**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»**

Інститут **ІКНІ**

Кафедра **ПЗ**

# ЗВІТ

До лабораторної роботи №11

На тему: «Стандартна бібліотека шаблонів.  Контейнери та алгоритми»

З дисципліни «Об’єктно-орієнтоване програмування»

**Лектор:** доцент каф. ПЗ

Коротєєва Т.О.

**Виконала:** ст.гр. ПЗ-23

Кохман О.В.

**Прийняла:** доцент каф. ПЗ

Коротєєва Т.О.

«_____»_____2022р.

∑ ____ _____.

Львів – 2022

**Тема:** Стандартна бібліотека шаблонів. Контейнери та алгоритми.

**Мета:** Навчитись використовувати контейнери стандартної бібліотеки шаблонів та вбудовані алгоритми.

## Теоретичні відомості

Стандартна бібліотека шаблонів (STL, “Standard Template Library”) — це частина Стандартної бібліотеки C++, яка містить набір шаблонів контейнерних класів (наприклад, **std::vector** і **std::array**), алгоритмів і ітераторів. Спочатку вона була сторонньою розробкою, але пізніше була включена в Стандартну бібліотеку C++. Якщо вам потрібен якийсь загальний клас чи алгоритм, то, швидше за все, в Стандартній бібліотеці шаблонів він вже є. Круто також те, що ви можете використовувати ці класи без необхідності писати і відлагоджувати їх самостійно (і розбиратися в їх реалізації). Крім того, ви отримуєте досить ефективні (і вже багато разів протестовані) версії цих класів. Недоліком є те, що не все так просто/очевидно з функціоналом.

Безумовно, найбільш використовуваним функціоналом **бібліотеки STL** є **контейнерні класи** (або як їх ще називають — ***«контейнери»***). Бібліотека STL містить багато різних контейнерних класів, які можна використовувати в різних ситуаціях. Якщо говорити в загальному, то **контейнери STL діляться на три основні категорії**: послідовні; асоціативні; адаптери.

**Послідовні контейнери** (або ***«контейнери послідовності»***) — це контейнерні класи, елементи яких знаходяться в послідовності. Їх визначальною характеристикою є те, що ви можете додати свій елемент в будь-яке місце контейнера. Найбільш поширеним прикладом послідовного контейнера є **масив**: при додаванні 4 елементів в масив, ці елементи перебуватимуть (в масиві) в точно такому ж порядку, в якому ви їх додали.

Починаючи з C++11, **STL містить 6 контейнерів послідовності**:

std::vector; std::deque; std::array;

std::list; std::forward_list; std::basic_string.

**Асоціативні контейнери** — це контейнерні класи, які автоматично сортують всі свої елементи (в тому числі і ті, які додаєте ви). За замовчуванням асоціативні контейнери виконують сортування елементів, використовуючи оператор порівняння <.

**set** — це контейнер, в якому зберігаються тільки унікальні елементи, і повторення заборонені. Елементи упорядковано відповідно до їх значень.

**multiset** — це set, але в якому допускаються повторювані елементи.

**map** (або ***«асоціативний масив»***) — це set, в якому кожен елемент є парою "ключ-значення". "Ключ" використовується для сортування та індексації даних і повинен бути унікальним, а "значення" — це фактичні дані.

**multimap** (або ***«словник»***) — це map, який допускає дублювання ключів. Всі ключі відсортовані в порядку зростання, і ви можете подивитися значення по ключу.

**Адаптери** — це спеціальні визначені контейнерні класи, які адаптовані для виконання конкретних завдань. Найцікавіше полягає в тому, що ви самі можете вибрати, який послідовний контейнер повинен використовувати адаптер.

**stack** (**стек**) — це контейнерний клас, елементи якого працюють за принципом **LIFO** (англ. *«**L**ast **I**n, **F**irst **O**ut» = «останнім прийшов, першим пішов»*), тобто елементи додаються (вносяться) в кінець контейнера і видаляються (виштовхуються) звідти ж (з кінця контейнера). Зазвичай в стеках використовується deque в якості послідовного контейнера за замовчуванням (що трохи дивно, оскільки vector був би більш підходящим варіантом), але ви також можете використовувати vector або list.

