Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Інститут комп’ютерних наук та інформаційних технологій

Кафедра програмного забезпечення



**ЗВІТ**

**Про виконання лабораторної роботи № 11**

*«Стандартна бібліотека шаблонів. Контейнери та алгоритми»*

**з дисципліни «***Об’єктно–орієнтоване програмування***»**

**Лектор:**

ст. викладач кафедри ПЗ

Коротєєва Т.О.

**Виконав:**

студ. групи ПЗ-12

Артех О.Ю.

**Прийняла:**

асистент кафедри ПЗ

Терендій О.В.

«\_\_\_» \_\_\_\_\_\_ 2020 р.

∑ = \_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_

Львів – 2020

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

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

**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 задовільянють заданій умові, містить третій контейнер.

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

**Хід роботи**

Код програми:

**functions.h**

#ifndef FUNCTIONS\_H

#define FUNCTIONS\_H

#include <iostream>

#include <iomanip>

#include <string>

#include <sstream>

#include <stack> *//* *first* *container*

#include <queue> *//* *second* *container*

*//* *basic* *type* *float*

*using* *namespace* std;

*//----------------------------------------------------------------------------------------------------------------------*

float **RandomFloat**(float min, float max);

bool **condition**(float temp);

void **swap**(float \*xp, float \*yp);

void **in\_bubbleSort**(float arr[], int n);

void **de\_bubbleSort**(float arr[], int n);

*//----------------------------------------------------------------------------------------------------------------------*

void **print\_stack**(stack<float> &temp);

void **add\_n\_random\_to\_stack**(stack<float> &temp, int n);

void **delete\_n\_top\_from\_stack**(stack<float> &temp, int n);

void **stack\_change**(stack<float> &s1, stack<float> &s2, int pos, int n);

void **sort\_stack\_by\_decrease**(stack<float> &temp);

void **sort\_stack\_by\_increase**(stack<float> &temp);

float \***search\_for\_el\_in\_stack**(stack<float> &temp);

void **add\_stacks**(stack<float> &s1, stack<float> &s2, queue<float> &q);

string **stack\_to\_str**(stack<float> &temp);

*//----------------------------------------------------------------------------------------------------------------------*

void **from\_stack\_to\_queue**(stack<float> &s, queue<float> &q);

void **print\_queue**(queue<float> &temp);

void **sort\_queue\_by\_increase**(queue<float> &temp);

int **number\_of\_el\_in\_queue**(queue<float> &temp);

string **queue\_to\_str**(queue<float> &temp);

*//----------------------------------------------------------------------------------------------------------------------*

#endif *//* *FUNCTIONS\_H*

**mainwindow.h**

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

QT\_BEGIN\_NAMESPACE

*namespace* **Ui** { *class* **MainWindow**; }

QT\_END\_NAMESPACE

*class* **MainWindow** : *public* QMainWindow

{

Q\_OBJECT

*public*:

**MainWindow**(QWidget \*parent = *nullptr*);

~***MainWindow***();

*private* slots:

void **on\_b1\_clicked**();

void **on\_b2\_clicked**();

void **on\_b3\_clicked**();

void **on\_b4\_clicked**();

void **on\_b5\_clicked**();

void **on\_b6\_clicked**();

void **on\_b7\_clicked**();

void **on\_b8\_clicked**();

void **on\_b9\_clicked**();

void **on\_b10\_clicked**();

*private*:

Ui::MainWindow \*ui;

};

#endif *//* *MAINWINDOW\_H*

**functions.cpp**

#include "functions.h"

float **RandomFloat**(float min, float max) {

float random = ((float) rand()) / (float) RAND\_MAX;

float range = max - min;

*return* (random \* range) + min;

}

bool **condition**(float temp) {

*return* (temp > 2);

}

void **swap**(float \*xp, float \*yp) {

float temp = \*xp;

\*xp = \*yp;

\*yp = temp;

}

void **in\_bubbleSort**(float arr[], int n) {

int i, j;

bool swapped;

*for* (i = 0; i < n - 1; i++) {

swapped = *false*;

*for* (j = 0; j < n - i - 1; j++) {

*if* (arr[j] > arr[j + 1]) {

swap(&arr[j], &arr[j + 1]);

swapped = *true*;

}

}

*if* (!swapped)

*break*;

