Obliczanie przybliżonej wielkości pola obszarów zamkniętych w języku C++

Obliczanie przybliżonej wielkości pola obszarów zamkniętych w języku C++

Źródło: Markus Spiske, domena publiczna.

W e-materiale Obliczanie przybliżonej wielkości pola obszarów zamkniętych dowiedzieliśmy się, jak przy użyciu metod numerycznych obliczyć pole powierzchni ograniczonej wykresem funkcji.

W tym e-materiale zaimplementujemy w języku C++ metody prostokątów i trapezów obliczające pole obszaru ograniczanego wykresem funkcji.

Implementacje w pozostałych językach programowania znajdziesz w e-materiałach:

- Obliczanie przybliżonej wielkości pola obszarów zamkniętych w języku Java,
- Obliczanie przybliżonej wielkości pola obszarów zamkniętych w języku Python.

Więcej zadań? Przejdź do e-materiału Obliczanie przybliżonej wielkości pola obszarów zamkniętych – zadania maturalne.

Twoje cele

- Użyjesz w praktyce wiedzy na temat metod prostokątów i trapezów.
- Przeanalizujesz algorytmy obliczania pola powierzchni pod wykresem metodami prostokątów i trapezów zapisane w języku C++.
- Zaimplementujesz poznane algorytmy służące do szacowania wartości pola obszaru zamkniętego w języku C++.

- Obliczysz pola powierzchni przykładowych figur.

Przeczytaj

Metoda prostokątów – implementacja w języku C++

> **Przykład 1**
>
> Napisz program, który wyznaczy pole powierzchni ograniczonej wykresem funkcji $f(x)$, osią OX oraz równoległymi do osi OY prostymi `xPoczatek` i `xKoniec` za pomocą metody prostokątów dla liczby przedziałów równej `liczbaProstokatow`. Wynik zaokrąglij do dwóch miejsc po przecinku.
>
> Program przetestuj dla funkcji `f(x)=x`, `xPoczatek = 0`, `xKoniec = 2` oraz liczby prostokątów równej 2.
>
> **Specyfikacja problemu:**
>
> *Dane*:
>
> - `xPoczatek` – liczba rzeczywista; początek przedziału
> - `xKoniec` – liczba rzeczywista; koniec przedziału
> - `liczbaProstokatow` – liczba naturalna dodatnia; liczba podprzedziałów
> - `f(x)` – funkcja rzeczywista zmiennej rzeczywistej
>
> *Wynik*:
>
> Program wypisuje przybliżoną wielkość pola ograniczonego wykresem funkcji `f(x)`, osią OX oraz prostymi `xPoczatek` i `xKoniec` wyznaczone metodą prostokątów z dokładnością do dwóch miejsc po przecinku przy danym przedziale.

Zacznijmy od dołączenia do programu potrzebnych bibliotek i deklaracji przestrzeni nazw. Ponieważ otrzymany wynik mamy zaokrąglić do dwóch miejsc po przecinku, wykorzystamy funkcję `round()` z biblioteki `<cmath>`.

```
#include <iostream>
#include <cmath>
using namespace std;
```

Deklarujemy również funkcję $f(x) = x$:

```
double f(double x) {
    return x;
}
```

W funkcji głównej programu umieścimy dwie zmienne typu `double`, które będą ograniczeniem rozważanego przedziału. Utworzymy również zmienną `wynik`, w której będziemy przechowywać przybliżoną wartość pola obszaru ograniczonego wykresem funkcji $f(x)$ i osią OX w zadanym przedziale `[xPoczatek, xKoniec]`. Inicjujemy ją wartością zerową:

```
#include <iostream>
#include <cmath>
using namespace std;

double f(double x) {
    return x;
}

int main() {
    double xPoczatek = 0.0, xKoniec = 2.0, wynik = 0.0;
}
```

Konieczne będzie również zdefiniowanie liczby prostokątów, to znaczy liczby odcinków, na które będziemy dzielić zadany przedział w celu uzyskania dokładniejszego przybliżenia.

```
#include <iostream>
#include <cmath>
using namespace std;

double f(double x) {
    return x;
}

int main() {
    double xPoczatek = 0.0, xKoniec = 2.0, wynik = 0.0;
    int liczbaProstokatow = 2;
}
```

