Programowanie i metody numeryczne

Zadania – seria 5.

Język C++: wyrażenia lambda, algorytmy STL.

Zadanie 1. `sampling` – Funkcje jako argumenty funkcji.

Napisz szablon funkcji

```
std::vector<double> sampling(double a, double b, std::size_t n, const auto& f)
```

zwracającej wektor wartości funkcji `f` obliczonych w `n` równoodległych punktach z przedziału $[a, b]$.

Korzystając z tego szablonu napisz program `sampling`, którego celem jest przygotowanie danych do narysowania wykresu funkcji

$$f(x) = \sin x^3,$$

określonej na przedziale zadanym przez użytkownika. Program powinien przyjmować jako argumenty wywołania trzy liczby, dwie rzeczywiste oraz jedną naturalną, odpowiadające kolejno wartościom `a`, `b` i `n`. Po wywołaniu program ma tworzyć w folderze roboczym plik o nazwie `sampling.txt`, którego każda linia zawiera dwie oddzielone spacją liczby: pierwsza jest argumentem, zaś druga – wartością funkcji $f(x)$ obliczoną dla tego argumentu. Zbiór argumentów powinien być złożony z `n` równoodległych punktach z przedziału $[a, b]$. Funkcja $f(x)$ powinna zostać zaimplementowana jako wyrażenie lambda.

Wykorzystując zewnętrzne narzędzie (np. skrypt napisany w języku Python) narysuj wykres funkcji $f(x)$ w zadanym przedziale na podstawie danych z pliku `sampling.txt`.

Zadanie 2. `natural` – Generator liczb naturalnych.

Napisz klasę `NaturalNumbers` implementującą generator kolejnych liczb naturalnych.

Zaimplementuj:

- publiczny alias `result_type` reprezentujący typ `std::size_t`,
- konstruktor jednoargumentowy, którego argument określa zakres generatora – generowane mają być liczby od 1 do liczby zadanej argumentem konstruktora,
- konstruktor dwuargumentowy, którego argumenty określają zakres generatora – generowane mają być liczby z przedziału zadanego argumentami konstruktora,
- iterator dwukierunkowy; powinien on zostać zdefiniowany jako klasa `NaturalNumbers::iterator`,
- publiczne metody `begin()` i `end()` zwracające iterator wskazujący na, odpowiednio, pierwszą liczbę z zakresu oraz liczbę o jeden większą od ostatniej liczby z zakresu.

Korzystając z tej klasy napisz program `natural`, który przyjmuje jako argumenty wywołania dwie liczby naturalne i wypisuje na standardowe wyjście liczby naturalne z przedziału określonego argumentami wywołania.

Zadanie 3. `normal` – Próbkowanie dużej ilości danych z rozkładu normalnego.

Napisz program `normal`, który wylosuje 100 000 liczb całkowitych z rozkładu normalnego o wartości średniej 0 i odchyleniu standardowym 3, posługując się generatorem liczb pseudolosowych *Mersenne Twister 19937*, a następnie wybierze losowo spośród nich 100 liczb. Program ma następnie stworzyć plik tekstowy `normal.txt` i zapisać w nim dane niezbędne do wykonania histogramu wybranych 100 liczb. Każda linia pliku powinna zawierać dwie oddzielone spacją liczby: pierwsza jest liczbą całkowitą, zaś druga – liczbą jej wystąpień.

Wykorzystując zewnętrzne narzędzie (np. skrypt napisany w języku Python) narysuj histogram danych z pliku `normal.txt`.

Zadanie 4. `quicksort` – Sortowanie szybkie.

Napisz szablon funkcji

```
template <typename RandomIt>
void QuickSort(RandomIt begin, RandomIt end)
```

wykonującej sortowanie metodą sortowania szybkiego. Funkcja powinna sortować dane zawarte pomiędzy dwoma iteratorami dostępu bezpośredniego `begin` i `end`.

Możesz wykorzystać algorytm biblioteki standardowej `std::partition` , który dzieli elementy leżące pomiędzy dwoma zadanymi iteratorami na dwie grupy: w pierwszej znajdują się elementy spełniające określony warunek, zaś w drugiej – pozostałe.

Korzystając z tego szablonu, napisz program `quicksort`, który wczytuje ze standardowego wejścia ciąg liczb całkowitych, a następnie wypisuje te liczby na standardowe wyjście w kolejności rosnącej.

Zadanie 5. `normalize` – Normalizacja danych.

Napisz szablon funkcji

```
template <typename ForwardIt, typename OutIt, typename T>
void Normalize(ForwardIt begin, ForwardIt end, OutIt out, T min, T max)
```

wykonującej normalizację danych. Funkcja powinna poddawać dane zawarte pomiędzy dwoma iteratorami jednokierunkowumi `begin` i `end` transformacji liniowej, w wyniku której największa spośród ich wartości będzie równa wartości zmiennej `max`, zaś najmniejsza spośród ich wartości – wartości zmiennej `min`. Otrzymane w ten sposób znormalizowane dane powinny być przekazywane do iteratora danych wyjściowych `out`.

Możesz wykorzystać algorytmy biblioteki standardowej:

- `std::minmax_element`, zwracającą parę iteratorów wskazujących na najmniejszy oraz największy spośród elementów leżących pomiędzy dwoma zadanymi iteratorami,
- `std::transform`, wykonującą zadaną transformację na wszystkich elementach leżących pomiędzy dwoma zadanymi iteratorami.

Korzystając z tego szablonu, napisz program `normalize`, który wczytuje ze standardowego wejścia ciąg liczb rzeczywistych oraz żądne wartości maksymalną i minimalną, a następnie wypisuje na standardowe wyjście znormalizowane liczby.

Opracowanie: Bartłomiej Zglinicki.