**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра САПР**

отчет

**по лабораторной работе №5**

**по дисциплине «****Объектно-ориентированное программирование»**

Тема: «Создание и использование массивов»

|  |  |  |
| --- | --- | --- |
| Студенты гр. 1335 |  | Максимов Ю. Е |

|  |  |  |
| --- | --- | --- |
| Преподаватель: |  | Новакова Н. Е. |

Санкт-Петербург

2024

# **1. Цель работы**

Изучение массивов в методах в языке C++ с помощью программного продукта компании CLion.

# **2. Анализ задачи**

Необходимо:

1. Написать программу, которая считывает данные из текстового файла, содержимое файла помещается в массив символов, необходимо посчитать количество гласных и согласных букв, количество строк и общее количество символов;
2. Написать программу, которая использует массивы для перемножения матриц.

# **3. Ход выполнения работы**

## **3.1 Упражнение 1**

В ходе выполнения данного упражнения написана программа, которая считывает данные из файла, записывает в массив и выводит на консоль информацию о количестве гласных, согласных букв и о количестве строк.

### 3.1.1 Пошаговое описание алгоритма

Считать данные из файла в массив и вычислить количество гласных, согласных букв и строк.

На экран пользователя вывести полученные числа.

### 3.1.2 Используемые классы и методы

В программе, написанной в данном упражнении, используются следующие методы:

- std::cout – служит для отображения на экране строк и значений переменных, переданных в метод в качестве параметров, с переходом на новую строку;

- std::cin – ожидает следующего нажатия клавиши пользователем;

- main() – служит для запуска программы.

### 3.4.3 Контрольный пример

На рис.3.1 представлены результаты выполнения программы 1.

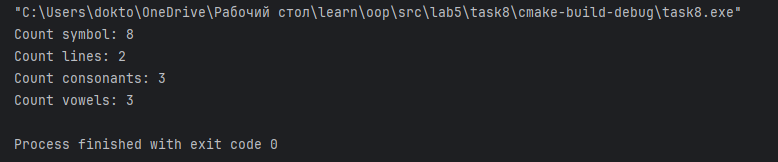
****

Рис.3.1 Контрольный пример для программы 1

Как видно из рисунка, на экран выведены значения количества гласных, согласных, строк, общее количество символов и длина.

### 3.2 Упражнение 2

В ходе выполнения данного упражнения, написанна программа, которая перемножает матрицы размера 2х2.

## **3.2.1 Пошаговое описание алгоритма**

Введение пользователем двух матриц размера 2х2.

Перемножение матриц.

На экран пользователя выводится полученная матрица.

### 3.2.2 Используемые классы и методы

В программе, написанной в данном упражнении, используются следующие методы:

- std::cout – служит для отображения на экране строк и значений переменных, переданных в метод в качестве параметров, с переходом на новую строку;

-std::cin – ожидает следующего нажатия клавиши пользователем [1];

- main() – служит для запуска программы.

### 3.2.3 Контрольный пример

На рис.3.2 представлены результаты выполнения программы 2.

