

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

**Лектор:**

доц. Коротєєва Т.О.

**Виконав:**

ст. гр. ПЗ-11

Солтисюк Д.А.

**Прийняла:**

доц. Коротєєва Т.О.

« \_\_ » \_\_\_\_\_ 2022 р.

$\Sigma$  = \_\_\_\_\_ .

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*)** – набір численних шаблонних класів мови C++, які надають розробникові найбільш гнучкі, поширені й використовувані інструменти для оперування даними. *STL* умовно розділяють на чотири частини за її вмістом – алгоритми, контейнери, функції, ітератори.

**Динамічні структури даних**– *STL* містить структури даних та методи для оперування даними, які поміщаються в них. У набір структур входять: list, stack, vector, deque, queue, map, array, ...

**Ітератори** – спосіб доступу до даних, які містяться у контейнерах. Шляхом поступового переміщення між елементами, ми можемо досягати до потрібного. У загальному розумінні, ітератор – це вказівник, проте має свої особливості як у плані використання, так і функціонування.

**Алгоритми** – *STL* вміщує велику кількість готових алгоритмів для оперування даними контейнерів. Найпоширеніший приклад алгоритму – сортування елементів у зростаючому\спадному порядку.

**Функції** – *STL* надає розробникові готові функції для виконання дій різної складності. Для кожного виду контейнера передбачений свій набір функцій. Найбазовішими та загальними для усіх є функції видалення та додавання.

## ЗАВДАННЯ ДО ЛАБОРАТОРНОЇ РОБОТИ

### Варіант 12

12	vector	map	char
----	--------	-----	------

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 користувацького 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 задовольняє заданій умові.
10. Перемістити елементи, що задовольняють 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 задовольняють заданій умові, містить третій контейнер.

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

# ТЕКСТ ПРОГРАМИ

## Файл widget.h

```
#pragma once

#include <QFile>
#include <QGridLayout>
#include <QPushButton>
#include <QTextEdit>
#include <QTextStream>
#include <QWidget>
#include <map>

class Widget : public QWidget {
    Q_OBJECT

    using OperatedType = const char *;

public:
    Widget(QWidget *parent = nullptr);
    void print_array();

    std::map<OperatedType, bool> map;
    std::vector<OperatedType> vector;
    std::vector<OperatedType> vector2;
private slots:
    void on_start();

private:
    QPushButton *start_btn;
    QTextEdit *output_1;
    QTextEdit *output_2;
    QTextEdit *output_3;
    QTextEdit *output_4;
};
```

## Файл main.cpp

```
#include <QApplication>

#include "widget.h"

int main(int argc, char *argv[]) {
    QApplication a(argc, argv);
    Widget w;
    w.show();
    return a.exec();
}
```

## Файл widget.cpp

```
#include "widget.h"
```

```

#include <algorithm>
#include <iostream>
#include <random>
#include <sstream>

template <typename T> std::string to_str(std::vector<T> &vec) {
    std::ostringstream oss;
    if (!vec.empty()) {
        std::copy(vec.begin(), vec.end() - 1, std::ostream_iterator<T>(oss, ", "));
        oss << vec.back();
    }
    return oss.str();
}

auto random_boolean = std::bind(std::uniform_int_distribution<>(0, 1),
                                std::default_random_engine());

std::string random_string(size_t length) {
    auto randchar = []() -> char {
        const char charset[] = "0123456789"
                                "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
                                "abcdefghijklmnopqrstuvwxyz";
        const size_t max_index = (sizeof(charset) - 1);
        return charset[rand() % max_index];
    };
    std::string str(length, 0);
    std::generate_n(str.begin(), length, randchar);
    return str;
}

struct filterChars {
    bool operator()(const char *s) { return strstr(s, "a") == s; }
};

void Widget::on_start() {
    map.empty();
    vector.clear();
    vector2.clear();

    for (auto i = 0; i < 10; i++) {
        vector.push_back(random_string(10).c_str());
        vector2.push_back(random_string(10).c_str());
    }
    this->output_1->setMarkdown(
        QString("### Vector1:\n%1").arg(QString::fromStdString(to_str(vector))));

    vector.erase(vector.begin() + 2, vector.begin() + 7);
    vector.at(3) = "Dima";
    vector.insert(vector.end(), vector2.begin(), vector2.end());
    std::sort(vector.begin(), vector.end(),
        [](const char *c1, const char *c2) { return strcmp(c1, c2) < 0; });
    auto find_res = std::find_if(vector.begin(), vector.end(), filterChars());

    this->output_2->setMarkdown(QString("### Vector1 sorted desc:\n%1"
                                        "\n### Vector2:\n%2"
                                        "\n### Find result:\n%3")
                                .arg(QString::fromStdString(to_str(vector)))
                                .arg(QString::fromStdString(to_str(vector2)))
                                .arg(QString(*find_res)));
}

