ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ ИМ. ПРОФ. М.А. БОНЧ-БРУЕВИЧА»

(СПбГУТ)

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Факультет Информационных технологий и программной инженерии

Кафедра Систем обработки данных

**Дисциплина «Технологии программирования»**

**ОТЧЕТ**

**за практическое занятие №8**

**Тема: STL. Обработка данных в динамических массивах**

Выполнил

Студент 2 курса, гр. ИСТ-422

\_\_\_\_\_\_\_\_\_\_\_\_\_ Ващук А.А.

Принял

Доцент кафедры БИС

\_\_\_\_\_\_\_\_\_\_\_\_\_ Медведев В.А.

Санкт-Петербург

2025 г.

**Тема и цель практического занятия:**

Получение практических навыков разработки и анализа объектно-ориентированных программ обработки данных средствами STL.

**Задание на практическое занятие:**

*Выберите один из вариантов задания.*

**Вариант А.**

**Задание 1.** Для структуры классов, спроектированной на *Практических занятиях №5* и *№6,* сначала создать вектор из 1000 объектов какого-либо из производных классов. Затем создать второй вектор, в который занести элементы первого вектора в обратном порядке. Рассчитать *Вычисляемый показатель*.

**Задание 2.** Добавить в середину вектора 500 элементов и рассчитать *Вычисляемый показатель*.

**Задание 3.** Произвести сортировку вектора по какому-либо параметру. Очистить вектор и убедиться, что он пуст.

**Задание 4.** Разработать меню для демонстрации работы программы.

**Текст программы:**

#include "../include/Doxbase.h"

#include "../include/Passport.h"

#include "../include/StudBilet.h"

#include <iostream>

#include <vector>

#include <algorithm>

#include <cstdlib>

#include <ctime>

using namespace DoxNamespace;

using namespace PassportNamespace;

using namespace StudBiletNamespace;

std::vector<Passport> passports1;

std::vector<Passport> passports2;

int count\_documents\_by\_year(const std::vector<Passport>& vec, int year) {

    return std::count\_if(vec.begin(), vec.end(),

        [year](const Passport& p) { return p.get\_year() == year; });

}

void print\_first\_n(const std::vector<Passport>& vec, size\_t n) {

    for (size\_t i = 0; i < n && i < vec.size(); ++i) {

        vec[i].print\_info();

    }

}

int main() {

    srand(time(0));

    int choice = 0;

    int target\_year = 2024;

    while (choice != 7) {

        std::cout << "\n=== Data Processing Menu ===";

        std::cout << "\n1. Create first vector (1000 objects)";

        std::cout << "\n2. Create second vector (reverse of first)";

        std::cout << "\n3. Calculate documents for " << target\_year;

        std::cout << "\n4. Add 500 elements in the middle";

        std::cout << "\n5. Sort by issue year";

        std::cout << "\n6. Clear all vectors";

        std::cout << "\n7. Exit";

        std::cout << "\nSelect operation: ";

        std::cin >> choice;

        system("cls");

        switch (choice) {

            case 1: { // Create first vector

                passports1.clear();

                for (int i = 0; i < 1000; ++i) {

                    int year = 2020 + rand() % 5; // 2020-2024

                    char series = 'A' + rand() % 26;

                    passports1.emplace\_back(year, series);

                }

                std::cout << "Vector 1 created: " << passports1.size() << " elements\n";

                break;

            }

            case 2: { // Create reversed vector

                if (passports1.empty()) {

                    std::cout << "Error: Create first vector first!\n";

                    break;

                }

                passports2 = std::vector<Passport>(

                    passports1.rbegin(), passports1.rend());

                std::cout << "Vector 2 (reversed) created: " << passports2.size() << " elements\n";

                std::cout << "Total documents: " << Doxbase::get\_total\_count() << "\n";

                break;

            }

            case 3: { // Calculate documents

                if (passports1.empty() || passports2.empty()) {

                    std::cout << "Error: Vectors not created!\n";

                    break;

                }

                int count1 = count\_documents\_by\_year(passports1, target\_year);

                int count2 = count\_documents\_by\_year(passports2, target\_year);

                std::cout << "Vector 1: " << count1 << " documents from " << target\_year << "\n";

                std::cout << "Vector 2: " << count2 << " documents from " << target\_year << "\n";

                break;

            }

            case 4: { // Add elements

                if (passports1.empty()) {

                    std::cout << "Error: Create first vector first!\n";

                    break;

                }

                auto it = passports1.begin() + passports1.size() / 2;

                passports1.insert(it, 500, Passport(target\_year, 'X'));

                std::cout << "500 elements added. New size: " << passports1.size() << "\n";

                std::cout << "Documents from " << target\_year << ": "

                          << count\_documents\_by\_year(passports1, target\_year) << "\n";

                break;

            }

            case 5: { // Sort

                if (passports1.empty()) {

                    std::cout << "Error: Vector is empty!\n";

                    break;

                }

                std::sort(passports1.begin(), passports1.end(),

                    [](const Passport& a, const Passport& b) {

                        return a.get\_year() < b.get\_year();

                    });

                std::cout << "Sorted. First 10 elements:\n";

                print\_first\_n(passports1, 10);

                break;

            }

            case 6: { // Clear

                passports1.clear();

                passports2.clear();

                std::cout << "Vectors cleared. Sizes: "

                          << passports1.size() << " and " << passports2.size() << "\n";

                std::cout << "Total documents: " << Doxbase::get\_total\_count() << "\n";

                break;

            }

            case 7: // Exit

                return 0;

            default:

                std::cout << "Invalid selection!\n";

        }

    }

}

**Выводы по созданному проекту и использованию средств языка программирования:**

Основные компоненты программы: Doxbase – базовый класс, Passport – производный класс, StudBilet – производный класс.

Глобальные векторы:

std::vector<Passport> passports1; // Первый вектор

std::vector<Passport> passports2; // Второй вектор (реверс первого)

Вспомогательные функции:

count\_documents\_by\_year() - подсчитывает документы определенного года

print\_first\_n() - выводит первые N элементов вектора

Все данные создаются случайно и добавляются в вектор с помощью его методов (emplace\_back).

Реверсивный вектор был сделан с помощью итераторов:

passports2 = std::vector<Passport>(passports1.rbegin(), passports1.rend());

Также в программе был использован алгоритм STL (count\_if, sort). Для подсчета документов.

Было реализовано меню, для удобства пользования.