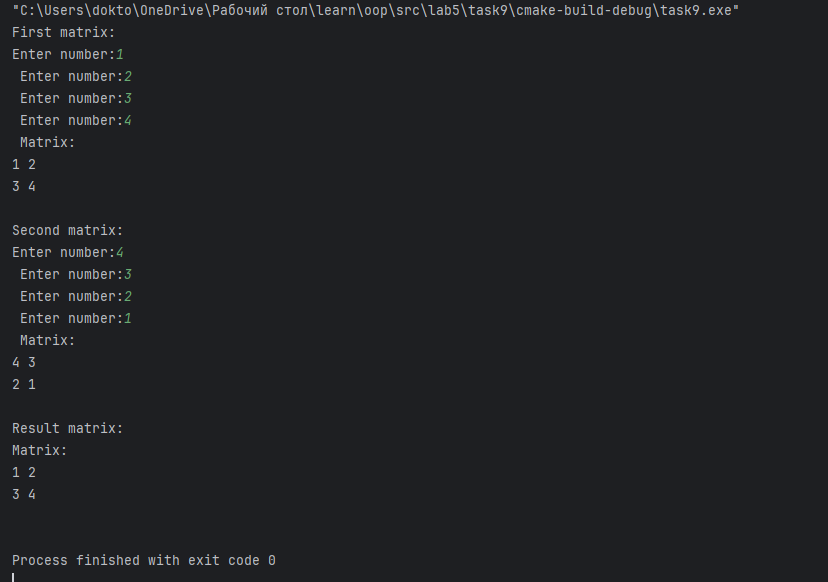
****

Рис.3.2 Контрольный пример для программы 2

Как видно из рисунка, на экран выведена матрица, равная перемножению матриц, введенных пользователем.

# **4. Листинг программы**

**Первая программа:**

**core.h**

//

// Created by Юлий Максимов on 11.06.2024.

//

#pragma once

namespace root {

class Core {

public:

static auto procces() -> void;

};

}

**core.cpp**

//

// Created by Юлий Максимов on 11.06.2024.

//

#include "iostream"

#include "core.h"

#include "read/ReadFile.h"

#include "sort/SortStr.h"

namespace {

/\*\*

\* Prints the given variable with the given string prefix.

\*

\* @param var The variable to be printed.

\* @param str The string prefix to be printed before the variable.

\*

\* @return void

\*

\* @throws None

\*/

template <typename T>

auto out(T var, const std::string &str) {

std::cout << str << var << std::endl;

}

}

/\*\*

\* Processes the input file and prints the count of symbols, lines, consonants, and vowels.

\*

\* @return void

\*

\* @throws None

\*/

auto root::Core::procces() -> void {

const auto [

countSymbol,

countLines,

countVowels,

countConsonants]

= str::SortStr::sort(str::ReadFile::read());

out(countSymbol, "Count symbol: ");

out(countLines, "Count lines: ");

out(countConsonants, "Count consonants: ");

out(countVowels, "Count vowels: ");

}

**ReadFile.cpp**

//

// Created by Юлий Максимов on 11.06.2024.

//

#include "ReadFile.h"

#include <fstream>

/\*\*

\* Reads the contents of a file and returns it as a string.

\*

\* @return The contents of the file as a string.

\*

\* @throws std::logic\_error If the file cannot be opened.

\*/

auto str::ReadFile::read() -> std::string {

std::string line;

std::ifstream in("../test.txt");

if (in.is\_open())

{

while (!in.eof())

{

std::string tmp;

std::getline(in, tmp);

line += tmp;

line += '\n';

}

}else {

throw std::logic\_error("Not find file");

}

in.close();

return line;

}

**ReadFile.h**

//

// Created by Юлий Максимов on 11.06.2024.

//

#pragma once

#include "string"

namespace str {

class ReadFile {

public:

static auto read() -> std::string;

};

}

**SortStr.h**

//

// Created by Юлий Максимов on 11.06.2024.

//

#pragma once

#include "string"

namespace str {

struct CountFile{

/\*\*

\* Constructs a CountFile object with the given values.

\*

\* @param i the number of consonants

\* @param x the number of lines

\* @param f the number of symbols

\* @param g the number of vowels

\*/

CountFile(const int i, const int x, const int f, const int g) {

countConsonants = i;

countLines = x;

countSymbol = f;

countVowels = g;

}

int countSymbol;

int countLines;

int countVowels;

int countConsonants;

};

class SortStr {

public:

static auto sort(const std::string& str) -> CountFile;

};

}

**SortStr.cpp**

//

// Created by Юлий Максимов on 11.06.2024.

//

#include "SortStr.h"

/\*\*

\* Sorts a given string and returns a CountFile object containing the count of symbols, lines, vowels, and consonants.

\*

\* @param str The string to be sorted.

\*

\* @return A CountFile object containing the count of symbols, lines, vowels, and consonants.