**queue** (**черга**) — це контейнерний клас, елементи якого працюють за принципом **FIFO** (англ. *«**F**irst **I**n, **F**irst **O**ut» = «першим прийшов, першим пішов»*), тобто елементи додаються (вносяться) в кінець контейнера, але видаляються (виштовхуються) з початку контейнера. За замовчуванням в черзі використовується deque в якості послідовного контейнера, але також може використовуватися і list.

**priority_queue** (**черга з пріоритетом**) — це тип черги, в якій всі елементи відсортовані (за допомогою оператора порівняння <). При додаванні елемента, він автоматично сортується. Елемент з найвищим пріоритетом (найбільший елемент) знаходиться на самому початку черги з пріоритетом, також, як і видалення елементів виконується з самого початку черги з пріоритетом.

## Індивідуальне завдання

Написати програму з використанням бібліотеки STL.

В програмі реалізувати наступні функції:

1. Створити об’єкт-контейнер (**1**) у відповідності до індивідуального варіанту і заповнити його даними користувацього типу, згідно варіанту.

2. Вивести контейнер.

3. Змінити контейнер, видаливши з нього одні елементи і замінивши інші.

4. Проглянути контейнер, використовуючи для доступу до його елементів ітератори.

5. Створити другий контейнер цього ж класу і заповнити його даними того ж типу, що і перший контейнер.

6. Змінити перший контейнер, видаливши з нього **n** елементів після заданого і добавивши опісля в нього всі елементи із другого контейнера.

7. Вивести перший і другий контейнери.

8. Відсортувати контейнер по спаданню елементів та вивести результати.

9. Використовуючи необхідний алгоритм, знайти в контейнері елемент, який задовільняє заданій умові.

10. Перемістити елементи, що задовільняють умові в інший, попередньо пустий контейнер (**2**). Тип цього контейнера визначається згідно варіанту.

11. Проглянути другий контейнер.

13. Відсортувати перший і другий контейнери по зростанню елементів, вивести результати.

15. Отримати третій контейнер шляхом злиття перших двох.

16. Вивести на екран третій контейнер.

17. Підрахувати, скільки елементів, що задовільнюють заданій умові, містить третій контейнер.

Оформити звіт до лабораторної роботи. Звіт має містити варіант завдання, код розробленої програми, результати роботи програми (скріншоти), висновок.