Następnie obliczamy długość jednego z boków każdego prostokąta. W tym celu przedział `[xPoczatek, xKoniec]` dzielimy na odcinki o długości równej `(xKoniec - xPoczatek) : liczbaProstokatow`. Odcinki te będą podstawami powstałych prostokątów.

```
```

```cpp
#include <iostream>
#include <cmath>
using namespace std;

double f(double x) {
    return x;
}

int main() {
    double xPoczatek = 0.0, xKoniec = 2.0, wynik = 0.0;
    int liczbaProstokatow = 2;

    double podstawaProstokata =
        (xKoniec - xPoczatek) / liczbaProstokatow;
}
```

Tworzymy pętlę iteracyjną, która będzie odpowiadać za obliczanie pól kolejnych prostokątów. Ponieważ zdefiniowaliśmy wcześniej zmienną `liczbaProstokatow`, musimy zadbać, aby instrukcje w pętli wykonywały się taką samą liczbę razy.

```cpp
#include <iostream>
#include <cmath>
using namespace std;

double f(double x) {
    return x;
}

int main() {
    double xPoczatek = 0.0, xKoniec = 2.0, wynik = 0.0;
    int liczbaProstokatow = 2;

    double podstawaProstokata =
        (xKoniec - xPoczatek) / liczbaProstokatow;

    for (int i = 0; i < liczbaProstokatow; i++) {

    }
}
```

Wewnątrz pętli należy dodać do zmiennej `wynik` obliczone pole aktualnego `i`-tego prostokąta. Zauważ, że długością wysokości prostokąta jest wartość funkcji w początku przedziału będącego podstawą prostokąta. Jest to wariant metody prostokątów, który nazywamy **wariantem lewych prostokątów**.

```
#include <iostream>
#include <cmath>
using namespace std;

double f(double x) {
    return x;
}

int main() {
    double xPoczatek = 0.0, xKoniec = 2.0, wynik = 0.0;
    int liczbaProstokatow = 2;

    double podstawaProstokata =
        (xKoniec - xPoczatek) / liczbaProstokatow;

    for (int i = 0; i < liczbaProstokatow; i++) {
        wynik += podstawaProstokata * f(xPoczatek + podstawaProst
    }
}
```

Możemy zredukować liczbę zbędnych operacji mnożenia (obliczania pól prostokątów), poprzez wyłączenie wspólnego czynnika przed nawias. Dzięki temu wewnątrz pętli będą jedynie sumowane wysokości prostokątów, a po jej zakończeniu pomnożymy otrzymaną sumę przez długość podstawy prostokąta.

```
#include <iostream>
#include <cmath>
using namespace std;

double f(double x) {
    return x;
}

int main() {
    double xPoczatek = 0.0, xKoniec = 2.0, wynik = 0.0;
```

```
    int liczbaProstokatow = 2;

    double podstawaProstokata =
        (xKoniec - xPoczatek) / liczbaProstokatow;

    for (int i = 0; i < liczbaProstokatow; i++) {
        wynik += f(xPoczatek + podstawaProstokata * i);
    }

    wynik *= podstawaProstokata;
}
```

Zaokrąglimy, a następnie wypiszemy wynik stanowiący pole ograniczone wykresem funkcji $f(x)$ w zadanym przedziale za pomocą metody prostokątów.

```
#include <iostream>
#include <cmath>
using namespace std;

double f(double x) {
    return x;
}

int main() {
    double xPoczatek = 0.0, xKoniec = 2.0, wynik = 0.0;
    int liczbaProstokatow = 2;

    double podstawaProstokata =
        (xKoniec - xPoczatek) / liczbaProstokatow;

    for (int i = 0; i < liczbaProstokatow; i++) {
        wynik += f(xPoczatek + podstawaProstokata * i);
    }

    wynik *= podstawaProstokata;
    wynik = round(wynik * 100) / 100;
    cout << wynik << endl;

    return 0;
}
```

Tak napisany program wyznaczy pole obszaru ograniczonego wykresem funkcji $f(x)$ w zadanym przedziale, czyli w przypadku tego przykładu wartość 1.

