**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 ИУ5-24б |  | преподаватель каф. ИУ5 |
| Байдаков В.М. |  |  |
| Подпись и дата: |  | Подпись и дата: |
|  |  |  |

Москва, 2024 г.

**Постановка задачи**

Целью лабораторной работы изучение возможностей стандартной библиотеки шаблонов.

**Задание -** Выполнить индивидуальные задания по варианту.

Сами задания(1-3 был взят 2 вариант 4-9 третий):

1.Пользователь вводит два слова по 8 символов (две переменные). Сколько раз во втором слове встречается последняя буква первого слова.

2.Пользователь вводит некоторую последовательность символов. Образовать новую последовательность, включив в нее символы исходной, кроме символов пробелов и скобок.

3.Дан текстовый файл. Записать в один новый файл только буквы из исходного файла и пробелы, в другой новый файл только цифры и пробелы.

4.Создайте вектор vector<float> и занесите в него 25 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.

6.Описать функцию, которая печатает Yes или No в зависимости от того, содержится ли заданное целое число x в заданном контейнере-списке list<int>.

7.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.

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.

9.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.

**Текст программы**

Header.h

#include <iostream>

#include <string>

#include <fstream>

#include <vector>

#include <list>

#include <map>

#include <cmath>

void first();

void second();

void third();

void fourth();

void fifth();

void sixth();

void seventh();

void eighth();

void ninth();

main.cpp

#include "header.h"

int main()

{

  system("chcp 65001");

  unsigned short ans;

  while (true)

  {

    std::cout << "\n\nвыберите задачу для демонстрации(нажмите 0 для выхода):";

    std::cin >> ans;

    std::cout << std::endl;

    std::cin.ignore(std::numeric\_limits<std::streamsize>::max(), '\n'); // эта функция очищает поток чтобы была возможность последовательного ввода для заданий без применения getline

    switch (ans)

    {

    case 1:

      first();

      break;

    case 2:

      second();

      break;

    case 3:

      third();

      break;

    case 4:

      fourth();

      break;

    case 5:

      fifth();

      break;

    case 6:

      sixth();

      break;

    case 7:

      seventh();

      break;

    case 8:

      eighth();

      break;

    case 9:

      ninth();

      break;

    case 0:

      return 0;

    default:

      std::cout << "попробуй снова))" << std::endl;

      break;

    }

  }

}

void first()

{

  std::string str1, str2;

  do

  {

    std::cout << "\nEnter 2 words with size 8:";

    std::cin >> str1 >> str2;

  } while (str1.length() != 8 || str2.length() != 8);

  char lastSymbol = str1[str1.length() - 1];

  unsigned count = 0;

  for (size\_t i = 0; i < str2.length(); i++)

  {

    if (str2[i] == lastSymbol)

    {

      count++;

    }

  }

  std::cout << "\nResult: " << count;

}

void second()

{

  std::string str1;

  std::cout << "\nEnter text:";

  std::getline(std::cin, str1);

  std::string str2 = "";

  for (size\_t i = 0; i < str1.length(); i++)

  {

    if (str1[i] != ' ' && str1[i] != '(' && str1[i] != ')' && str1[i] != '{' && str1[i] != '}')

    {

      str2 += str1[i];

    }

  }

  std::cout << "\nResult: " << str2;

}

void third()

{

  std::ifstream source{"text.txt"};

  std::ofstream second{"2.txt"};

  std::ofstream first{"1.txt"};

  char temp;

  for (source.get(temp); !source.eof(); source.get(temp))

  {

    if ((temp <= 'Z' && temp >= 'A') || (temp >= 'a' && temp <= 'z') || static\_cast<int>(temp) == 32)

    {

      first << temp;

    }

    else if ((temp >= '0' && temp <= '9') || static\_cast<int>(temp) == 32)

    {

      second << temp;

    }

  }

  source.close();

  first.close();

  second.close();

}

void print(const std::vector<float> &vec)

{

  for (float i : vec)

  {

    std::cout << i << ' ';

  }

  std::cout << '\n';

}

void fourth()

{

  std::vector<float> v;

  for (size\_t i = 0; i < 25; i++)

  {

    v.push\_back(std::rand() / (double)RAND\_MAX \* 10);