\*

\* @throws None

\*/

auto str::SortStr::sort(const std::string &str) -> CountFile {

CountFile tmp(0,0,0,0);

tmp.countSymbol = static\_cast<int>(str.size());

for(const auto &i: str) {

if(const auto x = std::tolower(i);

x == 'e' || x == 'y' || x == 'u' || x == 'i' || x == 'o' || x == 'a')

{

tmp.countVowels++;

}else if (i == '\n'){

tmp.countLines++;

}else {

tmp.countConsonants++;

}

}

return tmp;

}

**main.cpp**

#include "core/core.h"

int main() {

root::Core::procces();

return 0;

}

**CmakeLists.txt**

cmake\_minimum\_required(VERSION 3.15.0)

include\_guard(GLOBAL)

project(task8

VERSION 0.0.1

DESCRIPTION "task6 for OOP"

LANGUAGES C CXX

)

if(NOT CMAKE\_CXX\_STANDARD)

message(STATUS "[${PROJECT\_NAME}] setting c++ standard to c++23")

set(CMAKE\_CXX\_STANDARD 23)

set(CMAKE\_CXX\_STANDARD\_REQUIRED ON)

set(CMAKE\_CXX\_EXTENSIONS OFF)

endif()

add\_executable( ${PROJECT\_NAME}

${CMAKE\_CURRENT\_SOURCE\_DIR}/src/c++/main.cpp

${CMAKE\_CURRENT\_SOURCE\_DIR}/src/c++/core/Core.cpp

${CMAKE\_CURRENT\_SOURCE\_DIR}/src/c++/core/Core.h

${CMAKE\_CURRENT\_SOURCE\_DIR}/src/c++/utils/utils.h

)

target\_include\_directories(${PROJECT\_NAME}

PRIVATE

${CMAKE\_CURRENT\_SOURCE\_DIR}/src/c++

)

**Вторая программа:**

**core.h**

//

// Created by Юлий Максимов on 11.06.2024.

//

#pragma once

namespace root {

class Core {

public:

static auto procces() -> void;

};

}

**core.cpp**

//

// Created by Юлий Максимов on 11.06.2024.

//

#include "iostream"

#include "core.h"

#include "matrix/Matrix.h"

/\*\*

\* Processes the core functionality of the root::Core class.

\*

\* This function prompts the user to enter two matrices, creates them using the math::Matrix class,

\* and then performs matrix multiplication between the two matrices. The result is printed to the console.

\*

\* @throws None

\*/

auto root::Core::procces() -> void {

std::cout << "First matrix: " << std::endl;

math::Matrix matrix1;

matrix1.createMatrix();

matrix1.out();

std::cout << "Second matrix: " << std::endl;

math::Matrix matrix2;

matrix2.createMatrix();

matrix2.out();

std::cout << "Result matrix: " << std::endl;

math::Matrix matrix3 = matrix1 \* matrix2;

matrix1.out();

}

**Matrix.cpp**

//

// Created by dokto on 11.06.2024.

//

#include "matrix.h"

#include <iostream>

using VectorInt = std::vector<int>;

using MatrixInt = std::vector<VectorInt>;

namespace {

/\*\*

\* Reads an integer from the user input.

\*

\* @return The integer entered by the user.

\*

\* @throws None

\*/

auto inputNum() -> int {

int var;

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

std::cin >> var;

return var;

}

/\*\*

\* Generates a vector of integers by calling the `inputNum()` function twice.

\*

\* @return A vector of integers containing the values returned by `inputNum()`.

\*/

auto inputVec() -> VectorInt {

return {inputNum(), inputNum()};

}

/\*\*

\* Generates a matrix of integers by calling the `inputVec()` function twice.

\*

\* @return A matrix of integers containing the values returned by `inputVec()`.

\*/

auto createMatrix() -> MatrixInt {

return {{inputVec()}, {inputVec()}};

}

}

/\*\*

\* Default constructor for the math::Matrix class.

\*

\* This constructor initializes a math::Matrix object with default values.