| 2 | stack | queue | float |
|---|---|---|---|

## Код програми

Назва файлу: MyForm.h

```
#pragma once
#include <stack>
#include <algorithm>
#include <queue>
#include <random>
using namespace std;
namespace ProjectMain {
	using namespace System;
	using namespace System::ComponentModel;
	using namespace System::Collections;
	using namespace System::Windows::Forms;
	using namespace System::Data;
	using namespace System::Drawing;
	public ref class MyForm : public System::Windows::Forms::Form
	{
	public:
		MyForm(void)
		{
			InitializeComponent();
		}
	protected:
		~MyForm()
		{
			if (components)
			{
				delete components;
			}
		}
	private: System::Windows::Forms::Button^ button1;
	protected:
	private: System::Windows::Forms::TextBox^ textBox1;
	private: System::Windows::Forms::Button^ button2;
	private: System::Windows::Forms::RichTextBox^ richTextBox1;
	private: System::Windows::Forms::Button^ button3;
	private: System::Windows::Forms::Button^ button4;
	private: System::Windows::Forms::Button^ button5;
	private: System::Windows::Forms::Button^ button6;
	private: System::Windows::Forms::Button^ button7;
	private: System::Windows::Forms::Button^ button9;
	private: System::Windows::Forms::Button^ button8;
	private: System::Windows::Forms::Button^ button10;
	private: System::Windows::Forms::Button^ button11;
	private: System::Windows::Forms::Button^ button12;
	private: System::Windows::Forms::Button^ button13;
	private: System::Windows::Forms::Button^ button14;
	private: System::Windows::Forms::Button^ button15;
	private: System::Windows::Forms::Button^ button16;
	private: System::Windows::Forms::Button^ button17;
	private: System::Windows::Forms::Button^ button18;
	private:
```

```
            System::ComponentModel::Container ^components;
#pragma region Windows Form Designer generated code

#pragma endregion
	stack<float>* first;
	int firstLength = 0;
	stack<float>* second;
	queue<float>* queueFirst;
	queue<float>* merged;
public: void printArray(System::Windows::Forms::RichTextBox^ richTextBox,
stack<float>* object) {
	int length = object->size();
	float* array = new float[length];
	for (int i = 0; i < length; i++) {
		array[i] = object->top();
		object->pop();
	}
	for (int i = 0; i < length; i++) {
		richTextBox->Text += array[i].ToString("#,0.00") + " ";
	}
	richTextBox->Text += "\n";
	for (int i = length - 1; i >= 0; i--) {
		object->push(array[i]);
	}
}
private: System::Void createButton(System::Object^ sender, System::EventArgs^ e)
{
	first = new stack<float>;
	firstLength = System::Convert::ToInt64(textBox1->Text);
	random_device random_device;
	mt19937 generator(random_device());
	uniform_real_distribution<float> distribution(1, 20);
	while (first->size() < firstLength) {
		first->push(distribution(generator));
	}
	button1->Enabled = false;
}
private: System::Void PrintButton(System::Object^ sender, System::EventArgs^ e) {
	printArray(richTextBox1, first);
	button2->Enabled = false;
}
private: System::Void changeButton(System::Object^ sender, System::EventArgs^ e)
{
	int count = first->size() / 2;
	while (first->size() > count) {
		first->pop();
	}
	float length = first->size();
	float* array = new float[length];
	for (int i = 0; i < length; i++) {
		array[i] = first->top() + 1;
		first->pop();
	}
	for (int i = length - 1; i >= 0; i--) {
		first->push(array[i]);
	}
	button3->Enabled = false;
}
private: System::Void create2Button(System::Object^ sender, System::EventArgs^ e)
{
	second = new stack<float>;
	random_device random_device;
	mt19937 generator(random_device());
```

```
	uniform_real_distribution<float> distribution(1, 20);
	while (second->size() < firstLength) {
		second->push(distribution(generator));
	}
	button4->Enabled = false;
}
private: System::Void PrintStack2(System::Object^ sender, System::EventArgs^ e) {
	printArray(richTextBox1, second);
	button6->Enabled = false;
}
private: System::Void changeButton2(System::Object^ sender, System::EventArgs^ e)
{
	int count2 = first->size() / 2;
	int count3 = first->size() - count2;
	float* array2 = new float[count3];
	for (int i = 0; i < count2; i++) {
		first->pop();
	}
