Московский государственный технический

университет им. Н.Э. Баумана

Проверил:

Преподаватель каф. ИУ5

Подпись и дата:

Факультет «Информатика и система управления»

Кафедра ИУ5 «Система обработки информации и управления»

Выполнил:

Студент группы ИУ5-22Б

Яншин Т. И.

Подпись и дата:

Курс «Программирование на основе классов и шаблонов»

Отчёт по лабораторной работе №8

Стандартная библиотека шаблонов

## **Цель работы**

## Целью лабораторной работы изучение возможностей стандартной библиотеки шаблонов.

## **Задание**

**Вариант 10**

**При выполнении заданий 1, 2, 3 написать программу с использованием строковых переменных типа string**

1. Пользователь вводит два текста (две переменные). В каком из них больше слов? При условии, что слова разделяются только одним пробелом (сначала найти количество пробелов в каждом тексте).

2. Пользователь вводит некоторый текст. Образовать новую последовательность, включив в нее символы исходной, кроме точек.

3. Дан текстовый файл. Запишите в другой файл содержимое исходного файла, кроме латинских букв. Оформить обработку строки в виде функции. Произвести обработку каждой строки исходного файла, используя полученную функцию. Полученный результат вывести на экран.

**При выполнении заданий 4 и 5 написать программу с использованием контейнеров vector, 6 и 7 - list, 8 - map .**

4. Создайте два вектора для хранения имен абонентов и их телефонных номеров. Имена и номера телефонов должны вводиться пользователем. После окончания ввода распечатать имена и телефонные номера абонентов в виде строк: имя абонента – его номер. Выполнить поиск номера по имени абонента. Удалите найденный номер и имя абонента из векторов. Снова распечатайте векторы. Переделать программу: печатать векторы с помощью функции.

5. Ниже представлен пример класса Graduate (выпускник). Напишите программу для хранения объектов типа Graduate в векторе (подсказка: не забудьте для класса Graduate определить операторы < и ==):

class Graduate {

string name; // фамилия

double rating; // рейтинг

public:

Graduate() {

name = "";

rating = 0;

}

Graduate(string Name, double Rating) {

name = Name;

rating = Rating;

}

};

Создать массив объектов с помощью контейнера vector в main() и сразу занести в него информацию о 6 объектах: ("Иванов", 99), ("Петров", 430), ("Семенов", 70), ("Котов", 540), ("Белых", 230), ("Черных", 350). Вывести массив на экран таким образом, чтобы характеристики каждого объекта выводились в отдельной строке. Выведите на экран характеристики объекта, который имеет максимальный рейтинг. Увеличьте рейтинг каждого выпускника на число, которое введет пользователь и снова выведите массив на экран. Напишите функцию вывода массива на экран.

6. Создайте два контейнера для хранения имен абонентов и их телефонных номеров. Имена и номера телефонов должны вводиться пользователем. После окончания ввода распечатать имена и телефонные номера абонентов в виде строк: имя абонента – его номер. Выполнить поиск номера по имени абонента. Удалите найденный номер и имя абонента из контейнеров. Снова распечатайте контейнеры. Переделать программу: печатать контейнеры с помощью функции.

7. Ниже представлен пример класса Point. Напишите программу для хранения объектов типа Point в списке (подсказка: не забудьте для класса Point определить операторы < и ==)

class Point {

public:

double x, у;

Point() { x = у = 0; }

Point(double a, double b) {

x = а;

у = b;

}

};

Создать массив точек с помощью контейнера list в main() и сразу занести в него информацию о 6 точках: (1.2, 6.3), (4.0, 0.7), (7.2, 0.8), (5.3, 3.0), (4.9, 6.6), (9.3, 0.2). Вывести список на экран таким образом, чтобы координаты каждой точки выводились в отдельной строке. Выведите на экран координаты точки, которая наиболее удалена от центра координат. Сдвиньте все точки влево по оси абсцисс на расстояние, которое введет пользователь и снова выведите список на экран. Напишите функцию вывода списка на экран.

8. Создайте отображение map<int, double> и занесите в него пары 1 – 100.0, 2 – 200.0, 3 – 300.0, ..., 9 – 900.0. Выведите содержимое отображения на экран. Вычислите сумму значений в контейнере. Каждое значение разделите на полученную сумму и вновь выведите отображение.

9. Дано отображение с данными о количестве учащихся в разных классах школы:

{'1а': 21, '1б':20, '2':25, '3а': 20, '3б': 23, '4а':25, '4б':25, '5а': 26, '5б':25, '6a':19, '6б': 18, '7а':26, '7б':25, '8':25, '9':17, '10': 15, '11’: 14}  
В программе пользователь вводит год обучения школьников, а консоль выводится количество учащихся заданного года обучения, например:

Год обучения: 1

Количество: 41

Распечатать содержимое отображения в консоли, используя функцию.

