Zadania PK4 “Iteratory i Algorytmy STL”

Iteratory:

Zadanie 1.

a) Stwórz kontener typu vector wypełniając go 20 pierwszymi liczbami naturalnymi za pomocą metody **push_back()**. Następnie utwórz iterator dla niego i przypisz mu element równy 9, skorzystaj z funkcji **find()** do wykonania tej operacji. Wyświetl zawartość iteratora.
b) Skonwertuj utworzony iterator na iterator odwrócony. Po wykonaniu tego zadania użyj metody **base()** na iteratorze, aby dokonać konwersji iteratora odwróconego na zwykły. Dlaczego wartość iteratora uległa zmianie?

Zadanie 2.

a) Stwórz kontener **deque** typu double oraz dwa iteratory mu odpowiadające za wstawienie elementu na początek, a drugi na koniec.
b) Za pomocą odpowiednich iteratorów wstaw jeden element na początek i na koniec struktury. Wyświetl zawartość **deque**.
c) Wstaw następny element korzystając z metody **back_inserter()**
d) Skopiuj elementy i wstaw je na początek używając **copy()**, gdzie parametrami są: początek,koniec listy oraz **front_inserter()**.
e) Dla tego samego kontenera utwórz iterator wstawiający ogólny. Ustaw kontener dla następującej pozycji: **container.begin()+2**,następnie używając tej funkcji wstaw element.

Zadanie 3.

Posiadasz urządzenie, które nadaje częstotliwość z stacji radiowych. Nadaje ono częstotliwość z określoną zależnością Twoim zadaniem jest pierwsze znalezienie **PODWÓJNEGO** pojawienia się liczby.Częstotliwości jako liczby dodają się do siebie tak długo, aż nie nastąpi znalezienie szukanego powtórzenia.

Przykładowo: +1,-1 -> pierwsze znalezienie 0.

+3,+3,+4,-2,-4 -> pierwsze znalezienie 10

Opis przykładu +3,+3,+4,-2,-4:
3 + 3 = 6 (pierwsze wystąpienie 6)
6 + 4 = 10 (pierwsze wystąpienie 10)
10 + (-2) = 8 (pierwsze wystapienie 8)
8 + (-4) = 4 (pierwsze wystapienie 4)
Od tego momentu częstotliwości się powtarzają:
4 + 3 = 7 (pierwsze wystapienie 7)
7 + 3 = 10 (DRUGIE wystapienie 10) KONIEC opd. 10

//**ZALECANE STOSOWANIE RELEASE** , zadanie posiada testy jednostkowe!

a) Dla utworzonej klasy Frequency napisać ciała następujących metod:
    - **void SavetoVector(string namefile)** -> Otworzy plik do odczytu, a następnie używając **getline()** wpisze do wektora **<u>givenvalue</u>** zawartość pliku “input.txt”
    - **int PuzzleSolver()-**>Uzupełni funkcje sprawdzającą częstotliwość o dwa **fory** wykorzystujące iteratory oraz odpowiednią operację na iteratorach
    - **void Find()** -> Wyszuka w całym wektorze repeated, czy wynikowa liczba się tam znajduję i wpiszę na jej pozycję wartość 20. W innym przypadku określi za pomocą metody distance() odległość pomiędzy pierwszą, a ostatnią liczbą w wektorze.
      Dane znajdują się w pliku “input.txt”.

Algorytmy:

Zadanie 4.
Sprawdź działanie następujących funkcji:

replace(beg, end, old_val, new_val);
replace_if(beg, end, pred, new_val);
replace_copy(beg, end, dest, old_val, new_val);
replace_copy_if(beg, end, dest, pred, new_val);

Dla zadeklarowanych w pliku nagłówkowym "algorytmy.h" wektorów v1 (zawierający liczby) i v2 (pusty) uruchom poniższą funkcję i sprawdź zawartość obydwu wektorów.
Uzasadnij działanie funkcji.
remove_copy_if(v1.begin(), v1.end(), back_inserter(v2), [](int i){return i%2;});

Zadanie 5.
Zaimplementuj fragment kodu ze slajdów "Algorytmy zmieniające kolejność elementów" i "Przekazanie funkcji do algorytmu" w taki sposób, aby operował na elementach listy.

Zadanie 6.
Napisz program ładujący liczby z pliku “zad6.txt” i wypisujący liczby parzyste do jednego pliku, liczby nieparzyste do drugiego i liczby posortowane(bez duplikatów) do trzeciego. Na wyjście standardowe należy wypisać sumę liczb z pliku.
W zadaniu użyj iteratorów strumienia we/wy i algorytmów for_each, accumulate.
Nie wolno wykorzystywać operatorów << i >> do zapisu/odczytu.