Słownik

metoda prostokątów

metoda szacowania wartości całki za pomocą sumy pól prostokątów, będących przybliżeniem obszaru ograniczonego wykresem funkcji i osią OX

metoda trapezów

metoda szacowania wartości całki za pomocą sumy pól trapezów, będących przybliżeniem obszaru ograniczonego wykresem funkcji i osią OX

Symulacja interaktywna

Polecenie 1

Przeanalizuj prezentację, w której zostały przedstawione kolejne kroki rozwiązania zaprezentowanego zadania. Następnie porównaj pole ograniczonego obszaru obliczone przy użyciu metody trapezów z wynikiem obliczonym za pomocą metody prostokątów, który został przedstawiony w sekcji „Przeczytaj”.

Napisz program, który wyznaczy przybliżoną wartość pola ograniczonego wykresem funkcji $f(x)$ i osią OX w żadanym przedziale za pomocą metody trapezów. Wynik zaokrąglij do dwóch miejsc po przecinku.

Program przetestuj dla funkcji $f(x) = x$, przedziału $\langle 0;\ 2 \rangle$ oraz liczby podprzedziałów równej 2.

Specyfikacja problemu:

Dane:

- `xPoczatek` – liczba rzeczywista; początek przedziału
- `xKoniec` – liczba rzeczywista; koniec przedziału
- `liczbaTrapezow` – liczba naturalna dodatnia; liczba podprzedziałów
- `f(x)` – funkcja rzeczywista zmiennej rzeczywistej

Wynik:

Program wypisuje pole ograniczone wykresem funkcji `f(x)` wyznaczone metodą trapezów w zadanym przedziale z dokładnością do dwóch miejsc po przecinku.

1

Naszym zadaniem jest wyznaczenie pola obszaru ograniczonego wykresem funkcji $f(x)$ oraz osią x, przybliżonego przez trapezy prostokątne. Zacznijmy więc od dołączenia biblioteki standardowej, deklaracji przestrzeni nazw i stworzenia definicji funkcji.

```
#include <iostream>
using namespace std;

double f(double x) {
    return x;
}
```

2

Następnie dodajemy główną funkcję programu, w której definiujemy potrzebne zmienne, niektóre z nich inicjujemy.

```
#include <iostream>
using namespace std;

double f(double x) {
    return x;
}

int main() {
    double xPoczatek = 0,
xKoniec = 2, wysokosc,
wynik = 0;
    int liczbaTrapezow =
2;

    return 0;
}
```

Obliczanie pola ograniczonego wykresem funkcji $f(x)$ ma zostać przetestowane w przedziale $\langle 0;\ 2 \rangle$, zatem lewy koniec rozważanego przedziału `xPoczatek` został zainicjowany wartością 0, natomiast prawy koniec przedziału `xKoniec` wartością 2. Wysokość każdego z trapezów będzie taka sama i zostanie zapisana w zmiennej `wysokosc`. Zmienna `wynik` będzie odpowiedzialna za przechowywanie wartości oszacowanego pola pod wykresem.

Ostatnią zmienną jest `liczbaTrapezow`, która jest zainicjowana wartością 2. Jednak w praktyce w celu zwiększenia dokładności przybliżenia używa się większej liczby małych fragmentów, na które dzieli się przedział.

4

Mamy już wszystkie dane potrzebne do obliczenia długości wysokości trapezów:

```
#include <iostream>
using namespace std;

double f(double x) {
    return x;
}

int main() {
    double xPoczatek = 0, xKoniec = 2, wysokosc, wynik = 0;
    int liczbaTrapezow = 2;
    wysokosc = (xKoniec - xPoczatek) / liczbaTrapezow;

    return 0;
}
```