  }

  std::cout << "start:\n";

  print(v);

  for (float &i : v)

  {

    i = i \* i;

  }

  std::cout << "after sqrt:\n";

  print(v);

  v.erase(v.begin() + 5);

  std::cout << "result:\n";

  print(v);

}

class Card

{

private:

  std::string title;

  std::string author;

  unsigned number;

public:

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

  friend void fifth();

};

void fifth()

{

  std::vector<Card> v = {

      Card("how to become baddest guy", "kolya chernev", 3),

      Card("c++ best guide in da world", "olga yurevna", 5),

      Card("prikoly 2005", "kirill volkov", 6),

      Card("vs code guide", "ya", 7),

      Card("how to be coolest rector", "mikhail gordin", 0)};

  std::cout << "available:\n";

  for (const Card &c : v)

  {

    std::cout << "title: " << c.title << ", author: " << c.author << ", number: " << c.number << '\n';

  }

  std::string input;

  std::cout << "enter author:" << std::endl;

  std::getline(std::cin, input);

  bool haveFound = false;

  for (const Card &c : v)

  {

    if (c.author == input)

    {

      std::cout << "title: " << c.title << ", author: " << c.author << ", number: " << c.number << '\n';

      haveFound = true;

    }

  }

  if (!haveFound)

  {

    std::cout << "can't find((\n";

  }

}

void sixth()

{

  std::list<int> l = {-1, 4321, -2312, 123, -1212, 131, 124, -124124, 3, -4, 0};

  int x;

  std::cin >> x;

  for (int obj : l)

  {

    if (obj == x)

    {

      std::cout << "Yes\n";

      return;

    }

  }

  std::cout << "No\n";

}

class Complex

{

public:

  double real;

  double imag;

  Complex(double r = 0, double i = 0) : real(r), imag(i) {}

  Complex operator+(const Complex &c)

  {

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

  }

  Complex &operator+=(const Complex &c)

  {

    \*this = \*this + c;

    return \*this;

  }

  friend std::ostream &operator<<(std::ostream &out, const Complex &c)

  {

    if (c.imag > 0)

    {

      out << c.real << " +"

          << "i \* " << fabs(c.imag) << '\n';

    }

    else

    {

      out << c.real << " -"

          << "i \* " << fabs(c.imag) << '\n';

    }

    return out;

  }

};

void seventh()

{

  std::list<Complex> l = {

      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)};

  for (Complex c : l)

  {

    std::cout << c;

  }

  Complex s;

  for (Complex c : l)

  {

    s += c;

  }

  std::cout << "answer:\n"

            << s;

}

void eighth()

{

  std::map<int, int> pairs;

  for (int i = 0; i < 20; i++)

  {

    pairs[i + 1] = (i + 1) \* 10;

  }

  std::cout << "first\n";

  for (const auto &pair : pairs)

  {

    std::cout << pair.first << " - " << pair.second << '\n';

  }

  for (auto i = pairs.begin(); i != pairs.end();)

  {

    if (i->first % 2 != 0)

    {

      i = pairs.erase(i);

      // for (const auto &pair : pairs)

      // {

      //   std::cout << pair.first << " - " << pair.second << '\n';

      // }

    }

    else

    {

      i++;

    }

  }

  std::cout << "final:\n";

  for (const auto &pair : pairs)

  {

    std::cout << pair.first << " - " << pair.second << '\n';

  }

}

void ninth()

{

  std::map<std::string, unsigned> salaries{

      {"Ivanov", 45500},

      {"Petrov", 37000},

      {"Sidorov", 245000},

      {"Petrenko", 65000},

      {"Tovalds", 91200},

      {"Popov", 54600},

      {"Andrienko", 35000}};

  std::map<std::string, std::string> groups{

      {"Ivanov", "Research"},

      {"Petrov", "Research"},

      {"Sidorov", "Management"},

      {"Petrenko", "Management"},

      {"Tovalds", "Development"},

      {"Popov", "Research"},

      {"Andrienko", "Sales"}};

  std::map<std::string, unsigned> groupsSalaries;

  int totalSalary = 0;

  for (const auto &salaryRecord : salaries)