## **Текст программы**

Внешний CMakeLists.txt

cmake\_minimum\_required(VERSION 3.8)

set(CMAKE\_CXX\_STANDARD 20)

project(lab8)

set(SOURCE\_FILES

main.cpp

Point.hpp

Graduate.hpp

LabWork2\_No8.h

)

add\_executable(lab8 **${SOURCE\_FILES}**)

Main.cpp

#include "LabWork2\_No8.h"

#include "Graduate.hpp"

#include "Point.hpp"

static int countWords(const std::string& text) {

int spaces = 0;

for (char c : text) {

if (c == ' ') {

spaces++;

}

}

return spaces + 1;

}

static std::string removePoints(const std::string& text) {

std::string result;

for (char c : text) {

if (c != '.') {

result += c;

}

}

return result;

}

static std::string removeLatinLetters(const std::string& text) {

std::string result;

for (char c : text) {

if (!(c >= 'a' && c <= 'z') && !(c >= 'A' && c <= 'Z')) {

result += c;

}

}

return result;

}

static void printVector(const std::vector<std::string>& names, const std::vector<std::string>& numbers) {

for (size\_t i = 0; i < names.size(); ++i) {

std::cout << names[i] << " - " << numbers[i] << std::endl;

}

}

static void printList(const std::list<std::string>& names, const std::list<std::string>& numbers) {

auto itNames = names.begin();

auto itNumbers = numbers.begin();

for (; itNames != names.end() && itNumbers != numbers.end(); ++itNames, ++itNumbers) {

std::cout << \*itNames << " - " << \*itNumbers << std::endl;

}

}

void printMap(const std::map<std::string, int>& students) {

for (const auto& pair : students) {

std::cout << pair.first << " - " << pair.second << std::endl;

}

}

int main() {

setlocale(LC\_ALL, "Rus");

/\* Задание 1 \*/

std::string text1, text2;

std::cout << " === Задание 1 === \n";

std::cout << "Enter the first text: ";

std::getline(std::cin, text1);

std::cout << "Enter the second text: ";

std::getline(std::cin, text2);

int wordsText1 = countWords(text1);

int wordsText2 = countWords(text2);

if (wordsText1 > wordsText2) {

std::cout << "The first text has more words.\n";

}

else if (wordsText1 < wordsText2) {

std::cout << "The second text has more words.\n";

}

else {

std::cout << "Both texts have the same number of words.\n";

}

/\* Задание 2 \*/

std::cout << "\n === Задание 2 === \n";

std::string text;

std::cout << "Enter text with points: ";

std::getline(std::cin, text);

std::string newText = removePoints(text);

std::cout << "Text without points: " << newText << std::endl;

/\* Задание 3 \*/

std::cout << "\n === Задание 3 === \n см. файл output.txt \n";

std::ifstream inputFile("input.txt");

std::ofstream outputFile("output.txt");

std::string line;

while (std::getline(inputFile, line)) {

std::string newLine = removeLatinLetters(line);

outputFile << newLine << std::endl;

}

inputFile.close();

outputFile.close();

/\* Задание 4 \*/

std::cout << "\n === Задание 4 === \n";

std::vector<std::string> names;

std::vector<std::string> numbers;

std::string name, number;

while (true) {

std::cout << "For exit enter 'end'\n";

std::cout << "Enter name: ";

std::cin >> name;

if (name == "end") break;

std::cout << "Enter phone number: ";

std::cin >> number;

names.push\_back(name);

numbers.push\_back(number);

}

if (!names.empty()) {

printVector(names, numbers);

std::cout << "Enter name to search: ";

std::cin >> name;

auto it = std::find(names.begin(), names.end(), name);

if (it != names.end()) {

size\_t index = std::distance(names.begin(), it);

names.erase(names.begin() + index);

numbers.erase(numbers.begin() + index);

}

std::cout << name << " deleted.\n";

printVector(names, numbers);

}

else {

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

}

/\* Задание 5 \*/

std::cout << "\n === Задание 5 === \n";

std::vector<Graduate> graduates = {

Graduate("Иванов", 99),

Graduate("Петров", 430),

Graduate("Семенов", 70),

Graduate("Котов", 540),

Graduate("Белых", 230),

Graduate("Черных", 350)

};

printGraduates(graduates);

auto maxIter = std::max\_element(graduates.begin(), graduates.end());

std::cout << "Graduate with highest rating: " << maxIter->getName() << " - " << maxIter->getRating() << std::endl;

double increment;

std::cout << "Enter increment value: ";

std::cin >> increment;

for (auto& grad : graduates) {

grad.increaseRating(increment);

}

printGraduates(graduates);