5

Mając funkcję oraz wszystkie potrzebne zmienne, możemy przejść do szacowania wartości pola. Aby to zrobić, obliczymy pole każdego z trapezów. Użyjemy więc pętli `for`, aby przejść przez wszystkie trapezy.

```
#include <iostream>
using namespace std;

double f(double x) {
    return x;
}

int main() {
    double xPoczatek = 0, xKoniec = 2, wysokosc, wynik = 0;
    int liczbaTrapezow = 2;
    wysokosc = (xPoczatek + xKoniec) / liczbaTrapezow;

    for (int i = 0; i < liczbaTrapezow; i++) {

    }

    return 0;
}
```

6

Znamy już wartość wysokości każdego z trapezów. Jest ona taka sama dla każdego z nich. Policzmy więc długość lewej podstawy każdego z trapezów.

```
#include <iostream>
```

```
using namespace std;

double f(double x) {
    return x;
}

int main() {
    double xPoczatek = 0,
xKoniec = 2, wysokosc,
wynik = 0;
    int liczbaTrapezow =
2;
    wysokosc = (xPoczatek
+ xKoniec) /
liczbaTrapezow;

    for (int i = 0; i <
liczbaTrapezow; i++) {
        double
lewaPodstawa = f(xPoczatek
+ i * wysokosc);
    }

    return 0;
}
```

Następnym krokiem jest obliczenie długości prawej podstawy trapezu.

7

```
#include <iostream>
using namespace std;

double f(double x) {
    return x;
}

int main() {
    double xPoczatek = 0,
xKoniec = 2, wysokosc,
wynik = 0;
```

```
    int liczbaTrapezow = 2;
    wysokosc = (xPoczatek + xKoniec) / liczbaTrapezow;

    for (int i = 0; i < liczbaTrapezow; i++) {
        double lewaPodstawa = f(xPoczatek + i * wysokosc);
        double prawaPodstawa = f(xPoczatek + (i + 1) * wysokosc);
    }

    return 0;
}
```

Zauważ, że prawa podstawa `i`-tego trapezu ma taką samą długość, jak lewa podstawa kolejnego trapezu.

8

Na koniec, używając wzoru na pole trapezu, dodajemy pole bieżącej figury do zmiennej `wynik`.

```
#include <iostream>
#include <cmath>
using namespace std;

double f(double x) {
    return x;
}

int main() {
    double xPoczatek = 0, xKoniec = 2, wysokosc, wynik = 0;
```

```
11      int liczbaTrapezow =
    2;
12      wysokosc = (xPoczatek
    + xKoniec) /
    liczbaTrapezow;
13
14      for (int i = 0; i <
    liczbaTrapezow; i++) {
15          double
    lewaPodstawa = f(xPoczatek
    + i * wysokosc);
16          double
    prawaPodstawa =
    f(xPoczatek + (i + 1) *
    wysokosc);
17
18          wynik +=
    ((lewaPodstawa +
    prawaPodstawa) * wysokosc)
    / 2;
19      }
20      wynik = round(wynik *
    100) / 100;
21      cout << wynik << endl;
22
23      return 0;
24  }
```

Po przejściu przez wszystkie trapezy zaokrąglamy do dwóch miejsc po przecinku, a następnie wypisujemy `wynik`. Jako że chcemy wykorzystać funkcję `round()` z biblioteki `<cmath>`, to musimy dołączyć odpowiedni plik nagłówkowy

Możemy zminimalizować liczbę mnożeń poprzez wyłączenie wspólnego czynnika przed nawias. Dzięki temu w pętli `for` do zmiennej `wynik` dodajemy podstawy trapezów, a na końcu wykonujemy tylko jedną operację mnożenia tej sumy przez połowę wysokości.

```cpp
#include <iostream>
#include <cmath>
using namespace std;

double f(double x) {
    return x;
}

int main() {
    double xPoczatek = 0, xKoniec = 2, wysokosc, wynik = 0;
    int liczbaTrapezow = 2;
    wysokosc = (xPoczatek + xKoniec) / liczbaTrapezow;

    for (int i = 0; i < liczbaTrapezow; i++) {
        double lewaPodstawa = f(xPoczatek + i * wysokosc);
        double prawaPodstawa = f(xPoczatek + (i + 1) * wysokosc);

        wynik += (lewaPodstawa + prawaPodstawa);
    }
    wynik *= (wysokosc / 2);
    wynik = round(wynik * 100) / 100;
    cout << wynik << endl;

    return 0;
}
```

Polecenie 2

Przeanalizuj symulację interaktywną wizualizującą metody obliczanie pola ograniczonego wykresem funkcji $f(x)$ i osią OX. Zwróć uwagę na różnicę w interpretacji geometrycznej metody trapezów oraz metody prostokątów z punktem środkowym. Zastanów się, dlaczego wymienione metody różnią się wynikami.

