Zestaw zadań PPR2018

Uwaga: Zadania należy rozwiązywać samodzielnie. Kopiowanie rozwiązań z Internetu lub od kolegów skutkuje uzyskaniem wyniku 0 pkt.

Sposób przekazywania rozwiązań zadań do oceny zostanie podany wkrótce.

Zadanie dodatkowe do pierwszego sprawdzianu

Napisz program do obliczania całki oznaczonej z funkcji wielomianowej trzeciego stopnia dwoma metodami – trapezów oraz Monte-Carlo. Użytkownik podaje współczynniki wielomianu oraz zakres całkowania. Dla metody trapezów podaje także krok całkowania a dla metody Monte-Carlo liczbę losowań.

Wymagania

1) Sprawdzanie poprawności wprowadzonych danych,
2) Utrzymanie porządku w kodzie źródłowym – wcięcia, komentarze.

Zasady oceny

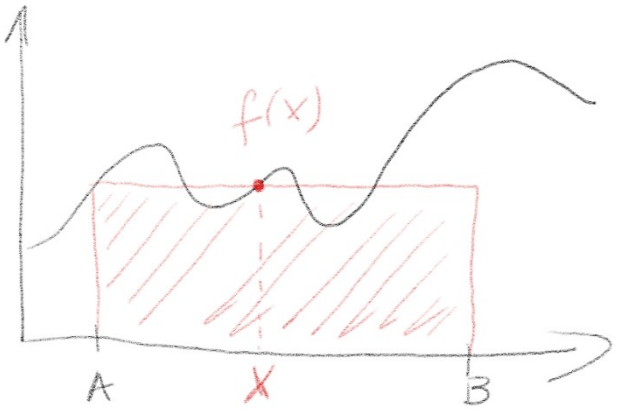
Za zadanie można uzyskać maksymalnie 20 punktów, które dodają się do wyniku pierwszego sprawdzianu, jednakże nie można w sumie otrzymać więcej niż 20 punktów.

Termin oddania

20 stycznia 2019 roku godz. 23:59
Zadania oddane po tym terminie nie zostaną ocenione.

Opis metody Monte-Carlo wyznaczenia całki.

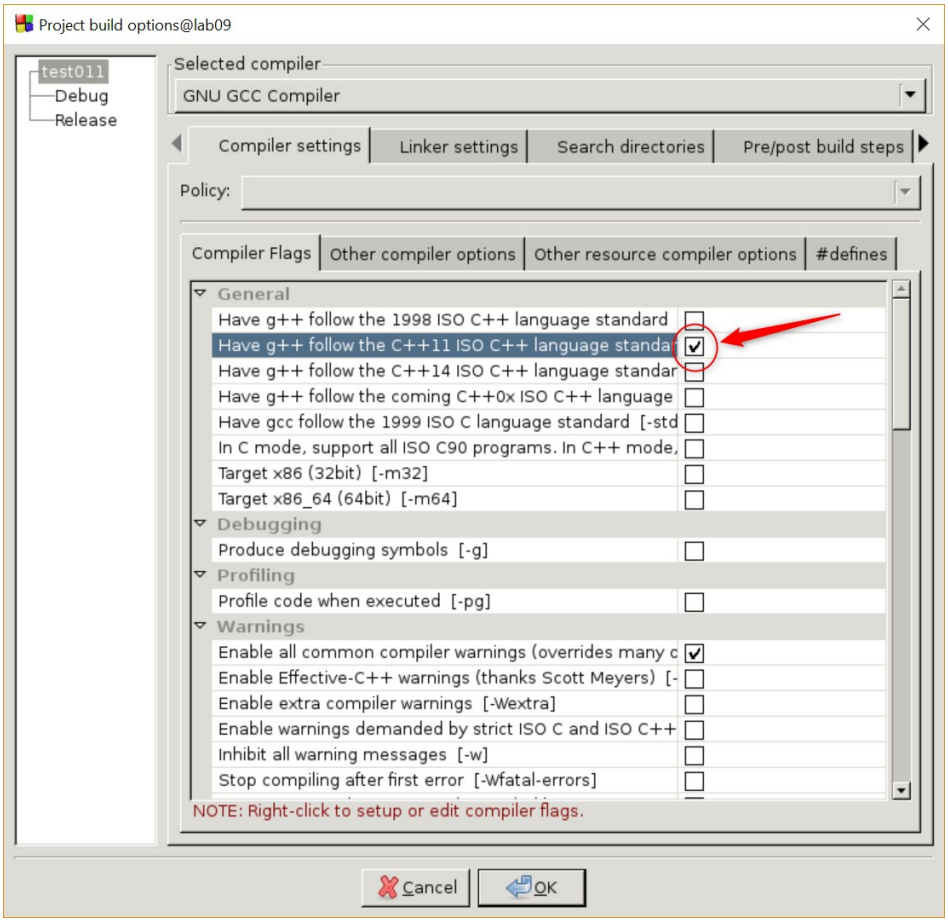
Dla zadanego przedziału całkowania [A,B] oraz funkcji f(x) losujemy liczbę X zawierającą się w tym przedziale i wyznaczamy pole powierzchni prostokąta o bokach B-A oraz f(X) - zachowując znak zależny od wartości f(X). Powtarzamy tą operację wielokrotnie uśredniając wyznaczone pola powierzchni – uzyskujemy przybliżenie całki.



