**Московский государственный технический**

**университет им. Н.Э. Баумана**

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Программирование на основе классов и шаблонов»

Отчет по лабораторной работе №8

«Стандартная библиотека шаблонов»

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы ИУ5-22Б |  | Преподаватель каф. ИУ5 |
| Поляков Л. С. |  | Бурмистрова М.В. |
| Подпись и дата: |  | Подпись и дата: |
|  |  |  |

Москва, 2024 г.

# Постановка задачи

Перед выполнением практической работы каждый студент получает индивидуальные задания. Демонстрация выполнения практической работы происходит только после выполнения всех индивидуальных заданий.

Порядок выполнения работы:

1. Проработать [примеры](https://iu5edu.ru/wiki/cpp2/docs/labs/lab8/ReferenceMaterial/), приведенные в лабораторной работе.
2. При выполнении заданий [1](https://iu5edu.ru/wiki/cpp2/docs/labs/lab8/IndividualTask/Task1/), [2](https://iu5edu.ru/wiki/cpp2/docs/labs/lab8/IndividualTask/Task2/), [3](https://iu5edu.ru/wiki/cpp2/docs/labs/lab8/IndividualTask/Task3/) написать программу с использованием строковых переменных типа string. Номер варианта - номер студента по списку в листе успеваемости.
3. При выполнении заданий [4](https://iu5edu.ru/wiki/cpp2/docs/labs/lab8/IndividualTask/Task4/) и [5](https://iu5edu.ru/wiki/cpp2/docs/labs/lab8/IndividualTask/Task5/) написать программу с использованием контейнеров vector, [6](https://iu5edu.ru/wiki/cpp2/docs/labs/lab8/IndividualTask/Task6/) и [7](https://iu5edu.ru/wiki/cpp2/docs/labs/lab8/IndividualTask/Task7/) - list, [8](https://iu5edu.ru/wiki/cpp2/docs/labs/lab8/IndividualTask/Task8/) - map . Номер варианта – остаток от деления на 10 номера студента по списку в листе успеваемости плюс 1.
4. Выполнить задание [9](https://iu5edu.ru/wiki/cpp2/docs/labs/lab8/IndividualTask/Task9/). Номер варианта – остаток от деления на 10 номера студента по списку в листе успеваемости плюс 1.

# Разработка алгоритма

## Структура проекта

.

├── CMakeLists.txt

├── Problems

│   ├── CMakeLists.txt

│   ├── Card.cpp

│   ├── Card.hpp

│   ├── Planet.cpp

│   ├── Planet.hpp

│   ├── problem\_1.cpp

│   ├── problem\_2.cpp

│   ├── problem\_3.cpp

│   ├── problem\_4.cpp

│   ├── problem\_5.cpp

│   ├── problem\_6.cpp

│   ├── problem\_7.cpp

│   ├── problem\_8.cpp

│   └── problem\_9.cpp

└── Resources

├── problem\_3.txt

└── problem\_3\_out.txt

## Задание №1

Вариант 21

### **Постановка задачи**

Пользователь вводит 3 строки - фамилия, имя и отчество учащегося. Образовать новую последовательность, оставить только фамилию и инициалы через пробел и точку.

### **Текст программы**

**problem\_1.cpp**

#include <iostream>  
  
  
std::wstring GetSurnameWithInitials(const std::wstring &lastname, const std::wstring &firstname, const std::wstring &patronymic){  
 return lastname + L" " + firstname[0] + L". " + patronymic[0] + L".";  
}  
  
int main () {  
 std::setlocale(LC\_ALL, "ru\_RU.UTF-8");  
 std::wstring lastname{}, firstname{}, patronymic{};  
 std::wcin >> lastname >> firstname >> patronymic;  
 std::wcout << GetSurnameWithInitials(lastname, firstname, patronymic);  
}

### **Анализ результатов**

Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание

## Задание №2

Вариант 21

### **Постановка задачи**

Пользователь вводит 2 слова в 2-х разных переменных одинаковой длины. Образовать новую последовательность, в которой должны чередоваться буквы первого и второго слова.

### **Текст программы**

**problem\_2.cpp**

#include <iostream>  
  
  
std::wstring NewSequence(const std::wstring &first, const std::wstring &second) {  
 if (first.length() != second.length()) {  
 return L"The words must be the same length";  
 }  
 std::wstring result{};  
 for (size\_t i{}; i < first.length(); ++i) {  
 result += first[i];  
 result += second[i];  
 }  
 return result;  
}  
  
int main() {  
 std::setlocale(LC\_ALL, "ru\_RU.UTF-8");  
 std::wstring firstWord{}, secondWord{};  
 std::wcin >> firstWord >> secondWord;  
 std::wcout << NewSequence(firstWord, secondWord);  
}

### **Анализ результатов**

Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описаниеИзображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

## Задание №3

Вариант 21

### **Постановка задачи**

Дан текстовый файл. Запишите в другой файл содержимое исходного файла, утроив восклицательный знак.

