Министерство науки и высшего образования Российской Федерации

федеральное государственное бюджетное образовательное учреждение высшего образования

«Санкт-Петербургский государственный технологический институт

(технический университет)»

**Дисциплина «Программирование»**

Отчёт по лабораторной работе №1.

«Использование языка программирования С++ для математических расчётов»

Вариант № 12

Выполнил: Нерадовский Артемий Андреевич,

студент 494 группы.

Преподаватели: Корниенко Иван Григорьевич,

Федин Алексей Константинович.

Санкт-Петербург

2020

**Постановка задачи**

В трехмерном пространстве пользователем задаются координаты (x, y, z) точки, а также координаты центра сферы и ее радиус.

При выполнении лабораторной работы для каждого объекта, использующегося в программе, должна быть создана соответствующая структура данных, а также нельзя использовать контейнеры и алгоритмы библиотеки STL или других сторонних библиотек. В программе должны быть предусмотрены два варианта ввода исходных данных: пользователем с клавиатуры или из пользовательского файла. Также в работе должна присутствовать возможность сохранения введенных пользователем с клавиатуры данных, сохранения результата работы программы и модульного тестирования.

Исходные данные

В качестве исходных данных программа использует:

* Координаты одной или нескольких точек, координаты центра сферы и ее радиус. В программе хранятся в структуре Point3D в виде семи переменных типа double: x0, y0, z0, x, y, z, radlenth. Единицы измерения - единичный отрезок системы координат.
* Могут быть введены пользователем либо в консоль, либо в файл “.txt”

Особые ситуации

Необходимо рассмотреть следующие особые ситуации:

* Отсутствие ожидаемых программой файлов на чтение, нахождение внутри файлов некорректных данных.
* Попытка пользователя обратиться к файлу с зарезервированным именем.
* Запись работы программы в уже существующий файл или создание недопустимого файла, попытка записи данных в файл, доступный только для чтения.

Математические методы и алгоритмы решения задач

Определить границы радиуса сферы можно с помощью уравнения (1), где “” это радиус сферы в квадрате, а “x, y, z, ” координаты точки и сферы соответсвенно. Вычислить расстояние от точки до центра сферы можно с помощью формулы (2), если это расстояние будет меньше, равно радиусу сферы или , то такая точка принадлежит данной сфере.

(1)

(2)

**Формат представления данных**

Для ввода данных точек и сферы внутри читаемого txt файла для обеспечения корректной работы должны быть записаны координаты центра сферы, длина ее радиуса и координаты точек. Все вводимые данные имеют тип double, при необходимости для корректной работы программы в качестве разделительного знака нужно использовать точку. Ввод производится строго в данном порядке.

Формат файлов для записи - текстовый txt.

Таблица 1 – Основные переменные программы

| **Имя структуры** | **Имя поля структуры** | **Тип данных** | **Описание** |
| --- | --- | --- | --- |
| Point3D  /  file\_var |  | double | Координата центра сферы по оси OX. |
|  | double | Координата центра сферы по оси OY. |
|  | double | Координата центра сферы по оси OZ. |
| x | double | Координата точки по оси OX. |
| y | double | Координата точки по оси OY. |
| z | double | Координата точки по оси OZ. |
| radlenth | double | Длина радиуса. |

Структура программы

Программа разбита на 9 модулей: cpp и h файлы интерфейса, cpp и h файлы вычислений, cpp и h файлы работы с файлами, cpp и h файлы модульных тестов и основной файл main.cpp.

Основная последовательность работы программы следующая: после запуска на экран выводится информация о программе и главное меню, после чего ожидается выбор пользователем дальнейших действий. Пункты главного меню: выполнить программу, вывести информацию о программе и авторе, выполнить тесты, выйти из программы. В случае выбора первого пункта на экран выведется вспомогательное меню. Пункты вспомогательного меню: выполнить программу через консоль, использовать значения из файла, обратно в меню. Посде ввода исходных данных с клавиатуры, в случае корректности исходных данных программа предложит сохранить исходные данные в файл, после выбора пользователя выведет результат и предложит сохранение результата в файл. После выбора пользователя программа выведет главное меню программы. Цикл будет продолжаться до тех пор, пока пользователь не закроет программу.