Liczby losowe

W bibliotece standardowej języka C istnieje funkcja rand(), która jest bardzo słabym generatorem liczb losowych. Biblioteka standardowa C++ w wersji standardu C++11 zawiera lepsze metody generacji liczb losowych. Przykład poniżej wykorzystuje generator Mersene Twister do generowania liczb losowych z przedziału <0,1> z rozkładem równomiernym. Aby odblokować użycie standardu C++11 w przypadku CodeBlocks w menu Project->Build Options należy zaznaczyć pole jak na rysunku poniżej, co powoduje dodanie w linii poleceń kompilatora G++ dodatkowej opcji kompilacji -std=c++11

```cpp
#include <iostream>
#include <random>

using namespace std;

// tworzenie generatora liczb losowych oraz
// rozkładu równomiernego w przedziale <0,1>
std::random_device rd;
std::mt19937 engine(rd());
std::uniform_real_distribution<double> dist{ 0.0, 1.0 };

// funkcja zwraca liczbę losową z rozkładu równomiernego <0,1>
double losowa()
{
	return dist(engine);
}

int main()
{
	// pobierz kolejna liczbe losowa
	double x = losowa();

	cout << "X=" << x << endl;
	return 0;
}
```



Rysunek 1. Włączanie wsparcia standardu C++ w CodeBlocks

Zadania domowe

Zasady zaliczenia zadań domowych
Każde z dwóch zadań domowych oceniane jest na 10 punktów. Ocenie podlega zarówno realizacja zadania jak i jego dokumentacja w postaci schematu blokowego wskazanego w treści zadania algorytmu (8 pkt za realizację, 2 za opis algorytmu).

Termin oddania
20 stycznia 2019 roku godz. 23:59
Zadania oddane po tym terminie nie zostaną ocenione.

Zadanie domowe 1

Macierze oraz użycie strumieni C++ do pobierania danych z pliku

W pliku dane.txt umieszczono zbiór danych w następującym formacie:

a) pierwsza liczba całkowita informuje ile macierzy znajduje się w pliku danych,
b) dalej znajdują się macierze o wymiarach MxN opisane w następujący sposób:
    a. dwie pierwsze liczby całkowite zawierają ilość wierszy i kolumn macierzy,
    b. następnie znajdują się wartości kolejnych pól macierzy podawane wierszami.

Przykładowy plik danych znajduje się na końcu zadania.

Napisz program, który wczytuje macierze z pliku i wykonuje operację mnożenia kolejnych macierzy (A×B×C×…×Y) sprawdzając jednocześnie czy można pomnożyć kolejną macierz przez wynik poprzedniego mnożenia. Ostateczny wynik obliczeń wypisz na ekran.

Przed wykonaniem zadania, stwórz schemat blokowy algorytmu liczącego iloczyn oraz wczytującego dane z pliku – schematy blokowe dołącz do rozwiązania zadania.

Przykładowy plik dane.txt

```
3
2 3
1.0 2.0 3.0 4.0 5.0 6.0
3 2
2.0 4.0 5.0 6.0 7.0 8.0
2 2
1.0 2.0 3.0 4.0
```

Zadanie domowe 2

Gra Gomoku

Zrealizuj program wspierający graczy w grze Gomoku (zwanej także Five in a Row) - https://pl.wikipedia.org/wiki/Gomoku

Program powinien:

1) Rysować aktualny stan planszy gry,
2) Przyjmować wskazanie pozycji kolejnego ruchu naprzemiennie dla gracza białego i czarnego,
3) Sprawdzać wygraną – czy w danej chwili gracz uzyskał 5 pionków w linii.

Przed wykonaniem zadania, stwórz schemat blokowy algorytmu sprawdzania wygranej – schemat blokowy dołącz do rozwiązania zadania.