> **Ważne!**
>
> W razie problemów z wyświetlaniem GeoGebry, uruchom ją na pełnym ekranie.

Więcej informacji na temat zaprezentowanych metod znajdziesz w e-materiale Obliczanie przybliżonej wielkości pola obszarów zamkniętych.

Sprawdź się

Pokaż ćwiczenia: 

Ćwiczenie 1

Napisz program, który wyznaczy przybliżoną wartość pola ograniczonego wykresem funkcji $f(x)$ i osią OX oraz odcinkiem: `[xPoczatek, xKoniec]` za pomocą metody prostokątów. Użyj wariantu średnich prostokątów. Wynik zaokrąglij do trzech miejsc po przecinku.

Przetestuj działanie programu dla funkcji $f(x) = -\mathrm{x}^2 + 16$ i przedziału $\langle -4, 4 \rangle$. Rozwiązanie powinno bazować na wykorzystaniu 1000 prostokątów.

Specyfikacja problemu:

Dane:

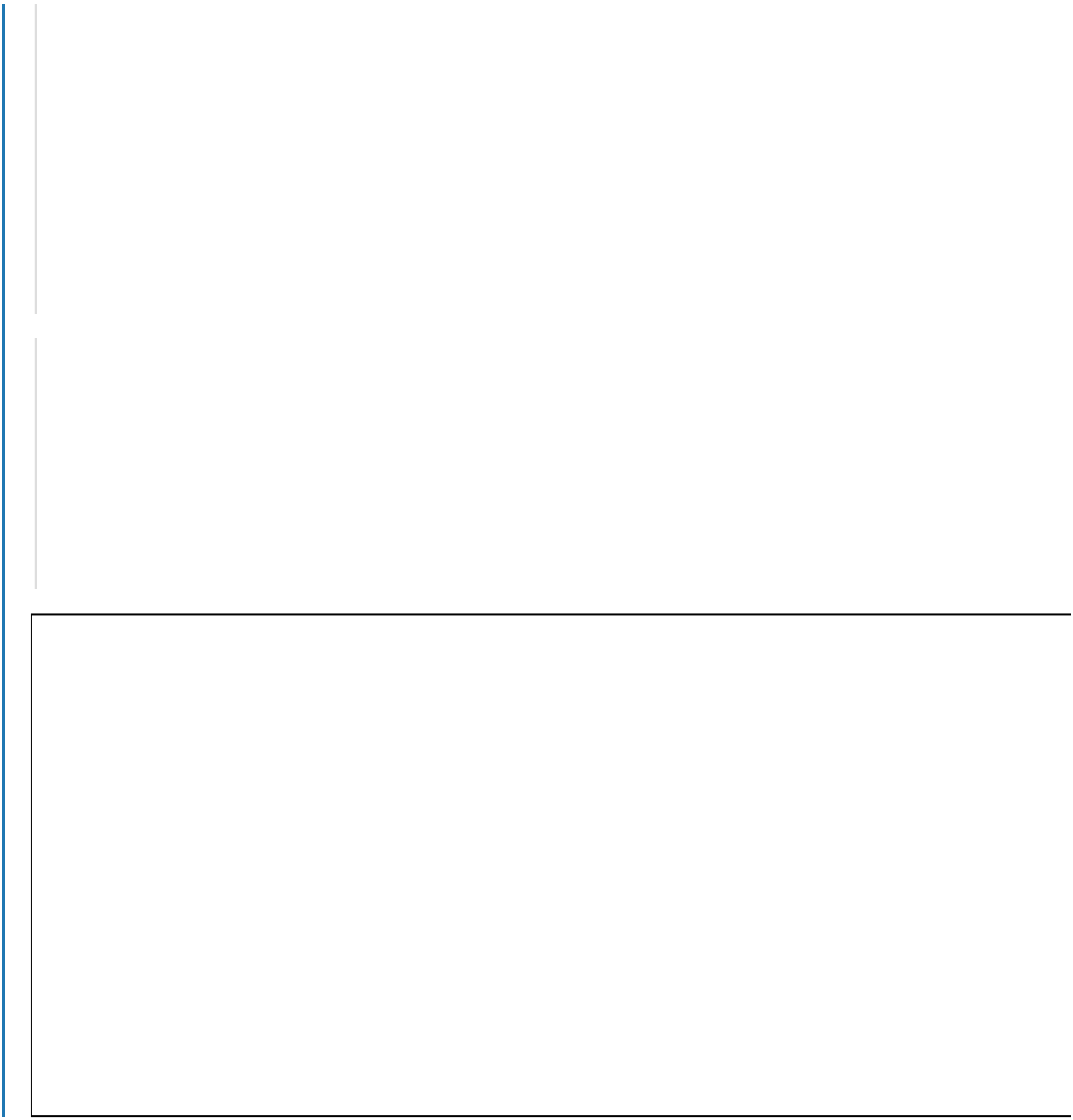
- `xPoczatek` – liczba rzeczywista; początek przedziału
- `xKoniec` – liczba rzeczywista; koniec przedziału
- `liczbaProstokatow` – liczba naturalna dodatnia; liczba podprzedziałów
- `f(x)` – funkcja rzeczywista zmiennej rzeczywistej