В модулях ui.cpp и ui.h содержится весь пользовательский интерфейс, который включает в себя информацию о программе и авторе, главное меню программы, меню выбора происхождения исходных данных, меню сохранения исходных данных, меню сохранения результата программы.

Модули functions.cpp и functions.h содержат основные вычислительные функции программы, среди которых: вычисление расстояния от точки до центра сферы.

Модули filework.cpp и filework.h содержат функции работы с файлами, среди которых: чтение исходных данных из файла, сохранение результата программы в файл.

Таблица 2 – Функции основного алгоритма

| **Имя** | **Описание** |
| --- | --- |
| double number\_check() | Функция для определения правильности ввода переменной типа double. |
| bool error\_check() | Функция для определения правильности ввода чисел. |
| double calc() | Функция для рассчета расстояния от точки до центра сферы. |
| void console\_way() | Функция, в которой проходят все вычисления при вводе исходных данных с клавиатуры. |
| bool file\_check\_size(string pFile) | Функция проверки файла на наличие в нем каких-либо данных. В качестве аргумента принимает путь файла. |
| int wtdw\_file(string& pfile) | Функция с меню, на случай если файл для сохранения данных, уже что-то в себе содержит. В качестве аргумента принимает путь файла. |

Продолжение таблицы 2

|  |  |
| --- | --- |
| **Имя** | **Описание** |
| bool file\_exist(string path) | Функция проверки существования файла. В качестве аргумента принимает путь файла. |
| bool save\_source(double\*\* arr, string pathfile, int modout) | Функция, для сохранения исходных данных. В качестве аргумента принимает массив, содержащий в себе координаты и длину радиуса, путь файла и переменную, определяющую в каком режиме открыть файл. |
| bool save\_results(int\*\* arr, string pathfile, int modout) | Функция, для сохранения результата, при вводе исходных данных с клавиатуры. В качестве аргумента принимает массив, содержащий в себе результат вычислений, путь файла и переменную, определяющую в каком режиме открыть файл. |
| void file\_work() | Функция, в которой проходят все расчеты, при использовании исходных данных из файла. |
| int result(double x0, double y0, double z0, double x, double y, double z, double radlenth) | Функция, которая определяет принадлежность точки к сфере. В качестве аргумента принимает координаты точки и центра сферы, а также длину радиуса. |

Продолжение таблицы 2

|  |  |
| --- | --- |
| **Имя** | **Описание** |
| bool file\_name\_check(string path) | Функция, проверяющая имя файла, из которого считывают данные, на зарезервированное системой и на наличие расширения «.txt». В качестве аргумента принимает путь к файлу. |
| void file\_print\_result(string adresFile, string pathFile, int modout) | Функция, для сохранения результата программы. В качестве аргумента принимает путь файла ввода, путь файла вывода, и переменную, определяющую в каком режиме открыть файл. |
| bool check\_read\_only(string filename) | Функция, проверяющая файл по указанному пути на наличие атрибута «Только для чтения». В качестве аргумента принимает путь к файлу. |