  {

    std::string employee = salaryRecord.first;

    unsigned salary = salaryRecord.second;

    std::string department = groups[employee];

    groupsSalaries[department] += salary;

    totalSalary += salary;

  }

  for (const auto &groupsalary : groupsSalaries)

  {

    std::cout << groupsalary.first << " " << groupsalary.second << '\n';

  }

  std::cout << "Overall " << totalSalary << '\n';

}

CMakeLists.txt

cmake\_minimum\_required(VERSION 3.23)

set(project "lab8")

project(${project})

set(CMAKE\_CXX\_STANDARD 17)

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

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

configure\_file(${CMAKE\_CURRENT\_SOURCE\_DIR}/1.txt

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

configure\_file(${CMAKE\_CURRENT\_SOURCE\_DIR}/2.txt

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

set(SOURCES

    main.cpp

)

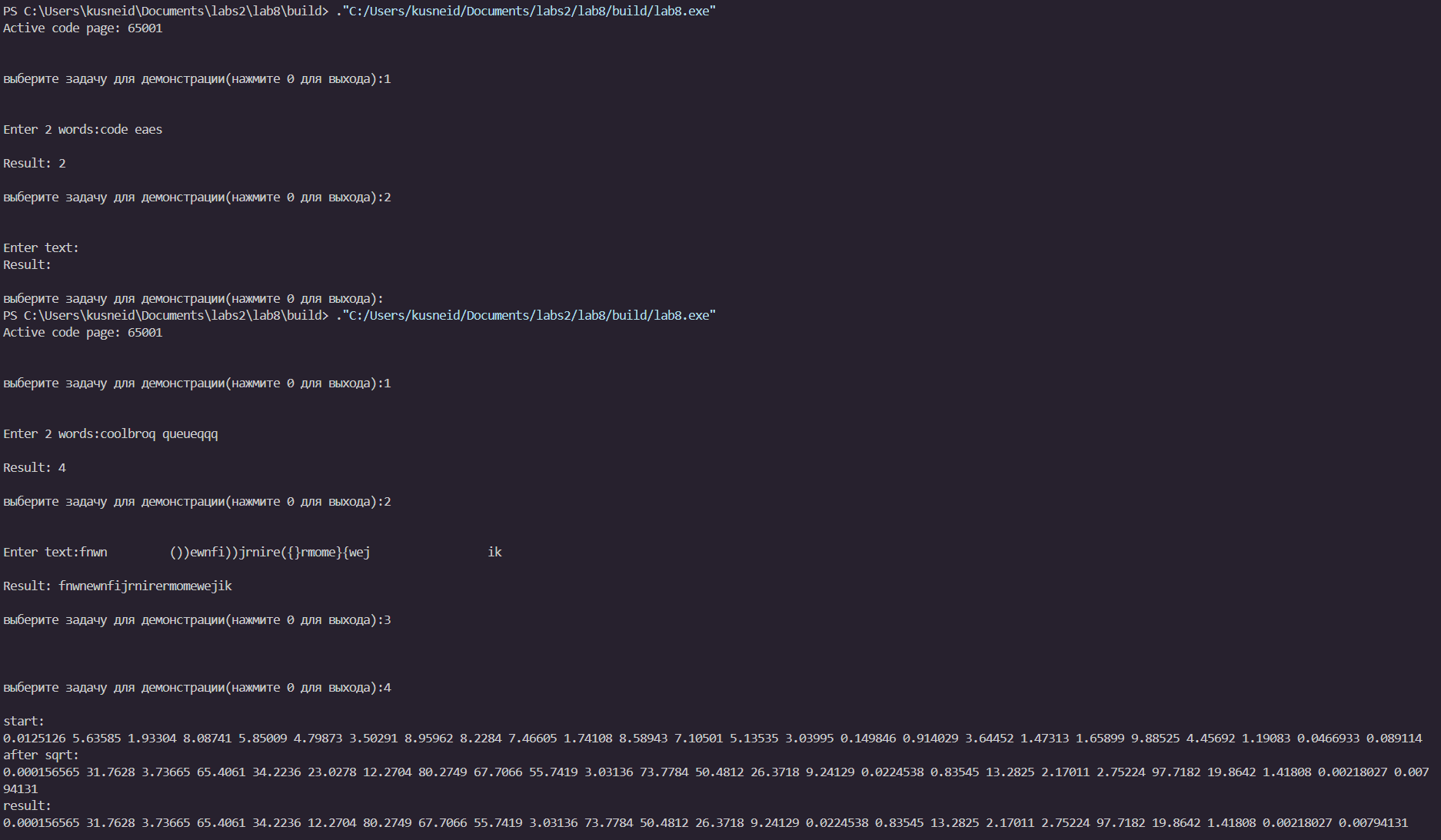
set(HEADER

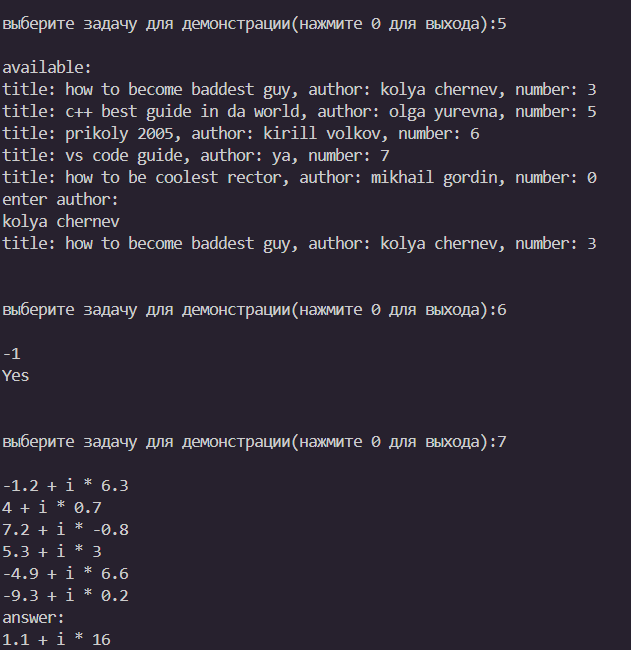
  header.h

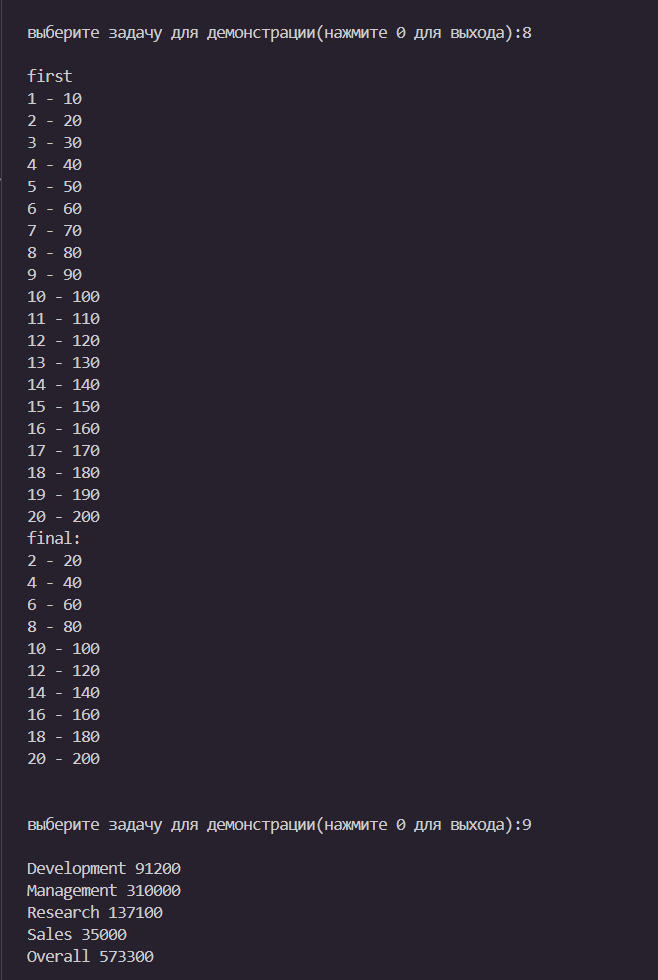
)

add\_executable(${project} ${SOURCES} ${HEADER})

Анализ результатов



****

****