Wynik:

Program wypisuje obliczone pole obszaru ograniczonego wykresem funkcji `f(x)` i osią OX w zadanym przedziale, podany wynik powinien być zaokrąglony z dokładnością do trzech miejsc po przecinku.

Twoje zadania

1. Program wypisuje przybliżoną wartość obszaru ograniczonego wykresem funkcji $f(x)$ w zadanym przedziale, zaokrąglony z dokładnością do trzech miejsc po przecinku.



Ćwiczenie 2

Napisz program, który wyznaczy metodą prostokątów przybliżoną wartość pola obszaru ograniczonego osią funkcji $f(x)$ i osią OX oraz odcinkiem: `[xPoczatek, xKoniec]`.

Przetestuj jego działanie dla funkcji $f(x) = x^3 - 5x^2 + 2$ i przedziału $\langle -2, 2 \rangle$. Zauważ, że w podanym przedziale funkcja może przyjmować wartości ujemne.

Rozwiązanie powinno bazować na wykorzystaniu 10000 prostokątów. Zastosuj wariant prawych prostokątów. Wynik zaokrąglij do dwóch miejsc po przecinku.

Specyfikacja problemu:

Dane:

- `xPoczatek` – liczba rzeczywista; początek przedziału
- `xKoniec` – liczba rzeczywista; koniec przedziału
- `liczbaProstokatow` – liczba naturalna dodatnia; liczba podprzedziałów
- `f(x)` – funkcja rzeczywista zmiennej rzeczywistej

Wynik:

Program wypisuje wyznaczoną za pomocą metody prostokątów przybliżoną wartość pola obszaru ograniczonego funkcją `f(x)` i osią OX w zadanym przedziale, zaokrągloną z dokładnością do dwóch miejsc po przecinku.

Twoje zadania

1. Program wyznacza za pomocą metody prostokątów przybliżoną wartość pola wykresu ograniczonego wykresem funkcji i osią OX, a następnie wypisuje go na standardowe wyjście. Wynik powinien być zaokrąglony do dwóch miejsc po przecinku.

Ćwiczenie 3

Napisz program, który oszacuje metodą trapezów pole obszaru ograniczonego osią funkcji $f(x)$ i osią OX oraz odcinkiem: `[xPoczatek, xKoniec]`.

Przetestuj jego działanie dla funkcji $f(x) = \cos(x - 3)$ i przedziału $\langle 1, 6 \rangle$. Zauważ, że w podanym przedziale funkcja może przyjmować wartości ujemne.