Блок-схемы алгоритмов программы

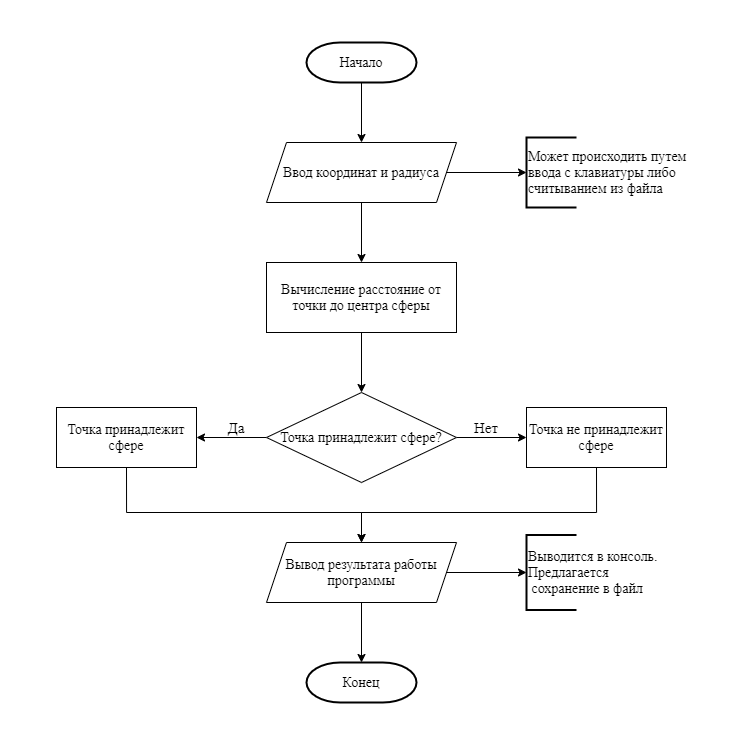


Рисунок 1 - Блок-схема основной функции программы

Описание хода выполнения лабораторной работы

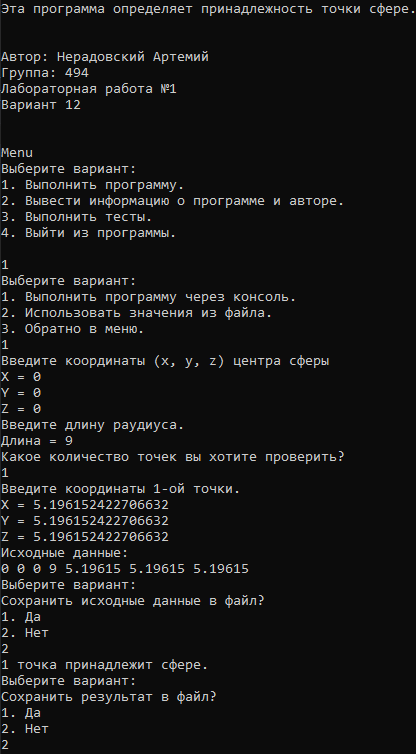
Первым шагом в выполнении данной лабораторной работы стало создание решения и проекта в среде разработки Microsoft Visual Studio C++ 2017. Изначально код программы писался в одном модуле, после чего этот модуль был разбит на пять частей: модуль пользовательского интерфейса, основной модуль работы программы, вычислительный модуль, модуль работы с файлами, модуль тестирования. После этого все модули были разбиты еще на две части: заголовочный файл и файл, содержащий описание функций.

Изначально была написана функция, выполняющая математические расчеты, необходимые для выполнения поставленной задачи, и функции проверки корректности вводимых значений. Затем был создан пользовательский интерфейс, включающий в себя информацию о программе и авторе, меню для каждого шага работы программы. После был написан ввод из файла и вывод в файл, модульные тесты.

В ходе работы над кодом программы были обработаны такие нестандартные случаи, как ввод пользователем некорректных данных, попытка использовать файл с зарезервированным OS Windows именем, попытка записи информации во внешний файл, имеющий атрибут «Только для чтения», а также запись результата в файл, в который ранее уже были записаны исходные данные.

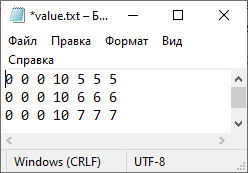
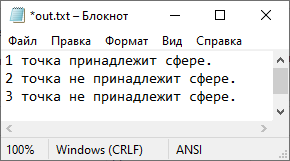
**Результаты работы программы**

В качестве результата корректной работы программы выводится сообщение принадлежности точки к сфере.



Ввод данных координат точки осуществляется в консоль. Пользователь может выбрать сохранить или не сохранять данные во внешний файл.

