Міністерство освіти і науки України

Національний лісотехнічний університет України

Кафедра інженерії програмного забезпечення

**ЗВІТ ДО ЛАБОРАТОРНОЇ РОБОТИ №8**

з навчальної дисципліни

«Об’єктно – орієнтоване програмування»

на тему: «Стандартна бібліотека шаблонів STL, алгоритми»

Виконав:

студент групи ІПЗС-11

Возьний С.П.

Перевірила:

Головата С. Б.

Львів – 2023

**Лабораторна робота №8**

**Тема:** Стандартна бібліотека шаблонів STL, алгоритми

**Мета:** використовувати алгоритми STL для роботи з шаблонними об’єктами цієї ж бібліотеки.

**Хід виконання**

**Завдання 1.** В одновимірному масиві, що складається з n елементів, обчислити: суму додатніх елементів масиву, добуток елементів масиву, розташованих між максимальним по модулю мінімальним по модулю елементами, впорядкувати елементи масиву по спаданню.

//Файл main.cpp

#include <iostream>

#include "Windows.h"

#include <algorithm>

#include <cstdlib>

#include <vector>

#include <cmath>

using namespace std;

void ShowArr(int arr[],int n) {

for (int i = 0; i < n; i++) {

cout << arr[i] <<" ";

}

cout << endl;

}

bool is\_positive(int x) {

return x >= 0;

}

int SumPosistiveElement(vector<int> Vec) {

int Sum = 0;

for (int i = 0; i < Vec.size(); i++) {

Sum += Vec[i];

}

return Sum;

}

bool minMaxElementAbs(int a, int b) {

return abs(a) < abs(b);

}

int DobMinMaxElementAbs(int arr[],int n) {

int\* Min = min\_element(arr, arr + n, minMaxElementAbs);

int index\_min = distance(arr, Min);

int\* Max = max\_element(arr, arr + n, minMaxElementAbs);

int index\_max = distance(arr, Max);

cout <<"Індекс мінімального значення: " << index\_min << endl;

cout << "Індекс максимального значення: " << index\_max << endl;

int dob = 1;

if (index\_min < index\_max) {

for (int i = index\_min+1; i < index\_max; i++) {

dob\*= arr[i];

}

}

else {

for (int i = index\_max+1; i < index\_min; i++) {

dob \*= arr[i];

}

}

return dob;

}

bool sort\_comparate(int a, int b) {

return a > b;

}

int main(){

SetConsoleOutputCP(1251);

SetConsoleCP(1251);

srand(static\_cast<unsigned>(std::time(0)));

int n;

cout << "Введіть розмір масиву n: ";

cin >> n;

int\* arr=new int[n];

for (int i = 0; i < n; i++) {

arr[i] = -100 + rand() % 200;

}

cout << "Масив рандомних чисел:" << endl;

ShowArr(arr, n);

vector<int> Vec;

copy\_if(arr, arr+n,back\_inserter(Vec),is\_positive);

cout << "Сума всіх додатіх елементів масиву: " << SumPosistiveElement(Vec)<<endl;

cout << "Добуток всіх елементів між мінімальним і максимальним по модулю: " << DobMinMaxElementAbs(arr, n) << endl;

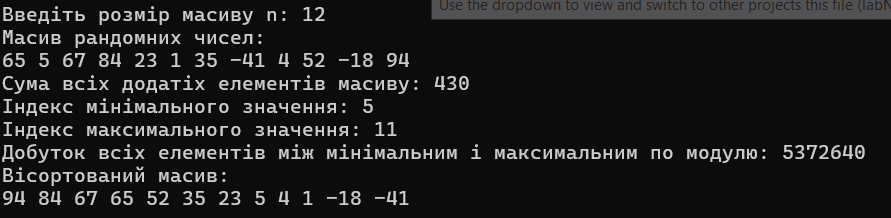
sort(arr, arr + n, sort\_comparate);

cout << "Вісортований масив: " << endl;

ShowArr(arr, n);

delete[] arr;

}

**Результати виконання програми:**

**Завдання 2.**

1.Створити контейнер, що містить об’єкти користувацького типу. Тип контейнера вибирається відповідно до варіанта завдання.

2. Відсортувати його по спаданню елементів.

3. Переглянути контейнер.

4. Використовуючи відповідний алгоритм, знайти в контейнері елемент, що задовольняє заданій умові.

5. Перемістити елементи, що задовольняють заданій умові в іншій (попередньо порожній) контейнер. Тип другого контейнера визначається варіантом завдання.

6. Переглянути другий контейнер.

7. Відсортувати перший і другий контейнери по зростанню елементів.