Rozwiązanie powinno bazować na wykorzystaniu 1000 trapezów. Wynik zaokrąglij do dwóch miejsc po przecinku.

Specyfikacja problemu:

Dane:

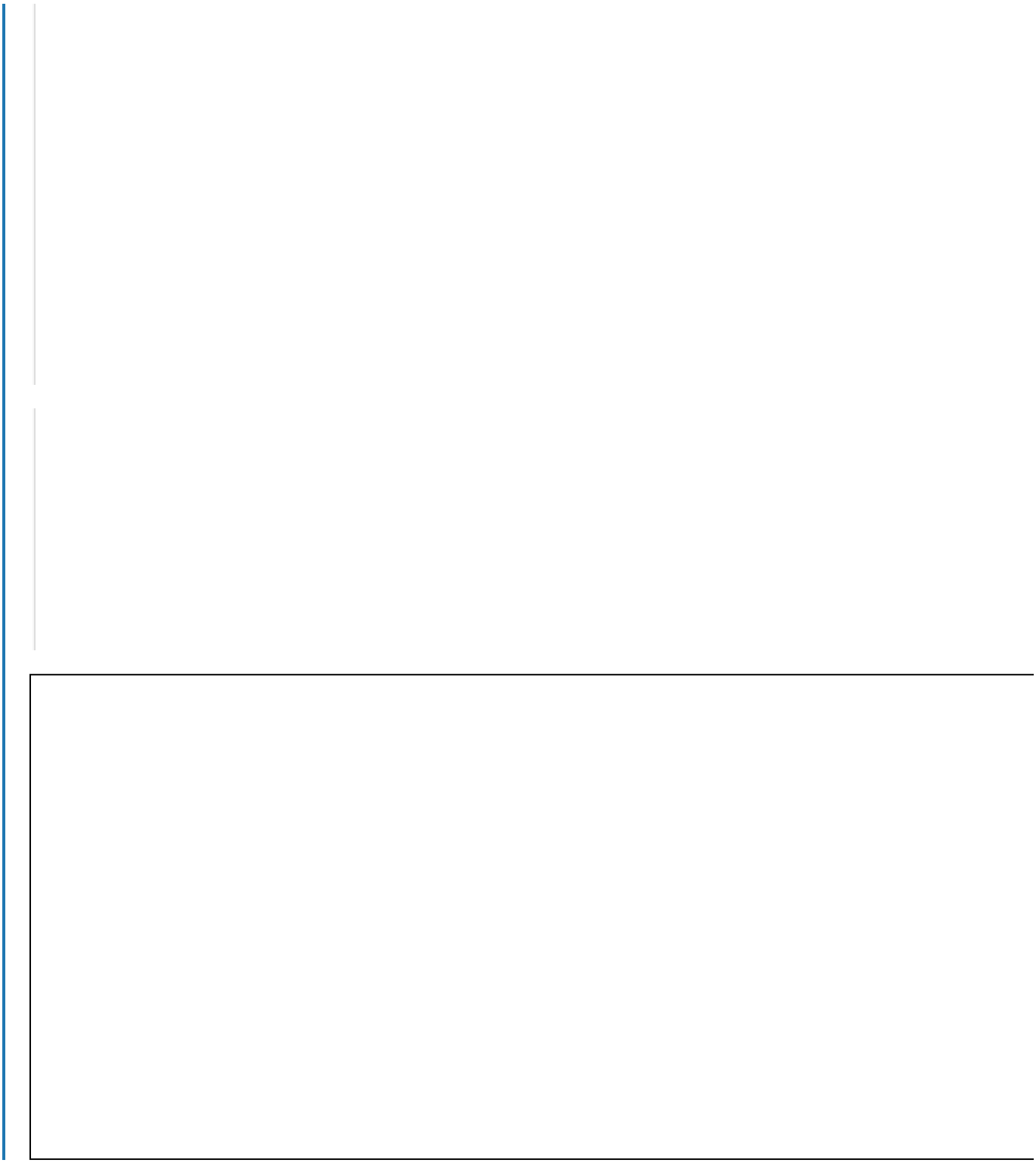
- `xPoczatek` – liczba rzeczywista; początek przedziału
- `xKoniec` – liczba rzeczywista; koniec przedziału
- `liczbaTrapezow` – liczba naturalna dodatnia; liczba podprzedziałów
- `f(x)` – funkcja rzeczywista zmiennej rzeczywistej

Wynik:

Program wypisuje oszacowane metodą trapezów pole obszaru ograniczonego funkcją `f(x)` i osią OX w zadanym przedziale, zaokrąglone z dokładnością do dwóch miejsc po przecinku.