\*

\* @return void

\*/

math::Matrix::Matrix() = default;

/\*\*

\* Initializes the matrix of the math::Matrix object by calling the ::createMatrix() function.

\*

\* @throws None

\*/

void math::Matrix::createMatrix() {

matrix = ::createMatrix();

}

/\*\*

\* Outputs the matrix to the console.

\*

\* This function iterates over the rows and columns of the matrix and prints

\* each element to the console. After printing each row, a new line is

\* printed. Finally, a new line is printed after printing the entire matrix.

\*

\* @return void

\*

\* @throws None

\*/

auto math::Matrix::out() -> void {

std::cout << "Matrix: " << std::endl;

for(const auto& i: matrix) {

for(const auto &k: i) {

std::cout << k << " ";

}

std::cout << std::endl;

}

std::cout << std::endl;

}

/\*\*

\* Multiplies two matrices together.

\*

\* This function multiplies matrix `a` by matrix `b` and returns the result as a new matrix `res`.

\* The resulting matrix `res` has the same number of rows as matrix `a` and the same number of columns as matrix `b`.

\* The elements of the resulting matrix `res` are computed by summing the products of the corresponding elements of matrix `a` and matrix `b`.

\*

\* @param a The first matrix to be multiplied.

\* @param b The second matrix to be multiplied.

\* @return The resulting matrix after the multiplication.

\*

\* @throws None.

\*/

math::Matrix math::operator\*(const Matrix &a, const Matrix &b) {

Matrix res;

res.matrix.resize(a.matrix.size());

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

for (int j = 0; j < b.matrix[0].size(); ++j) {

int sum = 0;

for (int k = 0; k < a.matrix[0].size(); ++k) {

sum += a.matrix[i][k] \* b.matrix[k][j];

}

res.matrix[i].push\_back(sum);

}

}

return **res;**

**}**

**Matrix.h**

//

// Created by dokto on 11.06.2024.

//

#pragma once

#include "vector"

namespace math{

class Matrix {

using VectorInt = std::vector<int>;

using MatrixInt = std::vector<VectorInt>;

public:

Matrix();

~Matrix() = default;

auto createMatrix() -> void;

auto out() -> void;

friend Matrix operator\*(const Matrix&a, const Matrix& b);

private:

MatrixInt matrix;

};

}

**main.cpp**

#include "core/core.h"

int main() {

root::Core::procces();

return 0;

}

**CmakeLists.txt**

cmake\_minimum\_required(VERSION 3.15.0)

include\_guard(GLOBAL)

project(task8

VERSION 0.0.1

DESCRIPTION "task6 for OOP"

LANGUAGES C CXX

)

if(NOT CMAKE\_CXX\_STANDARD)

message(STATUS "[${PROJECT\_NAME}] setting c++ standard to c++23")

set(CMAKE\_CXX\_STANDARD 23)

set(CMAKE\_CXX\_STANDARD\_REQUIRED ON)

set(CMAKE\_CXX\_EXTENSIONS OFF)

endif()

add\_executable( ${PROJECT\_NAME}

${CMAKE\_CURRENT\_SOURCE\_DIR}/src/c++/main.cpp

${CMAKE\_CURRENT\_SOURCE\_DIR}/src/c++/core/Core.cpp

${CMAKE\_CURRENT\_SOURCE\_DIR}/src/c++/core/Core.h

${CMAKE\_CURRENT\_SOURCE\_DIR}/src/c++/utils/utils.h

)

target\_include\_directories(${PROJECT\_NAME}

PRIVATE

${CMAKE\_CURRENT\_SOURCE\_DIR}/src/c++

)

# **5. Полученные результаты**

В ходе выполнения данной лабораторной работы нами были получены следующие результаты:

* в ходе работы программы 1 был создан массив из значений, взятых из файла, и посчитаны количества букв, длина.
* в ходе работы программы 2 был созданы массивы матриц размера 2х2 и выполнено их перемножение.

# **6. Выводы**

В ходе выполнения данной лабораторной работы:

* были изучены массивы в языке C++;