### **Текст программы**

**problem\_3.cpp**

#include <iostream>  
#include <fstream>  
#include <string>  
  
  
void DecorateText(std::ifstream &in, std::ofstream &out) {  
 std::string line{};  
 while (getline(in, line)) {  
 size\_t pos = 0;  
 while ((pos = line.find('!', pos)) != std::string::npos) {  
 line.replace(pos, 1, "!!!");  
 pos += 3;  
 }  
 out << line << std::endl;  
 }  
}  
  
  
int main() {  
 std::ifstream inputFile("../../Resources/problem\_3.txt");  
 std::ofstream outputFile("../../Resources/problem\_3\_out.txt");  
  
 if (!inputFile.is\_open()) {  
 std::cout << "Error opening file by path: ../../Resources/problem\_3.txt" << std::endl;  
 return 1;  
 }  
 if (!outputFile.is\_open()) {  
 std::cout << "Error opening file by path: ../../Resources/problem\_3\_out.txt" << std::endl;  
 return 1;  
 }  
  
 DecorateText(inputFile, outputFile);  
  
 inputFile.close();  
 outputFile.close();  
  
 std::cout << "Done" << std::endl;  
 return 0;  
}

### **Анализ результатов**

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описаниеИзображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

## Задание №4

Вариант 2

### **Постановка задачи**

Создайте три объекта vector<float> и заполните первые два объекта так, как в предыдущем примере. Напишите цикл for, который суммирует соответствующие элементы первых двух векторов и заносит результат в соответствующий элемент третьего вектора. Выведите содержимое всех трех векторов. Переделать программу: печатать векторы с помощью функции.

### **Текст программы**

**problem\_4.cpp**

#include <iostream>  
#include <vector>  
  
  
std::vector<double> SumVectors(const std::vector<double> &vec1, const std::vector<double> &vec2) {  
 if (vec1.size() != vec2.size()) {  
 return std::vector<double>(0);  
 }  
 std::vector<double> resVec(vec1.size());  
 for (size\_t i{}; i < vec1.size(); ++i) {  
 resVec[i] = vec1[i] + vec2[i];  
 }  
 return resVec;  
}  
  
void PrintVec(const std::vector<double> &v) {  
 if (v.empty()) {  
 std::cout << "Vector is empty" << std::endl;  
 return;  
 }  
 std::cout << "Vector:" << std::endl;  
 size\_t i{};  
 std::cout << v[i++];  
 for (; i < v.size(); ++i) {  
 std::cout << " " << v[i];  
 }  
 std::cout << std::endl;  
}  
  
void InitVector(std::vector<double> &v) {  
 for (size\_t i{}; i < v.size(); ++i) {  
 v[i] = 0.1 \* static\_cast<double>(i) \* (static\_cast<int>(i % 2) \* -1);  
 }  
}  
  
int main(){  
 std::vector<double> vec1(25), vec2(25);  
 InitVector(vec1);  
 InitVector(vec2);  
 PrintVec(vec1);  
 PrintVec(vec2);  
 PrintVec(SumVectors(vec1, vec2));  
 return 0;  
}

### **Анализ результатов**

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

## Задание №5

Вариант 2

### **Постановка задачи**

Создать класс с именем Planet для хранения следующей информации:

* название (string)
* количество спутников.

Создать массив планет с помощью контейнера vector в main().

Занести в вектор информацию о 9 планетах. Данные о планетах:

"Меркурий", 0

"Венера", 0

"Земля", 1

"Марс", 2

"Юпитер", 69

"Сатурн", 62

"Уран", 27

"Нептун", 14

"Плутон", 10

Вывести массив планет на экран, используя функцию.

В цикле определите элемент, у которого максимальное количество спутников.