8. Переглянути їх.

9. Одержати третій контейнер шляхом злиття перших двох.

10. Переглянути третій контейнер.

11. Підрахувати, скільки елементів, що задовольняють заданій умові, містить третій контейнер.

12. Визначити, є чи в третьому контейнері елемент, що задовольняє заданій умові.

***Зауваження:***

1. Для створення другого контейнера в програмі No3 можна використати або

алгоритм remove\_copy\_if, або визначити свій алгоритм copy\_if, якого немає в STL.

2. Для пошуку елемента в колекції можна використати алгоритм find\_if, або

for\_each, або binary\_search, якщо контейнер відсортований.

3. Умови пошуку й заміни елементів вибираються самостійно й для них пишеться функція – п

редикат.

4. Деякі алгоритми можуть не підтримувати використовувані у вашій програмі контейнери. Наприклад, алгоритм sort не підтримує контейнери, які не мають ітераторів довільного доступу. У цьому випадку варто написати свій алгоритм. Наприклад, для стека алгоритм сортування може виконуватися в такий спосіб: переписати стек у вектор, відсортувати вектор, переписати вектор у стек.

5. Для порівняння елементів при сортуванні по зростанню використовується операція <, що повинна бути перевантажена в користувацькому класі. Для сортування по убуванню варто написати функцію comp і використати другу версію алгоритму sort.



//Файл Lab\_8.h

#pragma once

#include <iostream>

using namespace std;

class Lab\_8{

float variable;

public:

Lab\_8();

Lab\_8(float variable);

friend ostream& operator<<(ostream& os, Lab\_8& ob);

bool operator>(Lab\_8& ob);

bool operator>(int i);

bool operator<(Lab\_8& ob);

bool operator==(int i);

float Variable() const;

};

//Файл Lab\_8.cpp

#include "Lab\_8.h"

#include <cstdlib>

#include <iostream>

using namespace std;

Lab\_8::Lab\_8() {

variable = -100 + static\_cast<float>(rand()) / (static\_cast<float>(RAND\_MAX / (200))) ;

}

Lab\_8::Lab\_8(float variable) {

this->variable = variable;

}

ostream& operator<<(ostream& os, Lab\_8& ob) {

os << "Значення об'єкту: " << ob.variable<<endl;

return os;

}

bool Lab\_8::operator>(Lab\_8 & ob) {

return this->variable > ob.variable;

}

float Lab\_8::Variable()const {

return variable;

}

bool Lab\_8::operator>(int i) {

return this->variable > i;

}

bool Lab\_8::operator<(Lab\_8& ob) {

return this->variable < ob.variable;

}

bool Lab\_8::operator==(int i) {

return this->variable == i;

}

//Файл main.cpp

#include <iostream>

#include "Windows.h"

#include <algorithm>

#include <cstdlib>

#include <deque>

#include "Lab\_8.h"

#include <vector>

#include <stack>

using namespace std;

template<class T>

void ShowArr(T Deque) {

for (int i = 0; i < Deque.size(); i++) {

cout << Deque[i] << " ";

}

cout << endl;

}

bool SortDesc(Lab\_8 a, Lab\_8 b) {

return a > b;

}

bool SortLess(Lab\_8 a, Lab\_8 b) {

return a < b;

}

bool searchh(Lab\_8 ob) {

const float eps = 1e-5;

return abs(ob.Variable() - 12.4f) < eps;

}

bool CopyStack(Lab\_8& ob) {

return ob > 0;

}

stack<Lab\_8> Displacement(vector<Lab\_8> tmpVec) {

stack<Lab\_8> Stack;

for (int i = 0; i < tmpVec.size(); i++) {

Stack.push(tmpVec[i]);

}

return Stack;

}

void ShowStack(stack<Lab\_8> Stack) {

while (!Stack.empty()) {

cout << "Значення об'єкта: " << Stack.top() << endl;

Stack.pop();

}

}

stack<Lab\_8> SortStackLess(stack<Lab\_8>& Stack) {

vector<Lab\_8> vec;

while (!Stack.empty()) {

vec.push\_back(Stack.top());

Stack.pop();

}

sort(vec.begin(), vec.end(), SortDesc);

stack<Lab\_8> Stack1 = Displacement(vec);

return Stack1;

}

vector<Lab\_8> AssociationDequeAndStack(deque<Lab\_8> Deque, stack<Lab\_8> Stack) {

vector<Lab\_8> tmpVec;

copy(Deque.begin(), Deque.end(), back\_inserter(tmpVec));

while (!Stack.empty()) {

tmpVec.push\_back(Stack.top());

