Programowanie i metody numeryczne

Zadania – seria 4.

Język C++: STL – kontenery i iteratory.

Zadanie 1. `lotto` – Losowanie liczb.

Napisz program `lotto`, który losuje sześć różnych liczb całkowitych z przedziału od 1 do 49 włącznie, a następnie wypisuje te liczby na standardowe wyjście w kolejności rosnącej. Wykorzystaj `std::set<T>` oraz generator liczb losowych *Mersenne Twister 19937*, zaimplementowany jako typ `std::mt19937` w pliku nagłówkowym `random`.

Przykładowe wykonanie
Wywołanie:
Windows: `lotto.exe`
macOS / Linux: `./lotto`
Wyjście:
```
1 7 14 16 36 46
```

Zadanie 2. `words` – Liczenie wyrazów.

Napisz program `words`, który wczytuje ze standardowego wejścia wyrazy aż do napotkania znaku końca pliku, a następnie wypisuje te wyrazy na standardowe wyjście w kolejności alfabetycznej, każdy tylko raz, podając przy każdym z nich ilość jego wystąpień. Wykorzystaj `std::map<Key, T>`.

Przykładowe wykonanie
Wywołanie:
Windows: `words.exe`
macOS / Linux: `./words`
Wejście
```
pies kot żółw kot kot pies żółw [EOF]
```
Wyjście
```
kot 3
pies 2
żółw 2
```

Zadanie 3. `mergesort` – Sortowanie przez scalanie.

Sortowanie przez scalanie (ang. *merge sort*) to jeden z algorytmów sortowania kolekcji danych, oparty na metodzie *divide et impera* (łac. *dziel i rządź*).

Pierwszym krokiem tego algorytmu jest podzielenie sortowanej kolekcji danych na dwie równe części, a następnie rekurencyjne zastosowanie algorytmu do każdej z nich. Rekurencja kończy się, gdy ilość elementów w każdej z części jest równa 1 – wówczas obie części można uznać za posortowane. Posortowane części należy wówczas połączyć w taki sposób, by zachować uporządkowanie elementów.

Napisz program `mergesort`, który wczytuje ze standardowego wejścia liczby całkowite aż do napotkania znaku końca pliku, a następnie wypisuje te liczby na standardowe wyjście w kolejności rosnącej. Program powinien sortować liczby posługując się algorytmem sortowania przez scalanie. Wykorzystaj `std::list<T>`.

Zadanie 4. `brackets` – Balansowanie ciągów nawiasów.

Ciąg znaków złożony z nawiasów `(`, `)`, `[`, `]` i {, } uważamy za *zbalansowany*, jeśli każdemu z nawiasów otwierających odpowiada właściwy nawias zamykający oraz jeśli nawiasy są poprawnie zagnieżdżone. Na przykład ciągi: `()[{}]`, `(){[]{([])}}` są zbalansowane, zaś ciągi `)(`, `[{]}`, `({[})` – nie.

Napisz program `brackets`, który sprawdza, czy ciąg nawiasów przekazany mu jako pierwszy argument wywołania jest zbalansowany i wypisuje na standardowe wyjście stosowną informację. Wykorzystaj `std::stack<T>`.

Zadanie 5. `roman` – Liczby rzymskie.

Napisz program `roman`, który:

- jeśli pierwszym argumentem wywołania jest `r`, zamienia przekazaną mu jako drugi argument liczbę całkowitą zapisaną za pomocą cyfr arabskich na liczbę zapisaną za pomocą cyfr rzymskich i wypisuje ją,
- jeśli pierwszym argumentem wywołania jest `a`, zamienia przekazaną mu jako drugi argument liczbę całkowitą zapisaną za pomocą cyfr rzymskich na liczbę zapisaną za pomocą cyfr arabskich i wypisuje ją.

Wykorzystaj `std::map<Key, T>`.

Opracowanie: Bartłomiej Zglinicki.