Języki Programowania, 13.12.2018
Zadanie 8

Dzisiejsze zadanie ma charakter projektowy. Przychodzi do Was promotor (albo szef w pracy :)) i prosi o zbadanie za pomocą detektora promieniowania pewne nieznane źródło promieniotwórcze – należy zmierzyć jego czas połowicznego rozpadu. Nasz detektor, jaki mamy na wyposażeniu laboratorium, to jednak prosty licznik, który po prostu zlicza ilość rozpadów w danej jednostce czasu (czyli na „wyjściu" mamy tylko jedną liczbę). Wiadomo, że aby określić jaki jest czas połowicznego rozpadu tego źródła, trzeba zmierzyć rozkład promieniowania i „dopasować" do niego parametry rozkładu Poissona, który opisuje prawo rozpadu promieniotwórczego (ilość jąder rozpadających się w czasie). Zatem aby zmierzyć źródło promieniotwórcze potrzebujemy stworzyć histogram z dużej próbki danych z detektora.

W naszym zadaniu promotor/szef prosi każdego z Was o stworzenie takiego narzędzia do pracy z detektorem – histogramu, który wyglądałby jak przykład przedstawiony pod spodem, oraz jego przetestowanie. Narzędzie trzeba napisać w sposób uniwersalny, to znaczy możemy zadać dowolną liczbę binów histogramu jak i jego dowolny przedział (dolną granicę pierwszego binu i górną granicę ostatniego binu). Histogram powinien również przechowywać informacje o ilości wejść dla wartości które były mniejsze (underflow) i większe (overflow) od przedziału histogramu.

```
Printing histogram HistTest
underflow: 2
overflow: 5
0: *
1: *****
2: *****
3: *******
4: ********
5: ******
6: *********
```

gdzie liczba `*` oznacza jedno wejście w binie o zadanym numerze.

Do wykonania (robimy po kolei):

Część I – klasa Histogram

1. Generujemy dużą próbkę liczb pseudolosowych z wybranych rozkładów (Gauss, Uniform czy Poisson) i zapisujemy wynik do pliku, np. **dane.Gauss.txt**
2. Tworzymy klasę `Histogram`. Następnie tworzymy plik z funkcją **main**, w którym będziemy otwierać zapisany wcześniej plik wpisywać dane do histogramu. Na końcu wypiszemy histogram na ekran. Nazwę wczytywanego pliku podajemy jako 1 parametr uruchomienia programu.
3. Na koniec przekierowujemy „output" uruchomionego programu do pliku **out.txt**.
   3 p.

Część II – użycie std::vector

Modyfikujemy nasz program tak (a najlepiej robimy kopię działającej klasy i ją modyfikujemy), by używał dynamicznej struktury danych `std::vector` (która pełni rolę dynamicznej tablicy) ze standardowej biblioteki szablonów C++ (STL – Standard Template Library). Obok listy jest to druga bardzo ważna struktura danych, którą należy znać.

Zauważmy, że użycie STL powoduje, że nie musimy „wymyślać koła od nowa", tylko używamy już gotowych i zoptymalizowanych struktur danych i algorytmów. W naszym przykładzie nie musimy się zatem martwić alokacją i zwalnianiem pamięci oraz niepotrzebne nam już będzie dodatkowe pole przechowujące ilość binów, gdyż to dostarcza nam „automatycznie" klasa `std::vector`.

2 p.

Przydatne linki:
http://www.cplusplus.com/reference/vector/vector/
http://pl.wikibooks.org/wiki/C++/Vector