	for (int i = 0; i < count3; i++) {
		array2[i] = first->top();
		first->pop();
	}
	for (int i = count3 - 1; i >= 0; i--) {
		first->push(array2[i]);
	}
	float* array3 = new float[firstLength];
	for (int i = 0; i < firstLength; i++) {
		array3[i] = second->top();
		second->pop();
	}
	for (int i = firstLength - 1; i >= 0; i--) {
		second->push(array3[i]);
		first->push(array3[i]);
	}
	button7->Enabled = false;
}
private: System::Void PrintBoth(System::Object^ sender, System::EventArgs^ e) {
	printArray(richTextBox1, first);
	printArray(richTextBox1, second);
	button9->Enabled = false;
}
private: System::Void sortButton(System::Object^ sender, System::EventArgs^ e) {
	int length2 = first->size();
	float* array4 = new float[length2];
	for (int i = 0; i < length2; i++) {
		array4[i] = first->top();
		first->pop();
	}
	sort(array4, array4 + length2);
	for (int i = length2 - 1; i >= 0; i--) {
		first->push(array4[i]);
	}
	button8->Enabled = false;
}
private: System::Void findMin(System::Object^ sender, System::EventArgs^ e) {
	richTextBox1->Text += "min = " + first->top().ToString("#,0.00") + "\n";
	button11->Enabled = false;
}
private: System::Void createQueue(System::Object^ sender, System::EventArgs^ e) {
	float* array5 = new float[firstLength];
	for (int i = 0; i < firstLength; i++) {
		array5[i] = first->top();
		first->pop();
```

```
	}
	for (int i = firstLength - 1; i >= 0; i--) {
		first->push(array5[i]);
	}
	queueFirst = new queue<float>;
	for (int i = 0; i < firstLength; i++) {
		if (array5[i] > 5 && array5[i] < 15) {
			queueFirst->push(array5[i]);
		}
	}
	button12->Enabled = false;
}
private: System::Void PrintQueue(System::Object^ sender, System::EventArgs^ e) {
	printQueue(richTextBox1, queueFirst);
	button13->Enabled = false;
}
public: void printQueue(System::Windows::Forms::RichTextBox^ richTextBox,
queue<float>* object) {
	int length = object->size();
	float* array = new float[length];
	for (int i = 0; i < length; i++) {
		array[i] = object->front();
		object->pop();
	}
	for (int i = 0; i < length; i++) {
		richTextBox->Text += array[i].ToString("#,0.00") + " ";
	}
	richTextBox->Text += "\n";
	for (int i = 0; i < length; i++) {
		object->push(array[i]);
	}
}
private: System::Void sortBoth(System::Object^ sender, System::EventArgs^ e) {
	int length2 = first->size();
	float* array4 = new float[length2];
	for (int i = 0; i < length2; i++) {
		array4[i] = first->top();
		first->pop();
	}
	sort(array4, array4 + length2);
	for (int i = 0; i < length2; i++) {
		first->push(array4[i]);
	}
	int length3 = queueFirst->size();
	float* array6 = new float[length3];
	for (int i = 0; i < length3; i++) {
		array6[i] = queueFirst->front();
		queueFirst->pop();
	}
	sort(array6, array6 + length3);
	for (int i = 0; i < length3; i++) {
		queueFirst->push(array6[i]);
	}
	button14->Enabled = false;
}
private: System::Void printQueueStack(System::Object^ sender, System::EventArgs^
e) {
	printArray(richTextBox1, first);
	printQueue(richTextBox1, queueFirst);
	button15->Enabled = false;
}
private: System::Void mergeButton(System::Object^ sender, System::EventArgs^ e) {
	merged = new queue<float>;
```

```
	while (!first->empty()) {
		merged->push(first->top());
		first->pop();
	}
	while (!queueFirst->empty()) {
		merged->push(queueFirst->front());
		queueFirst->pop();
	}
	button16->Enabled = false;
}
private: System::Void printMerged(System::Object^ sender, System::EventArgs^ e) {
	printQueue(richTextBox1, merged);
	button17->Enabled = false;
}
private: System::Void counterButton(System::Object^ sender, System::EventArgs^ e)
{
	int counter = 0;
	while (!merged->empty()) {
		if (merged->front() > 5 && merged->front() < 15) {
			counter++;
		}
		merged->pop();
	}
	richTextBox1->Text += "counter = " + System::Convert::ToString(counter);
	button18->Enabled = false;
}
private: System::Void PrintButtonn(System::Object^ sender, System::EventArgs^ e)
{
	printArray(richTextBox1, first);
	button5->Enabled = false;
}
private: System::Void PrintButton11(System::Object^ sender, System::EventArgs^ e)
{
	printArray(richTextBox1, first);
	button10->Enabled = false;
}
};
}
```