Рисунок 2 – Пример работы программы



Рисунки 3, 4 – Содержимое файлов out.txt и value.txt, после сохранения в них результата и исходных данных соответсвенно

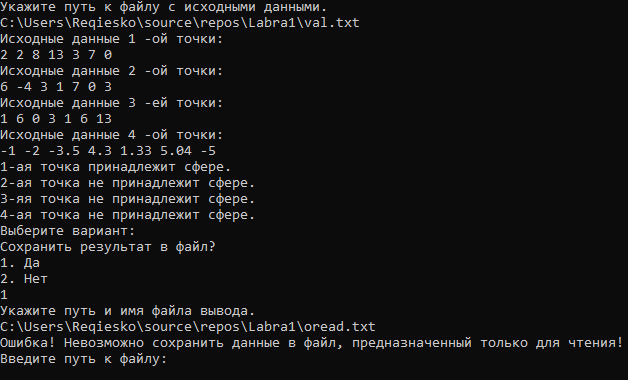
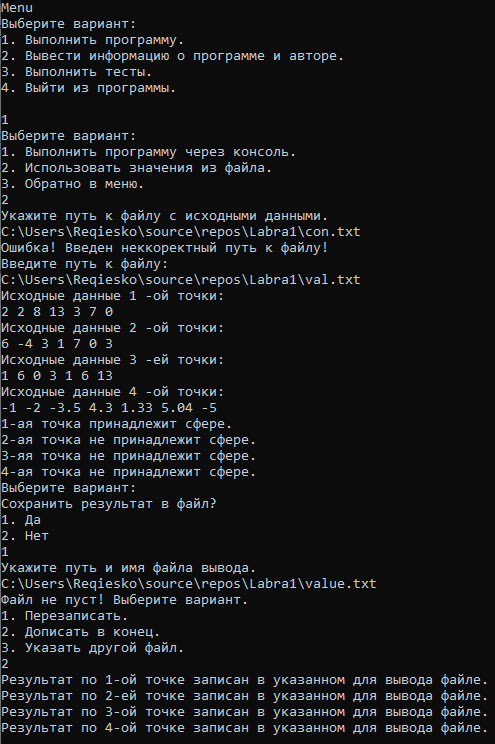


Рисунок 5 – попытка сохранить результат в файл, предназначенный только для чтения



Ввод данных точек осуществлен из файла. Пользователь ввел некорректный путь исходный данных, далее он решил сохранить результат работы в файл, который уже содержал информацию, после он выбрал дописать данные в конце файла.

Рисунок 6 – Другой пример работы программы

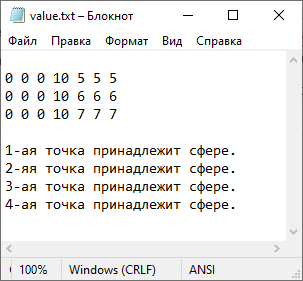


Рисунок 7 – Изменения в файле value.txt после записи в конец новых данных

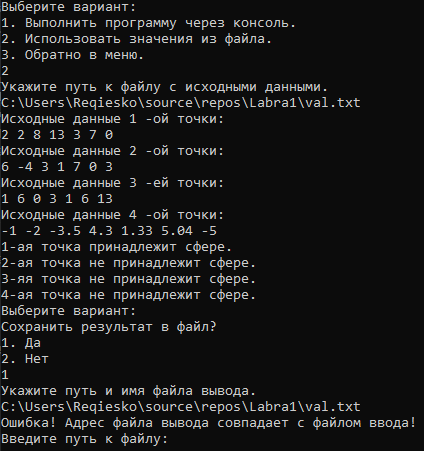


Рисунок 8 – Попытка сохранить результат в файл, содержащий исходные данные

Исходный текст программы

[Начало программы ---]

[Начало main.cpp --- ]

#include "functions.h"

int main() {

setlocale(LC\_ALL, "RU");

greeting();

menu();

return 0;

}

[Конец main.cpp --- ]

[Начало functions.cpp --- ]

#include "functions.h"

#include "tests.h"

#include "filework.h"

struct Point3D {

double x0, y0, z0, x, y, z, radlenth;

};

struct number {

int n;

};

