**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 ИУ5-23Б |  | преподаватель каф. ИУ5 |
| Лачина Екатерина Андреевна |  | Аладин Д.В. |
| Подпись и дата: |  | Подпись и дата: |

Москва, 2024 г.

**1) Задача**

1. Пользователь вводит два слова (две переменные). Сколько раз в первом слове встречается последняя буква второго слова.
2. Пользователь вводит некоторый текст. Образовать из него новый, в который включить информацию, заключенную между пробелом и запятой.
3. Дан текстовый файл. Запишите в другой файл содержимое исходного файла, нумеруя каждую строку.
4. Создайте вектор vector<float> и занесите в него 2525 вещественных чисел. Возведите каждое число в квадрат и сохраните результат в исходном элементе вектора. Выведите содержимое вектора до и после возведения в квадрат. Удалить из вектора элемент с индексом 5 и снова распечатать вектор. Переделать программу: печатать вектор с помощью функции.
5. 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 вектор из объектов класса Card (5 элементов).

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

1. Описать функцию, которая печатает Yes или No в зависимости от того, содержится ли заданное целое число x в заданном контейнере-списке list<int>.
2. Составить описание класса Complex для представления комплексных чисел с возможностью задания вещественной и мнимой частей числами типа double. В программе создать список из объектов класса Complex (6 элементов): (-1.2, 6.3), (4.0, 0.7), (7.2, -0.8), (5.3, 3.0), (-4.9, 6.6), (-9.3, 0.2). Распечатать список в виде:

-1.2 + i \* 6.3  
4.1 + i \* 0.7  
7.2 - i \* 0.8  
5.3 + i \* 3  
-4.9 + i \* 6.6  
-9.3 + i \* 0.2

Сложите все числа (у комплексных чисел отдельно складываются действительные и мнимые части) и результирующее число выведите на экран.

1. Создайте отображение map<int, int> и занесите в него пары 1 - 10, 2 - 20, 3 - 30, ..., 20 - 200. Выведите содержимое отображения на экран. Удалить пары с нечетными ключами и снова распечатайте отображение.
2. В первом отображении находится информация о зарплате сотрудников. Например:

Ivanov 45500  
Petrov 37000  
Sidorov 245000  
Petrenko 65000  
Tovalds 91200  
Popov 54600  
Andrienko 35000

Во втором отображении принадлежность сотрудников к отделам предприятия. Например:

Ivanov Research  
Petrov Research  
Sidorov Management  
Petrenko Management  
Tovalds Development  
Popov Research  
Andrienko Sales

Необходимо вычислить общий фонд зарплаты и отдельно по отделам. Результат вывести на экран. В данном случае (возможны опечатки):

Research 137100  
Management 310000  
Development 91200  
Sales 35000  
Overall 573300

Примечание: Для решения этой задачи необходимо завести ассоциативный массив (std::map), где ключом будет название отдела (строка), а значением – текущая сумма зарплат. В цикле перебирать пары первого отображения, необходимо находить по ключу отдел во втором отображении и прибавлять к имеющейся там (в ассоциативном массиве) сумме только что считанную.

**2) Листинг**

**Папка LabaEight**

**CMakeLists.txt**

cmake\_minimum\_required(VERSION 3.23)

set(project "LabaEight")

project(**${project}**)

set(CMAKE\_CXX\_STANDARD 17)

set(**${project}**\_SOURCES

LabaEight.cpp )

set(**${project}**\_HEADERS

LabaEight.h )

set(**${project}**\_SOURCE\_LIST

${**${project}**\_SOURCES}

${**${project}**\_HEADERS})

add\_library(**${project}**

STATIC

${**${project}**\_SOURCE\_LIST})

**LabaEight.cpp**

#include "LabaEight.h"

void printVector(std::vector<float>& vec) {

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

std::cout << vec[i] << " ";

}

std::cout << std::endl;

}

void printList(std::list<int>& cont, int x) {

std::list<int>::iterator iter;

bool contains = false;

if (!cont.empty()) {

for (iter = cont.begin(); iter != cont.end(); ++iter) {

if (\*iter == x) {

contains = true;

break;

}

}

std::cout << (contains ? "Yes" : "No") << std::endl;

}

else {

std::cout << "список пуст " << std::endl;

}

}

**LabaEight.h**

#include <iostream>

#include <string.h>

#include <string>

#include <fstream>

#include <vector>

#include <algorithm>

#include <list>

#include <map>

void printVector(std::vector<float>& vec);

void printList(std::list<int>& cont, int x);

class Card {

std::string title; // заглавие книги

std::string author; // автор

int number; // количество имеющихся экземпляров

public:

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

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

std::string get\_author(){

return author;

}

void print(){

std::cout << "книга: " << title << " автор: " << author << " количество: " << number << std::endl;

}

};

class Complex {

double real;

double imag;

public:

Complex(double r, double i) : real(r), imag(i) {}

void print() {

std::cout << real;

if (imag >= 0) {

std::cout << " + i \* " << imag;

}

else {

std::cout << " - i \* " << -imag;

}

std::cout << std::endl;

}

Complex operator+(Complex& other) {

return Complex(real + other.real, imag + other.imag);

}

};

**Главные файлы**

**CMakeLists.txt**

cmake\_minimum\_required(VERSION 3.23)

set(project "LAB8")

project(**${project}**)

set(CMAKE\_CXX\_STANDARD 17)

configure\_file(**${CMAKE\_CURRENT\_SOURCE\_DIR}**/input.txt

**${CMAKE\_CURRENT\_BINARY\_DIR}** COPYONLY)

configure\_file(**${CMAKE\_CURRENT\_SOURCE\_DIR}**/output.txt

**${CMAKE\_CURRENT\_BINARY\_DIR}** COPYONLY)

set(**${project}**\_SOURCES

main.cpp)

add\_subdirectory(LabaEight)

set(**${project}**\_SOURCE\_LIST

${**${project}**\_SOURCES})

add\_executable(**${project}**

${**${project}**\_SOURCE\_LIST})

target\_link\_libraries(**${project}** LabaEight)

**main.cpp**

#include "LabaEight/LabaEight.h"

int main() {

//setlocale(LC\_ALL, "Russian");

system("chcp 1251");

int x;

while (1) {

std::cout << "введите номер задания: ";

std::cin >> x;

std::cin.ignore(); // Очищаем входной буфер от символа новой строки

switch (x) {

case 1: {

// Task 1

//Пользователь вводит два слова(две переменные).

//Сколько раз в первом слове встречается последняя буква второго слова.

std::string word1, word2;

std::cout << "Введите два слова: ";

std::cin >> word1 >> word2;

char last\_letter = word2[word2.size() - 1];

int count = 0;

for (int i = 0; i < word1.length(); i++) {

if (word1[i] == last\_letter) {

count++;

}

}

std::cout << "в первом слове встречается последняя буква второго слова " << count << " раз(а)." << std::endl;

break;

}

case 2: {

//Task 2

//Пользователь вводит некоторый текст.

//Образовать из него новый, в который включить информацию, заключенную между пробелом и запятой.

std::string text;

std::cout << "Введите текст: ";

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

std::string result;

bool inside = false;

for (int i = 0; i < text.length(); i++) {

if (text[i] == ' ' && !inside) {

inside = true;

}

else if (text[i] == ',' && inside) {

inside = false;

}

else if (inside) {

result += text[i];

}

}

std::cout << "текст между пробелом и запятой: " << result << std::endl;

break;

}

case 3: {

//Task 3

// Дан текстовый файл. Запишите в другой файл содержимое исходного файла, нумеруя каждую строку

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

if (!inputfile.is\_open()) {

std::cout << "ошибка открытия input.txt" << std::endl;

return 1;

}

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

if (!outputfile.is\_open()) {

std::cout << "ошибка открытия output.txt" << std::endl;

return 1;

}

std::string line;

int col = 1;

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

outputfile << col << " : " << line << std::endl;

std::cout << col << " : " << line << std::endl;

col++;

}

inputfile.close();

outputfile.close();

std::cout << "нумерация строк в output.txt." << std::endl;

break;

}

case 4: {

//Task 4

std::vector<float> arr;

for (int i = 0; i < 25; ++i) {

arr.push\_back(i);

}

std::cout << "до возведения в квадрат:" << std::endl;

printVector(arr);

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

arr[i] \*= arr[i];

}

std::cout << "после возведения в квадрат:" << std::endl;

printVector(arr);

if (arr.size() > 5) {

arr.erase(arr.begin() + 5);

}

std::cout << std::endl << "после удаления элемента с индексом 5:" << std::endl;

printVector(arr);

break;