Twoje zadania

1. Program szacuje metodą trapezów pole wykresu ograniczonego wykresem danej funkcji i osią OX i wypisuje go na standardowe wyjście. Wypisany wynik powinien być zaokrąglony z dokładnością do dwóch miejsc po przecinku.



Dla nauczyciela

Autor: Maurycy Gast
Przedmiot: Informatyka

Temat: Obliczanie przybliżonej wielkości pola obszarów zamkniętych w języku C++

Grupa docelowa:

Liceum ogólnokształcące i technikum, liceum ogólnokształcące, technikum, zakres 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.

Treści nauczania – wymagania szczegółowe

I. Rozumienie, analizowanie i rozwiązywanie problemów.

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;

4) ilustruje i wyjaśnia rolę pojęć, obiektów i operacji matematycznych w projektowaniu rozwiązań problemów informatycznych i z innych dziedzin, posługuje się pojęciem logarytmu;

II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych.

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

3) sprawnie posługuje się zintegrowanym środowiskiem programistycznym przy pisaniu, uruchamianiu i testowaniu programów;

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

2) wykorzystuje znane sobie algorytmy przy rozwiązywaniu i programowaniu rozwiązań następujących problemów:

f) obliczanie przybliżonej wielkości pola obszarów zamkniętych;

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):

- Użyjesz w praktyce wiedzy na temat metod prostokątów i trapezów.
- Przeanalizujesz algorytmy obliczania pola powierzchni pod wykresem metodami prostokątów i trapezów zapisane w języku C++.
- Zaimplementujesz poznane algorytmy służące do szacowania wartości pola obszaru zamkniętego w języku C++.
- Obliczysz pola powierzchni przykładowych figur.

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: „Obliczanie przybliżonej wielkości pola obszarów zamkniętych w języku C++". Uczniowie zapoznają się z treściami w sekcji „Przeczytaj".

Faza wstępna:

1. Nauczyciel wprowadza uczniów szczegółowo w temat lekcji i jej cele. Może posłużyć się wyświetloną na tablicy zawartością sekcji „Wprowadzenie".
2. Prowadzący prosi uczniów, aby zgłaszali swoje propozycje pytań do tematu. Jedna osoba może zapisywać je na tablicy. Gdy uczniowie wyczerpią swoje pomysły, a pozostały jakieś ważne kwestie do poruszenia, nauczyciel je dopowiada.

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 „Symulacja interaktywna". Uczniowie wspólnie analizują prezentację, a następnie próbują obliczyć pola pod trzema innymi, dowolnymi funkcjami.
3. **Ćwiczenie umiejętności.** Uczniowie wykonują indywidualnie ćwiczenie nr 1 z sekcji „Sprawdź się", a następnie porównują swoje odpowiedzi z kolegą lub koleżanką.
4. Liga zadaniowa – uczniowie pracując w parach, wykonują ćwiczenie nr 2 z sekcji „Sprawdź się", a następnie dzielą się swoimi wynikami przez porównywanie napisanego kodu z inną grupą, która również zakończyła zadanie.
5. Uczniowie, nadal pracując w parach, rozwiązują ćwiczenie 3 z sekcji „Sprawdź się".

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. Nauczyciel prosi uczniów o podsumowanie zgromadzonej wiedzy w zakresie programowania w języku C++.

Praca domowa:

1. Uczniowie wykonują polecenie nr 2 z sekcji „Symulacja interaktywna”.

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:

- Uczniowie mogą wykorzystać multimedium w sekcji „Symulacja interaktywna” do przygotowania się do lekcji powtórkowej.