Point3D pt;

number pn;

const int ignor = 32767;

const int c\_point = 7;

double number\_check() {

double a;

while (!(cin >> a) || (cin.peek() != '\n'))

{

cin.clear();

while (cin.get() != '\n');

cout << "Ошибка! Пожалуйста введите число! " << endl;

}

return a;

}

bool error\_check() {

if (cin.fail())

{

cin.clear();

cin.ignore(ignor, '\n');

cout << "Пожалуйста введите цифру 1, 2 или 3." << endl;

return 0;

}

return 1;

}

bool file\_exist(string path) {

ifstream file(path);

file.open(path, ios::in);

if (file.is\_open())

return true;

else

return false;

}

void console\_way() {

string pathc, adres;

ifstream foutcheck;

cout << "Введите координаты (x, y, z) центра сферы " << endl;

cout << "X = ";

pt.x0 = number\_check();

cout << "Y = ";

pt.y0 = number\_check();

cout << "Z = ";

pt.z0 = number\_check();

cout << "Введите длину раудиуса. " << endl;

cout << "Длина = ";

pt.radlenth = number\_check();

if (pt.radlenth <= 0) {

cout << "Ошибка! Радиус не может быть отрицательным и нулевым! " << endl;

sub\_menu();

}

cout << "Какое количество точек вы хотите проверить? " << endl;

cin >> pn.n;

double\*\* arr\_source = new double\* [pn.n];

for (int i = 0; i < pn.n; i++) {

arr\_source[i] = new double[c\_point];

}

int\* arr\_res = new int[pn.n];

int i;

for (i = 0; i < pn.n; i++) {

arr\_source[i][0] = pt.x0;

arr\_source[i][1] = pt.y0;

arr\_source[i][2] = pt.z0;

arr\_source[i][3] = pt.radlenth;

cout << "Введите координаты " << i + 1 << "-ой точки." << endl;

cout << "X = ";

pt.x = number\_check();

cout << "Y = ";

pt.y = number\_check();

cout << "Z = ";

pt.z = number\_check();

arr\_source[i][4] = pt.x;

arr\_source[i][5] = pt.y;

arr\_source[i][6] = pt.z;

arr\_res[i] = result(pt.x0, pt.y0, pt.z0, pt.x, pt.y, pt.z, pt.radlenth);

}

cout << "Исходные данные: " << endl;

for (i = 0; i < pn.n; i++) {

for (int j = 0; j < c\_point; j++) {

cout << arr\_source[i][j] << " ";

}

cout << endl;

}

cout << "Выберите вариант:" << endl;

cout << "Сохранить исходные данные в файл? " << endl;

if (result\_from\_console() == 1) {

cout << "Введите путь к файлу для сохранения исходных данных. " << endl;

cin >> adres;

foutcheck.open(adres, ios::in);

while (!check\_read\_only(adres)) {

foutcheck.close();

cout << "Введите путь к файлу: " << endl;

cin >> adres;

foutcheck.open(adres);

}

while (file\_name\_check(adres)) {

foutcheck.close();

cout << "Ошибка! Некорректное путь или имя файла." << endl;

cout << "Введите путь к файлу: " << endl;;

cin >> adres;

foutcheck.open(adres, ios::in);

}

if (file\_exist(adres) == false) {

foutcheck.close();

save\_source(arr\_source, adres, rewrite);

}

else {

foutcheck.close();

if (file\_check\_size(adres) == true) {

save\_source(arr\_source, adres, rewrite);

}

else {

save\_source(arr\_source, adres, wtdw\_file(adres));

}

}

}

for (i = 0; i < pn.n; i++) {

if (arr\_res[i] == 1) {

cout << i + 1 << " точка принадлежит сфере." << endl;

}

else if (arr\_res[i] == 0) {

cout << i + 1 << " точка не принадлежит сфере." << endl;

}

}

cout << "Выберите вариант:" << endl;

cout << "Сохранить результат в файл? " << endl;

if (result\_from\_console() == 1) {

cout << "Введите путь к файлу для сохранения результата. " << endl;

cin >> pathc;

while (adres == pathc) {

cout << "Адреса файлов совпадают. Введите путь к файлу: " << endl;

cin >> pathc;