}

}

void **de\_bubbleSort**(float arr[], int n) {

int i, j;

bool swapped;

*for* (i = 0; i < n - 1; i++) {

swapped = *false*;

*for* (j = 0; j < n - i - 1; j++) {

*if* (arr[j] < arr[j + 1]) {

swap(&arr[j], &arr[j + 1]);

swapped = *true*;

}

}

*if* (!swapped)

*break*;

}

}

*//----------------------------------------------------------------------------------------------------------------------*

void **print\_stack**(stack<float> &temp) {

int size = temp.size();

float data[size];

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

cout << fixed << showpoint << setprecision(5) << temp.top() << " ";

data[i] = temp.top();

temp.pop();

}

*for* (int i = 0; i < size; i++)

temp.push(data[size - i - 1]);

cout << endl;

}

void **add\_n\_random\_to\_stack**(stack<float> &temp, int n) {

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

temp.push(RandomFloat(0, 10));

}

void **delete\_n\_top\_from\_stack**(stack<float> &temp, int n) {

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

temp.pop();

}

void **stack\_change**(stack<float> &s1, stack<float> &s2, int pos, int n) {

*auto* temp = *new* float[pos];

int s2\_size = s2.size();

*for* (int i = 0; i < pos; i++) {

temp[i] = s1.top();

s1.pop();

}

delete\_n\_top\_from\_stack(*s1*, n);

*auto* data\_s2 = *new* float[s2.size()];

*for* (int i = 0; i < s2\_size; i++) {

data\_s2[i] = s2.top();

s2.pop();

}

*for* (int i = 0; i < s2\_size; i++) {

s2.push(data\_s2[s2\_size - i - 1]);

s1.push(data\_s2[s2\_size - i - 1]);

}

*for* (int i = 0; i < pos; i++)

s1.push(temp[pos - i - 1]);

*delete*[] temp;

*delete*[] data\_s2;

}

void **sort\_stack\_by\_decrease**(stack<float> &temp) {

int size = temp.size();

*auto* arr = *new* float[size];

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

arr[i] = temp.top();

temp.pop();

}

de\_bubbleSort(arr, size);

*for* (int i = 0; i < size; i++)

temp.push(arr[size - i - 1]);

*delete*[] arr;

}

void **sort\_stack\_by\_increase**(stack<float> &temp) {

int size = temp.size();

*auto* arr = *new* float[size];

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

arr[i] = temp.top();

temp.pop();

}

in\_bubbleSort(arr, size);

*for* (int i = 0; i < size; i++)

temp.push(arr[size - i - 1]);

*delete*[] arr;

}

float \***search\_for\_el\_in\_stack**(stack<float> &temp) {

int size = temp.size();

*auto* arr = *new* float[size];

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

arr[i] = temp.top();

temp.pop();

}

*for* (int i = 0; i < size; i++)

temp.push(arr[size - i - 1]);

*for* (int i = 0; i < size; i++)

*if* (condition(arr[i]))

*return* &(arr[i]);

*delete*[] arr;

*return* *nullptr*;

}

void **add\_stacks**(stack<float> &s1, stack<float> &s2, queue<float> &q) {

int size1 = s1.size();

int size2 = s2.size();

*auto* temp1 = *new* float[size1];

*auto* temp2 = *new* float[size2];

*for* (int i = 0; i < size1; i++) {

temp1[i] = s1.top();

q.push(s1.top());

s1.pop();

}

*for* (int i = 0; i < size1; i++)

s1.push(temp1[size1 - i - 1]);

*for* (int i = 0; i < size2; i++) {

temp2[i] = s2.top();

q.push(s2.top());

s2.pop();

}

*for* (int i = 0; i < size2; i++)

s2.push(temp2[size2 - i - 1]);

*delete*[] temp1;

*delete*[] temp2;

}

string **stack\_to\_str**(stack<float> &temp) {

ostringstream stream;

int size = temp.size();

float data[size];

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

stream << fixed << showpoint << setprecision(5) << temp.top() << " ";

data[i] = temp.top();

temp.pop();

}

*for* (int i = 0; i < size; i++)

temp.push(data[size - i - 1]);

string str = stream.str();

*return* str;

}

*//----------------------------------------------------------------------------------------------------------------------*

void **from\_stack\_to\_queue**(stack<float> &s, queue<float> &q) {

int size = s.size();

