Programowanie II R

Zadania – seria 12.

STL: iteratory.

Zadanie 1. `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 2. `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 następujące 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.

Zadanie 3. `natural` – Generator liczb naturalnych.

Napisz klasę `NaturalNumbers` implementującą generator kolejnych liczb naturalnych.

Zaimplementuj:

- 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.

Opracowanie: Bartłomiej Zglinicki.