```

```

);

std::for_each(vector.begin(), vector.end(), [this](const char *s) {
    if (filterChars()(s)) {
        map.insert({s, random_boolean()});
    }
});

std::vector<const char *> vector_from_map;
for (auto el : map) {
    vector_from_map.push_back(el.first);
}
std::sort(vector.begin(), vector.end());
std::sort(vector_from_map.begin(), vector_from_map.end());

this->output_3->setMarkdown(
    QString("### Vector1 sorted asc:\n%1"
            "\n### Map1 sorted asc:\n%2")
        .arg(QString::fromStdString(to_str(vector)))
        .arg(QString::fromStdString(to_str(vector_from_map))));

std::vector<const char *> merged_vector(vector.size() +
                                         vector_from_map.size());
std::merge(vector.begin(), vector.end(), vector_from_map.begin(),
            vector_from_map.end(), merged_vector.begin());

int count =
    std::count_if(merged_vector.begin(), merged_vector.end(), filterChars());
this->output_4->setMarkdown(
    QString("### Merged vector:\n%1"
            "\n### %2 elements satisfy filter")
        .arg(QString::fromStdString(to_str(merged_vector)))
        .arg(QString::number(count)));
}

Widget::Widget(QWidget *parent) : QWidget(parent) {
    auto *main_layout = new QGridLayout;

    this->start_btn = new QPushButton("Start");

    this->output_1 = new QTextEdit;
    this->output_2 = new QTextEdit;
    this->output_3 = new QTextEdit;
    this->output_4 = new QTextEdit;

    main_layout->addWidget(this->output_1, 0, 0);
    main_layout->addWidget(this->output_2, 0, 1);
    main_layout->addWidget(this->output_3, 1, 0);
    main_layout->addWidget(this->output_4, 1, 1);
    main_layout->addWidget(this->start_btn, 2, 0);

    connect(this->start_btn, &QPushButton::released, this, &Widget::on_start);

    setLayout(main_layout);
}

```

## РЕЗУЛЬТАТИ

**Vector1:**  
dbLMZROaAf, dbLMZROaAf, dbLMZROaAf,  
dbLMZROaAf, dbLMZROaAf, dbLMZROaAf,  
dbLMZROaAf, dbLMZROaAf, dbLMZROaAf,  
dbLMZROaAf

**Vector1 sorted desc:**  
Dima, ZDAWhxgqRq, ZDAWhxgqRq, ZDAWhxgqRq,  
ZDAWhxgqRq, ZDAWhxgqRq, ZDAWhxgqRq,  
ZDAWhxgqRq, ZDAWhxgqRq, ZDAWhxgqRq,  
ZDAWhxgqRq, dbLMZROaAf, dbLMZROaAf,  
dbLMZROaAf, dbLMZROaAf

**Vector2:**  
ZDAWhxaaRa. ZDAWhxaaRa. ZDAWhxaaRa.

**Vector1 sorted asc:**  
Dima, ZDAWhxgqRq, ZDAWhxgqRq, ZDAWhxgqRq,  
ZDAWhxgqRq, ZDAWhxgqRq, ZDAWhxgqRq,  
ZDAWhxgqRq, ZDAWhxgqRq, ZDAWhxgqRq,  
ZDAWhxgqRq, dbLMZROaAf, dbLMZROaAf,  
dbLMZROaAf, dbLMZROaAf

**Map1 sorted asc:**

**Merged vector:**  
Dima, ZDAWhxgqRq, ZDAWhxgqRq, ZDAWhxgqRq,  
ZDAWhxgqRq, ZDAWhxgqRq, ZDAWhxgqRq,  
ZDAWhxgqRq, ZDAWhxgqRq, ZDAWhxgqRq,  
ZDAWhxgqRq, dbLMZROaAf, dbLMZROaAf,  
dbLMZROaAf, dbLMZROaAf

**0 elements satisfy filter**

Start

Рис. 1 Інтерфейс програми

## ВИСНОВКИ

Виконавши лабораторну роботу №11, отримав необхідні знання для реалізації алгоритмів з використанням шаблонів-контейнерів стандартної бібліотеки шаблонів. Використав отримані вміння на практиці, реалізувавши програму з використанням даних контейнерів.