*auto* arr = *new* float[size];

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

q.push(s.top());

arr[i] = s.top();

s.pop();

}

*for* (int i = 0; i < size; i++)

s.push(arr[size - i - 1]);

*delete*[] arr;

}

void **print\_queue**(queue<float> &temp) {

int size = temp.size();

*auto* arr = *new* float[size];

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

cout << fixed << showpoint << setprecision(5) << temp.front() << " ";

arr[i] = temp.front();

temp.pop();

}

*for* (int i = 0; i < size; i++)

temp.push(arr[i]);

cout << endl;

*delete*[] arr;

}

void **sort\_queue\_by\_increase**(queue<float> &temp) {

int size = temp.size();

*auto* arr = *new* float[size];

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

arr[i] = temp.front();

temp.pop();

}

in\_bubbleSort(arr, size);

*for* (int i = 0; i < size; i++)

temp.push(arr[i]);

*delete*[] arr;

}

int **number\_of\_el\_in\_queue**(queue<float> &temp) {

int size = temp.size();

*auto* arr = *new* float[size];

int elements = 0;

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

arr[i] = temp.front();

*if* (condition(arr[i]))

elements++;

temp.pop();

}

*for* (int i = 0; i < size; i++)

temp.push(arr[i]);

*delete*[] arr;

*return* elements;

}

string **queue\_to\_str**(queue<float> &temp){

ostringstream stream;

int size = temp.size();

*auto* arr = *new* float[size];

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

stream << fixed << showpoint << setprecision(5) << temp.front() << " ";

arr[i] = temp.front();

temp.pop();

}

*for* (int i = 0; i < size; i++)

temp.push(arr[i]);

*delete*[] arr;;

string str = stream.str();

*return* str;

}

**main.cpp**

#include "mainwindow.h"

#include <QApplication>

int **main**(int argc, char \*argv[])

{

QApplication a(*argc*, argv);

MainWindow w;

w.show();

*return* a.exec();

}

**mainwindow.cpp**

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include "functions.h"

stack<float> s1, s2;

queue<float> q1, q2;

MainWindow::**MainWindow**(QWidget \*parent)

: QMainWindow(parent)

, ui(*new* Ui::MainWindow)

{

ui->setupUi(*this*);

}

MainWindow::~***MainWindow***()

{

*delete* ui;

}

void MainWindow::**on\_b1\_clicked**()

{

add\_n\_random\_to\_stack(*s1*, 5);

QString temp = QString::fromStdString(stack\_to\_str(*s1*));

ui->stack1->setText(temp);

}

void MainWindow::**on\_b2\_clicked**()

{

delete\_n\_top\_from\_stack(*s1*,ui->b2\_n->text().toInt());

add\_n\_random\_to\_stack(*s1*,ui->b2\_m->text().toInt());

QString temp = QString::fromStdString(stack\_to\_str(*s1*));

ui->stack1->setText(temp);

}

void MainWindow::**on\_b3\_clicked**()

{

add\_n\_random\_to\_stack(*s2*, 5);

QString temp = QString::fromStdString(stack\_to\_str(*s2*));

ui->stack2->setText(temp);

}

void MainWindow::**on\_b4\_clicked**()

{

stack\_change(*s1*, *s2*, ui->b4\_pos->text().toInt(),ui->b4\_n->text().toInt());

QString temp1 = QString::fromStdString(stack\_to\_str(*s1*));

ui->stack1->setText(temp1);

QString temp2 = QString::fromStdString(stack\_to\_str(*s2*));

ui->stack2->setText(temp2);

}

void MainWindow::**on\_b5\_clicked**()

{

sort\_stack\_by\_decrease(*s1*);

QString temp = QString::fromStdString(stack\_to\_str(*s1*));

ui->stack1->setText(temp);

}

void MainWindow::**on\_b6\_clicked**()

{

ui->b6\_res->setNum(\*search\_for\_el\_in\_stack(*s1*));

}

void MainWindow::**on\_b7\_clicked**()

{

from\_stack\_to\_queue(*s1*,*q1*);

QString temp = QString::fromStdString(queue\_to\_str(*q1*));

ui->queue1->setText(temp);

}

void MainWindow::**on\_b8\_clicked**()

{

sort\_stack\_by\_increase(*s1*);

sort\_queue\_by\_increase(*q1*);