}

foutcheck.open(pathc, ios::in);

while (!check\_read\_only(pathc)) {

foutcheck.close();

cout << "Введите путь к файлу: " << endl;

cin >> pathc;

while (adres == pathc) {

cout << "Адреса файлов совпадают. Введите путь к файлу: " << endl;

cin >> pathc;

}

foutcheck.open(pathc);

}

while (file\_name\_check(pathc)) {

foutcheck.close();

cout << "Ошибка! Некорректное путь или имя файла." << endl;

cout << "Введите путь к файлу: " << endl;

cin >> pathc;

while (adres == pathc) {

cout << "Адреса файлов совпадают. Введите путь к файлу: " << endl;

cin >> pathc;

}

foutcheck.open(pathc, ios::in);

}

if (file\_exist(pathc) == false) {

foutcheck.close();

save\_results(arr\_res, pathc, rewrite);

}

else {

foutcheck.close();

if (file\_check\_size(pathc) == true) {

save\_results(arr\_res, pathc, rewrite);

}

else {

save\_results(arr\_res, pathc, wtdw\_file(pathc));

}

}

}

for (i = 0; i < pn.n; i++) {

delete[] arr\_source[i];

}

delete[] arr\_source;

delete[] arr\_res;

menu();

}

void save\_source(double\*\* arr, string pathfile, int modout) {

ofstream foutsource;

if (modout == 1) {

foutsource.open(pathfile, ios::out);

foutsource << " " << endl;

}

if (modout == 2) {

foutsource.open(pathfile, ios::app);

foutsource << " " << endl;

}

for (int i = 0; i < pn.n; i++) {

for (int j = 0; j < c\_point; j++) {

foutsource << arr[i][j] << " ";

}

foutsource << "\n";

}

foutsource.close();

cout << "Исходные данные успешно сохранены! " << endl;

}

void save\_results(int\* arr, string pathfile, int modout) {

ofstream foutcon;

if (modout == 1) {

foutcon.open(pathfile, ios::out);

foutcon << " " << endl;

}

if (modout == 2) {

foutcon.open(pathfile, ios::app);

foutcon << " " << endl;

}

for (int i = 0; i < pn.n; i++) {

if (arr[i] == 1) {

foutcon << i + 1 << " точка принадлежит сфере." << endl;

}

else if (arr[i] == 0) {

foutcon << i + 1 << " точка не принадлежит сфере." << endl;

}

}

foutcon.close();

system("cls");

cout << "Результат успешно сохранен! " << endl;

}

double calc() {

double dotlenth = abs(sqrt(pow((pt.x - pt.x0), 2) + pow((pt.y - pt.y0), 2) + pow((pt.z - pt.z0), 2)));

return dotlenth;

}

bool file\_check\_size(string pFile) {

ifstream file\_check\_size;

file\_check\_size.open(pFile, ios::in);

file\_check\_size.seekg(0, ios::end);

if (file\_check\_size.tellg() == 0) {

file\_check\_size.close();

return true;

}

else {

file\_check\_size.close();

return false;

}

}

int wtdw\_file(string& pFile) {

bool d = true;

ifstream foutcheck;

int del;

while (d) {

cout << "Файл не пуст! Выберите вариант." << endl;

cout << "1. Перезаписать." << endl;

cout << "2. Дописать в конец." << endl;

cout << "3. Указать другой файл." << endl;

cin >> del;

if (!error\_check()) {

continue;

}

cin.ignore(ignor, '\n');

switch (del) {

case rewrite: {

return rewrite;

}

case add: {

return add;

}

case newfile: {

bool trg = true;

while (trg) {

string adresF;

cout << "Введите путь к файлу. " << endl;

cin >> pFile;

foutcheck.open(pFile, ios::in);

while (file\_name\_check(pFile)) {

foutcheck.close();

cout << "Ошибка! Некорректный путь или имя файла." << endl;

cout << "Введите путь к файлу: " << endl;

cin >> pFile;

foutcheck.open(pFile, ios::in);