/\* Задание 6\*/

std::cout << "\n === Задание 6 === \n";

std::list<std::string> names1;

std::list<std::string> numbers1;

std::string name1, number1;

while (true) {

std::cout << "Enter name (or 'end' to finish): ";

std::cin >> name1;

if (name1 == "end") break;

std::cout << "Enter phone number: ";

std::cin >> number1;

names1.push\_back(name1);

numbers1.push\_back(number1);

}

printList(names1, numbers1);

if (!names1.empty()){

std::cout << "Enter name to search: ";

std::cin >> name1;

auto it = std::find(names1.begin(), names1.end(), name1);

if (it != names1.end()) {

auto numIt = numbers1.begin();

std::advance(numIt, std::distance(names1.begin(), it));

names1.erase(it);

numbers1.erase(numIt);

}

std::cout << name1 << " deleted.\n";

printList(names1, numbers1);

}

else {

std::cout << "List is empty\n";

}

/\* Задание 7 \*/

std::cout << "\n === Задание 7 === \n";

std::list<Point> points = {

{1.2, 6.3},

{4.0, 0.7},

{7.2, 0.8},

{5.3, 3.0},

{4.9, 6.6},

{9.3, 0.2}

};

std::cout << "Initial list of points:\n";

printPoints(points);

auto maxIter1 = std::max\_element(points.begin(), points.end());

if (maxIter1 != points.end()) {

std::cout << "Point farthest from origin: (" << maxIter1->x << ", " << maxIter1->y << ")\n";

}

else {

std::cerr << "Error finding the farthest point.\n";

}

double shift;

std::cout << "Enter shift value: ";

std::cin >> shift;

for (auto& point : points) {

point.x -= shift;

}

std::cout << "List after shifting:\n";

printPoints(points);

/\* Задание 8 \*/

std::cout << "\n === Задание 8 === \n";

std::map<int, double> data;

for (int i = 1; i <= 9; ++i) {

data[i] = i \* 100.0;

}

for (const auto& pair : data) {

std::cout << pair.first << " - " << pair.second << std::endl;

}

double sum = 0;

for (const auto& pair : data) {

sum += pair.second;

}

std::cout << "Sum of values: " << sum << std::endl;

for (auto& pair : data) {

pair.second /= sum;

}

for (const auto& pair : data) {

std::cout << pair.first << " - " << pair.second << std::endl;

}

/\* Задание 9 \*/

std::cout << "\n === Задание 9 === \n";

std::map<std::string, int> students = {

{"1а", 21}, {"1б", 20}, {"2", 25},

{"3а", 20}, {"3б", 23}, {"4а", 25},

{"4б", 25}, {"5а", 26}, {"5б", 25},

{"6a", 19}, {"6б", 18}, {"7а", 26},

{"7б", 25}, {"8", 25}, {"9", 17},

{"10", 15}, {"11", 14}

};

std::cout << "Enter the year of study: ";

int year;

std::cin >> year;

int totalStudents = 0;

for (const auto& pair : students) {

if (stoi(pair.first.substr(0, 2)) == year) {

totalStudents += pair.second;

}

}

std::cout << "Year of study: " << year << std::endl;

std::cout << "Number of students: " << totalStudents << std::endl;

printMap(students);

return 0;

}

Graduate.hpp

#include "LabWork2\_No8.h"

class Graduate {

std::string name;

double rating;

public:

Graduate() : name(""), rating(0) {}

Graduate(std::string Name, double Rating) : name(Name), rating(Rating) {}

double getRating() const { return rating; }

std::string getName() const { return name; }

void increaseRating(double value) {

rating += value;

}

bool operator<(const Graduate& other) const {

return rating < other.rating;

}

bool operator==(const Graduate& other) const {

return rating == other.rating;

}

};

void printGraduates(const std::vector<Graduate>& graduates) {

for (const auto& grad : graduates) {

std::cout << grad.getName() << " - " << grad.getRating() << std::endl;

}

}

Point.hpp

#include "LabWork2\_No8.h"

class Point {

public:

double x, y;

Point() : x(0), y(0) {}

Point(double a, double b) : x(a), y(b) {}

double distance() const {

return sqrt(x \* x + y \* y);

}

bool operator<(const Point& other) const {

return this->distance() < other.distance();

}

bool operator==(const Point& other) const {

return this->x == other.x && this->y == other.y;

}

};

void printPoints(const std::list<Point>& points) {

for (const auto& point : points) {

std::cout << "(" << point.x << ", " << point.y << ")" << std::endl;

}

}

LabWork2\_No8.h

#pragma once

#include <iostream>

#include <string>

#include <fstream>

#include <vector>

#include <algorithm>

#include <list>

#include <map>

**Тестирование**

|  |  |
| --- | --- |
|  |  |

|  |  |
| --- | --- |
|  |  |

|  |
| --- |
| Отдельно. Тест для задания 3 с файлом |
|  |
|  |