QString temp = QString::fromStdString(queue\_to\_str(*q1*));

ui->queue1->setText(temp);

QString temp1 = QString::fromStdString(stack\_to\_str(*s1*));

ui->stack1->setText(temp1);

}

void MainWindow::**on\_b9\_clicked**()

{

add\_stacks(*s1*,*s2*,*q2*);

QString temp = QString::fromStdString(queue\_to\_str(*q2*));

ui->queue2->setText(temp);

}

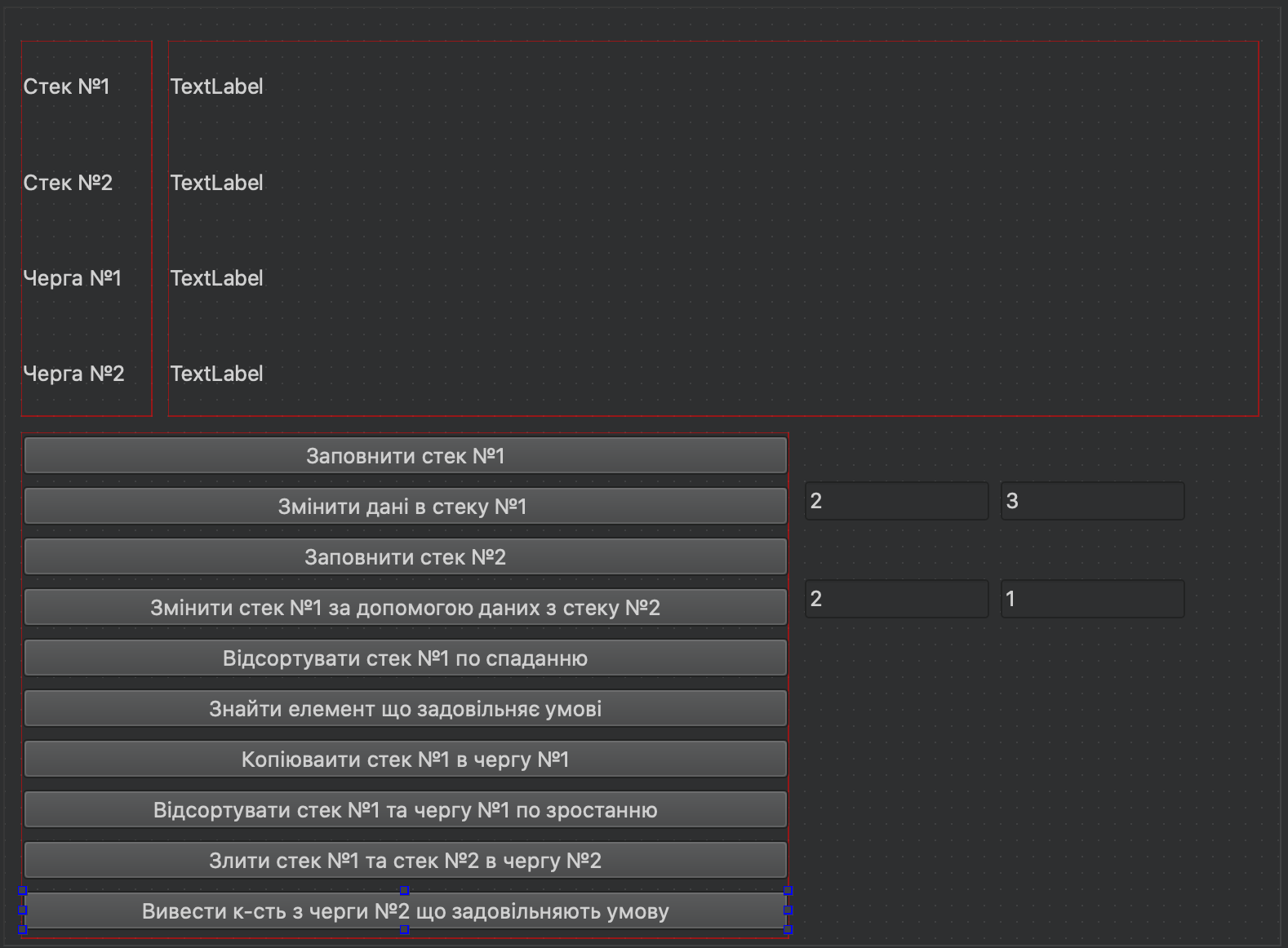
void MainWindow::**on\_b10\_clicked**()