}

while (!check\_read\_only(pFile)) {

foutcheck.close();

cout << "Введите путь к файлу: " << endl;

cin >> pFile;

foutcheck.open(pFile);

}

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

ifstream newfile(pFile);

newfile.close();

return rewrite;

}

else {

bool check = file\_check\_size(pFile);

foutcheck.close();

if (check) {

return 0;

}

else {

return wtdw\_file(pFile);

}

}

}

}

default: {

cout << "Пожалуйста введите цифру 1, 2 или 3." << endl;

}

}

}

}

[Конец functions.cpp --- ]

[Начало functions.h --- ]

#pragma once

#include <iostream>

#include <string>

#include <fstream>

#include <istream>

#include "Windows.h"

#include <filesystem>

#include "ui.h"

using namespace std;

using namespace std::experimental::filesystem;

bool file\_check\_size(string pFile);

int wtdw\_file(string& pfile);

void console\_way();

double calc();

double number\_check();

bool error\_check();

bool file\_exist(string path);

void save\_source(double\*\* arr, string pathfile, int modout);

void save\_results(int\* arr, string pathfile, int modout);

enum clearfile {

rewrite = 1,

add = 2,

newfile

};

[Конец functions.h --- ]

[Начало filework.cpp --- ]

#pragma once

#include <iostream>

#include <string>

#include <fstream>

#include <istream>

#include "Windows.h"

#include <filesystem>

#include "ui.h"

using namespace std;

using namespace std::experimental::filesystem;

bool file\_check\_size(string pFile);

int wtdw\_file(string& pfile);

void console\_way();

double calc();

double number\_check();

bool error\_check();

bool file\_exist(string path);

void save\_source(double\*\* arr, string pathfile, int modout);

void save\_results(int\* arr, string pathfile, int modout);

enum clearfile {

rewrite = 1,

add = 2,

newfile

};

[Конец filework.cpp --- ]

[Начало filework.h --- ]

#pragma once

#include "functions.h"

#include <filesystem>

using namespace std::experimental::filesystem;

void file\_work();

bool file\_name\_check(string path);

void file\_print\_result(string adresFile, string pathFile, int modout);

bool check\_read\_only(string filename);

[Конец filework.h --- ]

[Начало ui.cpp --- ]

#include "ui.h"

#include "tests.h"

#include "filework.h"

const int ignor = 32767;

void menu() {

menu\_choice floor = perform;

int variant = floor;

cout << endl;

while (true) {

cout << "Menu" << endl;

cout << "Выберите вариант:" << endl;

cout << "1. Выполнить программу." << endl;

cout << "2. Вывести информацию о программе и авторе." << endl;

cout << "3. Выполнить тесты. " << endl;

cout << "4. Выйти из программы. " << endl;

cout << endl;

cin >> variant;

if (!error\_check()) {

continue;

}

cin.ignore(ignor, '\n');

switch (variant) {

case perform: {

sub\_menu();

break;

}

case info: {

cout << endl;

greeting();

menu();

break;

}

case tests: {

system("cls");

test();

menu();

break;

}

case out: {

cout << "Программа завершена." << endl;

exit(0);

break;

}

default: {

cout << "Пожалуйста введите цифру 1, 2 или 3." << endl;

}

}

cin.clear();

}

}

void sub\_menu() {

sub\_menu\_choice floor = console;

int var = floor;

while (true) {

cout << "Выберите вариант:" << endl;

cout << "1. Выполнить программу через консоль. " << endl;

cout << "2. Использовать значения из файла. " << endl;

cout << "3. Обратно в меню. " << endl;

cin >> var;

if (!error\_check()) {

continue;

}

cin.ignore(ignor, '\n');

switch (var) {

case console: {

console\_way();

break;

}

case file: {

file\_work();

}

case back: {

menu();

}

default: {

cout << "Пожалуйста введите цифру 1, 2 или 3." << endl;

sub\_menu();

}

}

cin.clear();

