mają najczęściej dwie klasy pasażerskie (biznes i ekonomiczną). W klasie biznes są trzy miejsca w rzędzie, a w klasie ekonomicznej sześć. Zdefiniuj funkcję `miejscaWSamolocie(rzedyBiznes, rzedyEkonomiczna)`, która w wyniku poda liczbę miejsc w samolocie. Parametr `rzedyBiznes` oznacza liczbę rzędów foteli w klasie biznes, a parametr `rzedyEkonomiczna` oznacza liczbę rzędów foteli w klasie ekonomicznej. Przetestuj działanie funkcji, jeśli parametr `rzedyBiznes` może przyjmować wartości od 3 do 7, a parametr `rzedyEkonomiczna` od 15 do 40, w zależności od wersji.

Jeśli którykolwiek z parametrów będzie poza dopuszczalnym zakresem, funkcja powinna zwrócić wartość -1.

Specyfikacja:

Dane:

- `miejscaWSamolocie(rzedyBiznes, rzedyEkonomiczna)` – funkcja do zdefiniowania
- `rzedyBiznes` – liczba naturalna należąca do przedziału *<a, b>*
- `rzedyEkonomiczna` – liczba naturalna należąca do przedziału *<m, n>*

Wynik:

Program wyświetla, ile jest miejsc danej klasy w samolocie.

Twoje zadania

1. Program drukuje wartości zwracane przez funkcję `miejscaWSamolocie` dla przykładowych danych.

Dla nauczyciela

Autor: Maurycy Gast
Przedmiot: Informatyka

Temat: 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):

- Scharakteryzujesz budowę funkcji w języku C++.
- Przeanalizujesz kilka przykładów, dzięki którym zrozumiesz, czym jest i jak działa funkcja.

- Wykonasz kilka zadań związanych z funkcjami w języku C++.

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: „Funkcje w języku C++". Uczniowie mają zapoznać 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.** Nauczyciel prosi wybranego ucznia lub uczniów o przedstawienie sytuacji problemowej związanej z tematem lekcji.

Faza realizacyjna:

1. **Praca z tekstem.** Nauczyciel ocenia, na podstawie informacji na platformie, stan przygotowania uczniów do zajęć. Jeżeli jest ono niewystarczające, prosi wybraną osobę

o przedstawienie najważniejszych informacji z sekcji „Przeczytaj”.
2. **Praca z multimedium.** Nauczyciel wyświetla zawartość sekcji „Film samouczek”. Uczniowie pracując w parach, rozwiązują problem opisany w poleceniu 1. Następnie porównują swoje rozwiązanie z zaprezentowanym w filmie.
3. **Ćwiczenie umiejętności.** Uczniowie wykonują ćwiczenia nr 1 i 2 z sekcji „Sprawdź się”. Nauczyciel sprawdza poprawność wykonanych zadań, omawiając je wraz z uczniami.
4. Liga zadaniowa – uczniowie pracując w parach, wykonują ćwiczenie nr 3 z sekcji „Sprawdź się”, a następnie dzielą się swoimi wynikami przez porównywanie napisanego kodu z inną grupą, która również zakończyła zadanie.

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 piszą program, którego zadaniem będzie policzenie sumy dwóch zmiennych podanych przez użytkownika.

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:

- Multimedia w sekcjach: „Film samouczek”, „Przeczytaj”, „Sprawdź się” można wykorzystać jako materiał służący powtórzeniu materiału.