### **Текст программы**

**problem\_5.cpp**

#include <iostream>  
#include <vector>  
#include <iomanip>  
#include "Planet.hpp"  
  
  
void PrintPlanets(const std::vector<Planet> &planets) {  
 std::pair<std::string, size\_t> maxSatelliteCount{};  
 for (const auto &planet : planets) {  
 if (maxSatelliteCount.second < planet.getSatelliteCount()) {  
 maxSatelliteCount.first = planet.getName();  
 maxSatelliteCount.second = planet.getSatelliteCount();  
 }  
 std::cout << std::quoted(planet.getName()) << " " << planet.getSatelliteCount() << std::endl;  
 }  
 std::cout << std::endl << "Больше всего спутников: " << maxSatelliteCount.first << " - " << maxSatelliteCount.second << std::endl;  
}  
  
  
int main() {  
 std::vector<Planet> planets(9);  
 planets[0].setName("Меркурий");  
 planets[0].setSatelliteCount();  
 planets[1].setName("Венера");  
 planets[1].setSatelliteCount();  
 planets[2].setName("Земля");  
 planets[2].setSatelliteCount(1);  
 planets[3].setName("Марс");  
 planets[3].setSatelliteCount(2);  
 planets[4].setName("Юпитер");  
 planets[4].setSatelliteCount(69);  
 planets[5].setName("Сатурн");  
 planets[5].setSatelliteCount(62);  
 planets[6].setName("Уран");  
 planets[6].setSatelliteCount(27);  
 planets[7].setName("Нептун");  
 planets[7].setSatelliteCount(14);  
 planets[8].setName("Плутон");  
 planets[8].setSatelliteCount(10);  
 PrintPlanets(planets);  
}

**Planet.hpp**

#ifndef LAB\_8\_PROBLEM\_5\_HPP  
#define LAB\_8\_PROBLEM\_5\_HPP  
  
  
#include <iostream>  
  
class Planet {  
private:  
 std::string name{"DEFAULT\_PLANET"};  
 size\_t satelliteCount{};  
public:  
 Planet();  
 Planet(std::string \_name, size\_t \_satelliteCount);  
 [[nodiscard]] std::string getName() const;  
 [[nodiscard]] size\_t getSatelliteCount() const;  
 void setName(const std::string &newName);  
 void setSatelliteCount(size\_t newSatelliteCount=0);  
};  
  
  
#endif //LAB\_8\_PROBLEM\_5\_HPP

**Planet.cpp**

#include "Planet.hpp"  
  
#include <utility>  
  
Planet::Planet() = default;  
  
Planet::Planet(std::string \_name, const size\_t \_satelliteCount) :  
 name(std::move(\_name)), satelliteCount(\_satelliteCount)  
 {  
  
 }  
  
std::string Planet::getName() const {  
 return name;  
}  
  
size\_t Planet::getSatelliteCount() const {  
 return satelliteCount;  
}  
  
void Planet::setName(const std::string &newName) {  
 this->name = newName;  
}  
  
void Planet::setSatelliteCount(const size\_t newSatelliteCount) {  
 this->satelliteCount = newSatelliteCount;  
}

### **Анализ результатов**

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

## Задание №6

Вариант 2

### **Постановка задачи**

Описать функцию, которая считает количество положительных элементов заданного контейнера-списка list<int>, а затем распечатывает это значение (выдает в стандартный поток cout). Продемонстрировать её работу.

### **Текст программы**

**problem\_6.cpp**

#include <iostream>  
#include <list>

void CountPositive(const std::list<int> &lst) {  
 auto positiveCount{std::count\_if(lst.cbegin(), lst.cend(), [](int n) -> bool {return n > 0;})};  
 std::cout << "Total positive numbers: " << positiveCount << std::endl;  
}  
  
int main() {  
 std::list<int> integers{1, -12, 54, -42, 228, 1337, -14, 2000, 98, 5};  
 CountPositive(integers);  
}

### **Анализ результатов**

Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание

## Задание №7

Вариант 2

### **Постановка задачи**

Создать класс:

class Card {

string title; // заглавие книги

string author; // автор

int number; // количество имеющихся экземпляров

public:

Card() : title(""), author(""), number(0) {}

Card(string t, string a, int n) : title(t), author(a), number(0) {}

};

В программе создать list из объектов класса Card (5 элементов).

Распечатать список. Пользователь вводит автора, программа выводит книги автора или сообщение об их отсутствии.

### **Текст программы**

**problem\_7.cpp**

#include <iostream>  
#include <list>  
#include "Card.hpp"  
  
  
void PrintCards(const std::list<Card> &cards) {  
 for (const auto &card : cards) {  
 card.print();  
 }  
}  
  
void PrintCardsByAuthor(const std::list<Card> &cards, const std::string& author) {  
 bool found{};  
 for (const auto &card : cards) {  
 if (card.getAuthor() == author) {  
 found = true;  
 card.print();  
 }  
 }  
 if (!found) {  
 std::cout << "Книг от данного автора не найдено" << std::endl;  
 }  
}  
  
int main() {  
 std::list<Card> cards{};  
 cards.emplace\_back("Война и мир", "Толстой", 1);  
 cards.emplace\_back("Drei Cameraden", "Remarque", 2);  
 cards.emplace\_back("Мастер и Маргарита", "Булгаков", 3);  
 cards.emplace\_back("Герой нашего времени", "Лермонтов", 4);  
 cards.emplace\_back("Повести Белкина", "Пушкин", 5);  
 PrintCards(cards);  
 std::string authorToFind{};  
 for(;;) {  
 std::cout << "\nВведите автора для поиска: ";  
 std::cin >> authorToFind;  
 if (authorToFind == "exit") {  
 break;  
 }  
 PrintCardsByAuthor(cards, authorToFind);  
 }  
}