}

case 5: {

//Task 5

std::vector<Card> library;

library.push\_back(Card("Война и мир", "Толстой", 3));

library.push\_back(Card("преступление и наказание", "достоевский", 5));

library.push\_back(Card("1984", "оруэлл", 2));

library.push\_back(Card("мастер и маргарита", "булгаков", 4));

library.push\_back(Card("инженерная графика", "горячкина", 1));

std::cout << "библиотека:" << std::endl;

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

library[i].print();

}

std::string input\_au;

std::cout << "введите имя автора: ";

std::getline(std::cin, input\_au);

bool found = false;

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

if (library[i].get\_author() == input\_au) {

library[i].print();

found = true;

}

}

if (!found) {

std::cout << "у меня нет такой книги" << std::endl;

}

break;

}

case 6: {

//Task 6

// Описать функцию, которая печатает Yes или No в зависимости от того,

// содержится ли заданное целое число x в заданном контейнере-списке list<int>.

std::list<int> my\_list = { 1, 2, 3, 4, 5 };

int x;

std::cout << "введите целое число: ";

std::cin >> x;

printList(my\_list, x);

break;

}

case 7: {

//Task 7

//вычислить сумму всех комплексных чисел

std::list<Complex> complex\_arr = {Complex(-1.2, 6.3), Complex(4.0, 0.7), Complex(7.2, -0.8),

Complex(5.3, 3.0), Complex(-4.9, 6.6), Complex(-9.3, 0.2)

};

std::cout << "мой cписок компл чисел:" << std::endl;

std::list<Complex>::iterator iter;

for (iter = complex\_arr.begin(); iter != complex\_arr.end(); ++iter) {

iter -> print();

}

Complex sum(0, 0);

std::list<Complex>::iterator iter2;

for (iter2 = complex\_arr.begin(); iter2 != complex\_arr.end(); ++iter2) {

sum = sum + \*iter2;

}

std::cout << "Сумма всех комплексных чисел: ";

sum.print();

break;

}

case 8: {

//Task 8

//Создайте отображение map<int, int> и занесите в него пары 1 - 10, 2 - 20, 3 - 30, ...,

//20 - 200.

//Выведите содержимое отображения на экран.Удалить пары с нечетными ключами и

//снова распечатайте отображение.

std::map<int, int> my\_map;

for (int i = 1; i <= 20; ++i) {

my\_map[i] = i \* 10;

}

std::cout << "до удаления пар с нечетными ключами:" << std::endl;

std::map<int, int>::iterator it;

for (it = my\_map.begin(); it != my\_map.end(); it++) {

std::cout << it->first << " - " << it->second << std::endl;

}

for (it = my\_map.begin(); it != my\_map.end(); it++) {

if (it->first % 2 != 0) {

it = my\_map.erase(it);

}

}

// Вывод содержимого отображения после удаления пар с нечетными ключами

std::cout << "содержимое отображения после удаления пар с нечетными ключами:" << std::endl;

for (it = my\_map.begin(); it != my\_map.end(); it++) {

std::cout << it->first << " - " << it->second << std::endl;

}

break;

}

case 9: {

//Task 9

std::map<std::string, int> salary\_map = {

{"Ivanov", 45500},

{"Petrov", 37000},

{"Sidorov", 245000},

{"Petrenko", 65000},

{"Tovalds", 91200},

{"Popov", 54600},

{"Andrienko", 35000}

};

//принадлежность сотрудников к отделам

std::map<std::string, std::string> department\_map = {

{"Ivanov", "Research"},

{"Petrov", "Research"},

{"Sidorov", "Management"},

{"Petrenko", "Management"},

{"Tovalds", "Development"},

{"Popov", "Research"},

{"Andrienko", "Sales"}

};

int total\_salary = 0;

//по отделам

std::map<std::string, int> department\_salary;

std::map<std::string, int>::iterator it;

for (it = salary\_map.begin(); it != salary\_map.end(); ++it) {

total\_salary += it->second; // прибавляем зарплату к общему фонду

const std::string& name = it->first;

const std::string& department = department\_map[name];

department\_salary[department] += it->second;

}

std::cout << "Research " << department\_salary["Research"] << std::endl;

std::cout << "Management " << department\_salary["Management"] << std::endl;

std::cout << "Development " << department\_salary["Development"] << std::endl;

std::cout << "Sales " << department\_salary["Sales"] << std::endl;

std::cout << "Overall " << total\_salary << std::endl;

break;

}

default: {

std::cout << "неверный номер задания" << std::endl;

break;

}

}

}

return 0;

}

**input.txt output.txt**

1. **Тестирование**



 