}

}

int result\_from\_console() {

saveinfile floor = yes;

int var = floor;

int way = 0;

bool sw = true;

while (sw) {

cout << "1. Да " << endl;

cout << "2. Нет " << endl;

cin >> var;

if (!error\_check()) {

continue;

}

cin.ignore(ignor, '\n');

switch (var) {

case yes: {

way = 1;

sw = false;

break;

}

case no: {

sw = false;

break;

}

default: {

cout << "Пожалуйста введите цифру 1 или 2. " << endl;

}

}

}

return way;

}

void greeting() {

cout << "Эта программа определяет принадлежность точки сфере." << endl;

cout << endl;

cout << "Автор: Нерадовский Артемий" << endl;

cout << "Группа: 494" << endl;

cout << "Лабораторная работа №1" << endl;

cout << "Вариант 12" << endl;

cout << endl;

}

[Конец ui.cpp --- ]

[Начало ui.h --- ]

#pragma once

#include "functions.h"

enum menu\_choice {

perform = 1,

info = 2,

tests = 3,

out = 4

};

enum sub\_menu\_choice {

console = 1,

file = 2,

back = 3

};

enum saveinfile {

yes = 1,

no = 2

};

void menu();

void sub\_menu();

int result\_from\_console();

void greeting();

[Конец ui.h --- ]

[Начало test.cpp --- ]

#include "tests.h"

const int count\_of\_tests = 5;

int result(double x0, double y0, double z0, double x, double y, double z, double radlenth) {

double r;

double dotlenth;

r = pow(radlenth, 2);

dotlenth = abs(sqrt(pow((x - x0), 2) + pow((y - y0), 2) + pow((z - z0), 2)));

if (r == pow(dotlenth, 2) || dotlenth < radlenth) {

return 1;

}

else {

return 0;

}

}

void test() {

double x0, y0, z0, x, y, z, radlenth;

int count = 0;

string expected\_res\_str;

string actual\_res\_str;

int expected\_res;

int actual\_res;

int result\_test[count\_of\_tests] = {0, 1, 1, 0, 0};

double coordinates[count\_of\_tests][7] = { {0, 0, 0, -10, 10, 10, 10}, {0, 0, 0, 2, 2, 2, 5}, {6, 0, -11, 12, 5, -9, 15}, {-5.5, 4.5, -2, 7, 1, 0, 11}, {2.4, -7.28, 3.51, -5.7, 11, 3.84, 9} };

for (int i = 0, j = 0; i < count\_of\_tests;) {

expected\_res = result\_test[i];

x0 = coordinates[i][j];

j++;

y0 = coordinates[i][j];

j++;

z0 = coordinates[i][j];

j++;

x = coordinates[i][j];

j++;

y = coordinates[i][j];

j++;

z = coordinates[i][j];

j++;

radlenth = coordinates[i][j];

j++;

actual\_res = result(x0, y0, z0, x, y, z, radlenth);

if (expected\_res == 1) {

expected\_res\_str = "Точка принадлежит сфере. ";

}

if (expected\_res == 0) {

expected\_res\_str = "Точка не принадлежит сфере. ";

}

if (actual\_res == 1) {

actual\_res\_str = "Точка принадлежит сфере. ";

}

if (actual\_res == 0) {

actual\_res\_str = "Точка не принадлежит сфере. ";

}

if (actual\_res == expected\_res) {

count++;

}

else if (actual\_res != expected\_res) {

cout << "Тест №" << i + 1 << " провалился." << endl;

cout << "Ожидался результат: " << expected\_res\_str << endl;

cout << "Полученный результат: " << actual\_res\_str << endl;

cout << endl;

}

i++;

j = 0;

}

if (count == count\_of\_tests) {

cout << "Тестирование прошло успешно." << endl;

cout << endl;

}

else {

cout << "Тестирование провалилось." << endl;

cout << endl;

}

}

[Конец test.cpp --- ]

[Начало test.h --- ]

#pragma once

#include "functions.h"

int result(double x0, double y0, double z0, double x, double y, double z, double radlenth);

void test();

[Конец test.h --- ]