Stack.pop();

}

return tmpVec;

}

bool Search12\_4(Lab\_8& ob) {

float epsilon = 1e-5;

return abs(ob.Variable()-12.4)<epsilon;

}

int main(){

SetConsoleOutputCP(1251);

SetConsoleCP(1251);

srand(static\_cast<unsigned>(std::time(0)));

deque<Lab\_8> Deque;

Lab\_8 ob[10];

Lab\_8 ob1(12.4);

for (int i = 0; i < 10; i++) {

Deque.push\_back(ob[i]);

}

Deque.insert(Deque.begin() + Deque.size() / 2, ob1);

cout << "Виведення черги: " << endl;

ShowArr(Deque);

sort(Deque.begin(), Deque.end(), SortDesc);

cout << "Вісортована черга: " << endl;

ShowArr(Deque);

deque<Lab\_8>::iterator it=find\_if(Deque.begin(), Deque.end(), searchh);

int index = distance(Deque.begin(), it);

cout << "Індекс шуканого елемента: " << index << endl;

cout << "Шуканий елемент: "<<endl<<Deque[index ]<< endl;

vector<Lab\_8> tmpVec;

copy\_if(Deque.begin(), Deque.end(),back\_inserter(tmpVec), CopyStack);

stack<Lab\_8> Stack;

Stack = Displacement(tmpVec);

cout << "Виведення стеку: " << endl;

ShowStack(Stack);

sort(Deque.begin(), Deque.end(), SortLess);

cout << "Черга відсортована за зростанням: " << endl;

ShowArr(Deque);

cout << "Виведення відсортованого стеку: " << endl;

Stack=SortStackLess(Stack);

ShowStack(Stack);

cout << "Виведення вектора що об'єднує двонапрямлену чергу та стек: " << endl;

vector<Lab\_8> Three = AssociationDequeAndStack(Deque, Stack);

ShowArr(Three);

cout<<"Кількість значень в третьому контейнері>0: "<<count\_if(Three.begin(), Three.end(), CopyStack);

vector<Lab\_8>::iterator it1=find\_if(Three.begin(), Three.end(), Search12\_4);

if (it1 != Three.end()) {

cout<<endl << "Шуканий елент є у контейнері під індексом: " << distance(Three.begin(), it1) << endl;

cout << "Кількість цих елементі у контейнері: " <<count\_if(Three.begin(),Three.end(), Search12\_4)<< endl;