**Card.hpp**

#ifndef LAB\_8\_CARD\_HPP  
#define LAB\_8\_CARD\_HPP  
  
  
#include <iostream>  
  
class Card {  
private:  
 std::string title;  
 std::string author;  
 int number{};  
public:  
 Card();  
 Card(std::string, std::string, int);  
 void print() const;  
 [[nodiscard]] std::string getAuthor() const ;  
};  
  
  
#endif //LAB\_8\_CARD\_HPP

**Card.cpp**

#include "Card.hpp"  
  
Card::Card() = default;  
  
Card::Card(std::string \_title, std::string \_author, const int \_number) :  
 title(std::move(\_title)), author(std::move(\_author)), number(\_number)  
 {  
  
 }  
  
void Card::print() const {  
 std::cout << "Название: " << title << std::endl;  
 std::cout << "Автор: " << author << std::endl;  
 std::cout << "Количество экземпляров: " << number << std::endl;  
 std::cout << "--------------------------" << std::endl;  
}  
  
std::string Card::getAuthor() const {  
 return author;  
}

### **Анализ результатов**

Изображение выглядит как текст, снимок экрана, Шрифт, меню

Автоматически созданное описание

## Задание №8

Вариант 2

### **Постановка задачи**

Создайте отображение map<int, double> и занесите в него пары 1 - 1.1, 2 - 2.2, 3 - 3.3, 4 - 4.4, 5 - 5.5, ..., 9 - 9.9. Выведите содержимое отображения на экран. Удалить пары с четными ключами и снова распечатайте отображение.

### **Текст программы**

**problem\_8.cpp**

#include <iostream>  
#include <map>  
  
  
void PrintMap(const std::map<int, double> &map) {  
 for (const auto& [integer, decimal] : map) {  
 std::cout << integer << ": " << decimal << std::endl;  
 }  
 std::cout << std::endl;  
}  
  
void DeleteEven(std::map<int, double> &map) {  
 for (auto& [key, value] : map) {  
 if (!(key % 2)) {  
 map.erase(key);  
 }  
 }  
}  
  
int main() {  
 std::map<int, double> map {  
 {1, 1.1},  
 {2, 2.2},  
 {3, 3.3},  
 {4, 4.4},  
 {5, 5.5},  
 {6, 6.6},  
 {7, 7.7},  
 {8, 8.8},  
 {9, 9.9},  
 };  
 PrintMap(map);  
 DeleteEven(map);  
 PrintMap(map);  
}

### **Анализ результатов**

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Автоматически созданное описание

## Задание №9

Вариант 2

### **Постановка задачи**

В отображении находятся записи о соответствии номеров телефона (7 цифр) и фамилии. Например:

5671234 Ivanov  
3214567 Petrov  
9871234 Sidorov

В массиве находятся номера телефонов. Например:

string mas[4] = { "1112233", "9871234", "5671234", "5556688" };

Осуществить поиск фамилий из заданного отображения, соответствующих телефонам из массива mas. Найденные фамилии вывести на консоль. В данном примере:

Sidorov  
Ivanov

### **Текст программы**

**problem\_9.cpp**

#include <iostream>  
#include <map>  
  
  
void FindCompliances(std::map<std::string, std::string> &phoneBook, std::string \*phoneNumbers, const size\_t length) {  
 for(size\_t i{}; i < length; ++i) {  
 if (phoneBook.count(\*(phoneNumbers + i)) > 0) {  
 std::cout << phoneBook[\*(phoneNumbers + i)] << std::endl;  
 }  
 }  
}  
  
  
int main() {  
 constexpr size\_t arrSize{4};  
 std::map<std::string, std::string> phoneBook{  
 {"5671234", "Ivanov"},  
 {"3214567", "Petrov"},  
 {"9871234", "Sidorov"}  
 };  
 std::string phoneNumbers[arrSize] = {"1112233", "9871234", "5671234", "5556688"};  
  
 FindCompliances(phoneBook, phoneNumbers, arrSize);  
}

### **Анализ результатов**

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

# Вывод

Я научился

* Работать со стандартной библиотекой шаблонов STL
* Использовать структуры данных из STL такими как vector, list и map
* Работать со статическими библиотеками с использованием Cmake