Назва файлу: MyForm.cpp

```
#include "MyForm.h"
using namespace ProjectMain;
int main() {
    Application::EnableVisualStyles();
    Application::SetCompatibleTextRenderingDefault(false);
    Application::Run(gcnew MyForm());
    return 0;
}
```

## Протокол роботи

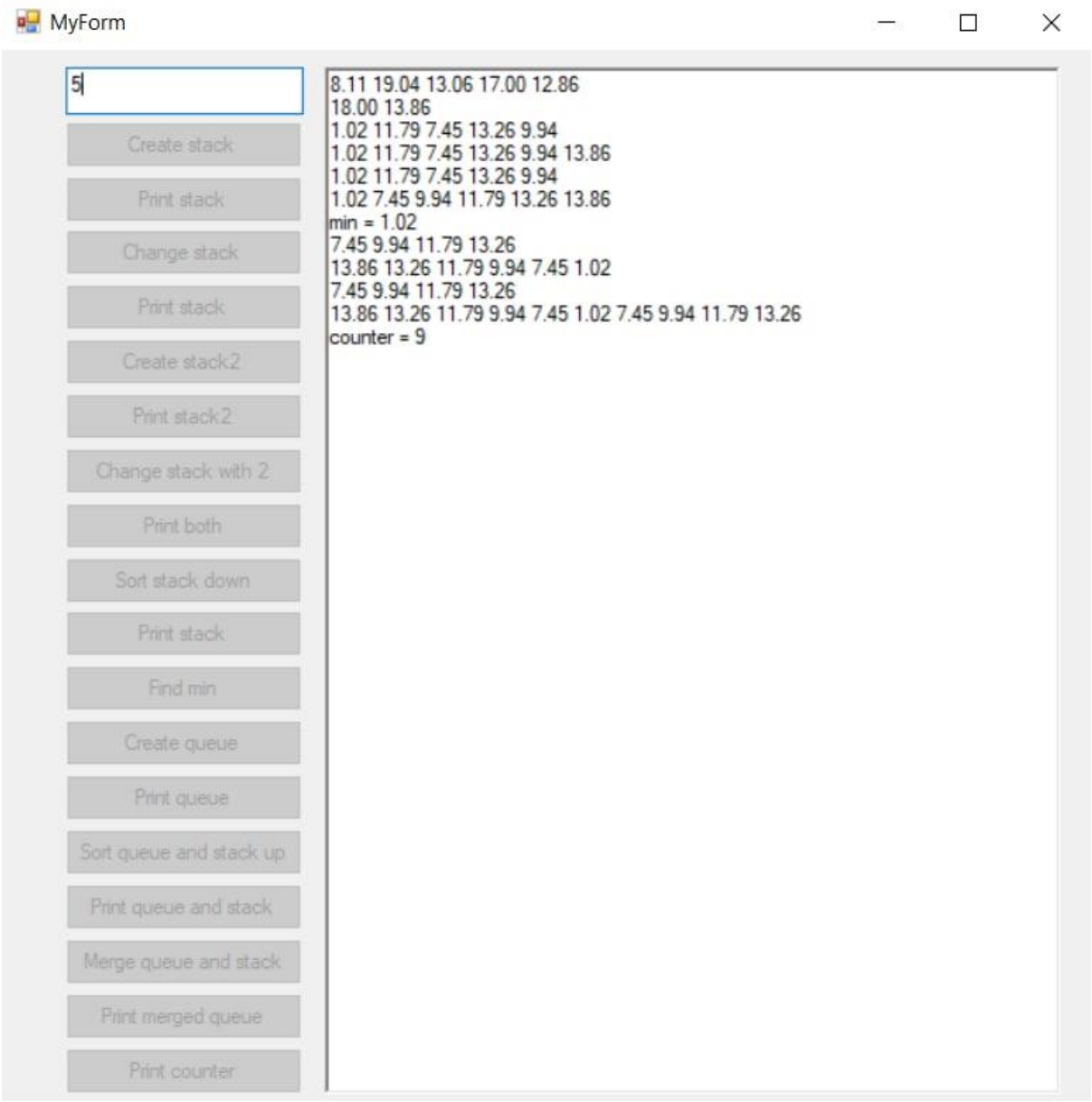


Рис. 1 Результат роботи програми.

## Висновок

На цій лабораторній роботі я дізналась про стандартну бібліотеку шаблонів в с++, також дізналась про бібліотеку алгоритмів і реалізувала за допомогою цих двох бібліотек програму та продемонструвала отримані результати на формі у Visual Studio 2022.