}

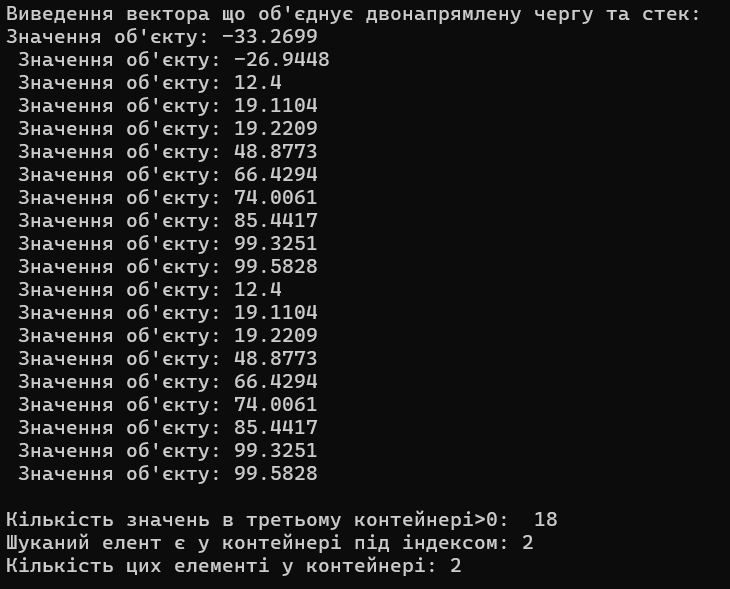
else {

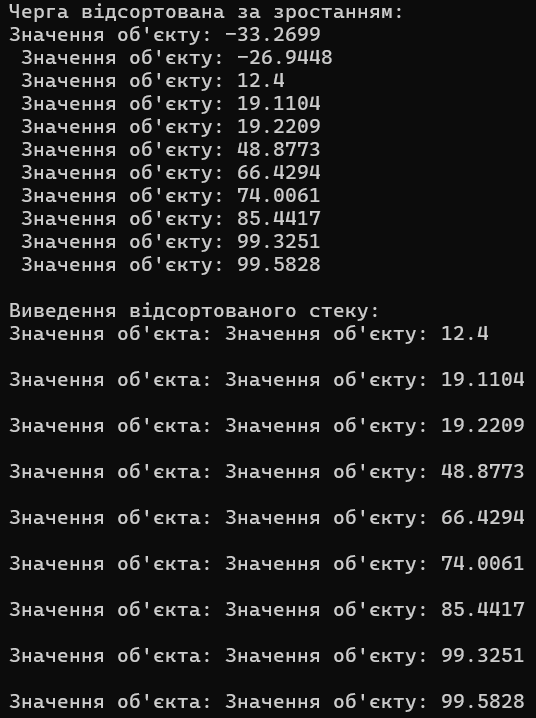
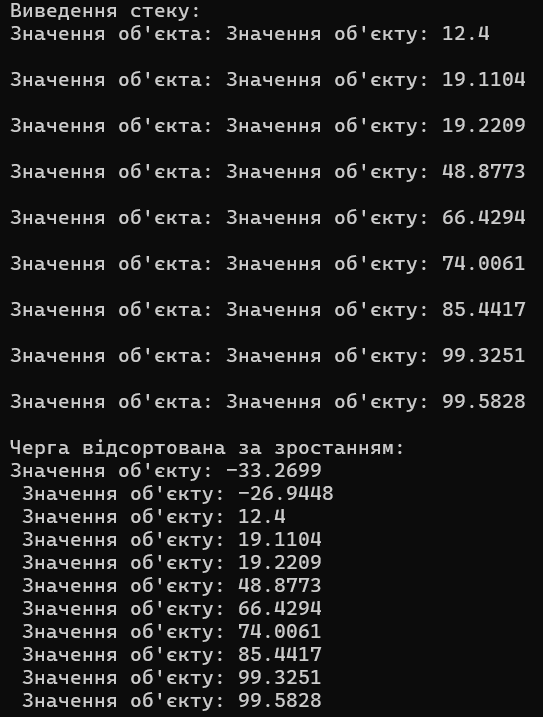
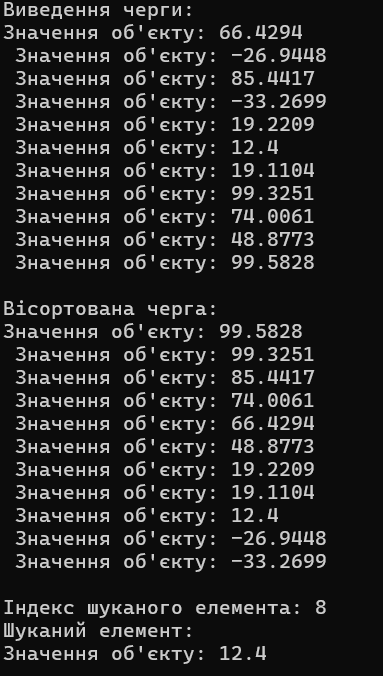
cout<<endl << "В третьому контейнері немає цього елементу" << endl;

}

}

**Результати виконання програми:**





**Контрольні запитання**

1. **На які категорії поділяються алгоритми STL?**

Операції що не модифікують, що модифікують, сортування, мінімуми і максимуми, для роботи з множинами.

1. **Яку бібліотеку слід підключити для роботи з алгоритмами STL?**

#include<algorithm>

1. **Яка різниця між алгоритмами групи модифіковані операції з послідовностями і немодифіковані операції з послідовностями?**

Вони відрізняються тим, що одні модифікують контейнери а інші ні тобто змінюють їх.

1. **Які алгоритми сортування STL Ви знаєте? Наведіть приклад.**

sort(), partial\_sort(), stable\_sort()

1. **Яка різниця між алгоритмом count\_if і count?**

count: підраховує кількість елементів у послідовності, які дорівнюють заданому значенню.

count\_if: підраховує кількість елементів у послідовності, які задовольняють певну умову або предикат.

1. **Для чого призначений алгоритм for\_each?**

for\_each використовується для застосування певної функції або операції до кожного елемента у послідовності

1. **Яка різниця між алгоритмами iter\_swap, swap і swap\_ranges?**

iter\_swap: обмінює значення двох елементів, вказаних їх ітераторами.

swap: обмінює значення двох змінних.

swap\_ranges: обмінює елементи двох діапазонів (послідовностей) тієї ж довжини.

1. **Які алгоритми переміщення Ви знаєте?**

move: Переміщує ресурси з одного об'єкта до іншого без копіювання.

move\_backward: Переміщує елементи з одного діапазону в інший, починаючи з кінця діапазону, щоб уникнути перекриття.

copy: Копіює елементи з одного діапазону в інший.