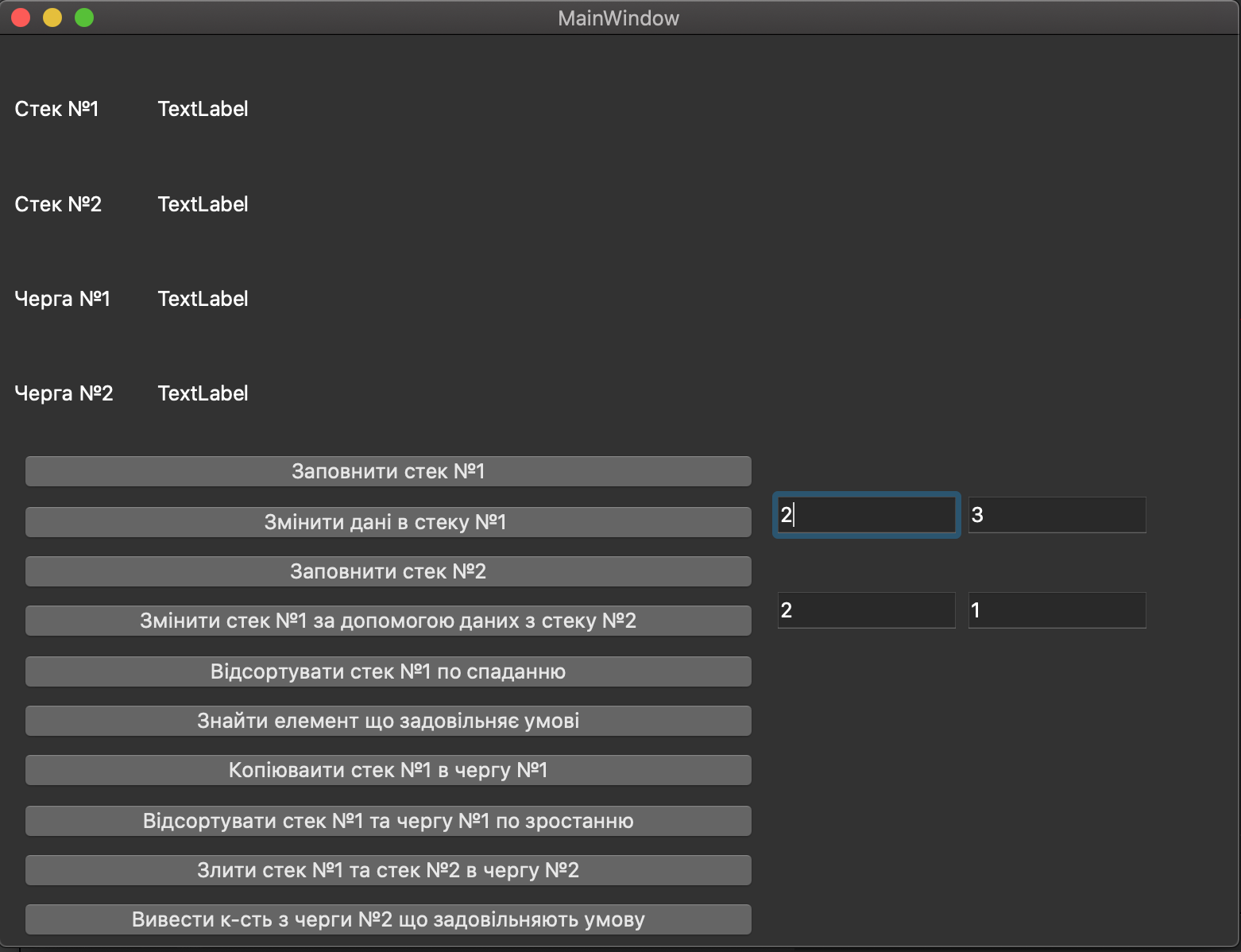
{

ui->b10\_res->setNum(number\_of\_el\_in\_queue(*q2*));

}

Результат виконання:





**Висновки**

Під час виконання лабораторної роботи я навчився використовувати STL, використовуючи її контейнери та алгоритми і отримані знання закріпив на практиці, показавши роботу з контейнерами stack і queue.