

МИНОБРНАУКИ РОССИИ

федеральное государственное бюджетное образовательное учреждение высшего образования

«Санкт-Петербургский государственный технологический институт (технический университет)»

УГНС 09.00.00 Информатика и вычислительная техника

Направление подготовки 09.03.01 Информатика и вычислительная

техника

Направленность (профиль) Системы автоматизированного проектирования

Форма обучения очная

Факультет Информационных технологий и управления

Кафедра Систем автоматизированного проектирования и управления

Учебная дисциплина Программирование

Курс I Группа 404

Исполнитель: обучающийся

Отчёт по лабораторной работе № 2 Вариант № 22

группы 404 Рухлова Ксения Алексеевна

(дата, подпись)

Проверили: Корниенко Иван Григорьевич

(дата, подпись) Федин Алексей Константинович

Санкт-Петербург 2021

**Содержание**

[1 Постановка задачи 3](#_Toc72445045)

[2. Исходные данные 3](#_Toc72445046)

[3. Особые ситуации 3](#_Toc72445047)

[4. Математические методы и алгоритмы решения задач 3](#_Toc72445048)

[5. Форматы представления данных 4](#_Toc72445049)

[6. Структура программы 5](#_Toc72445050)

[7. Описание хода выполнения лабораторной работы 5](#_Toc72445051)

[8. Блок-схема алгоритма решения задачи 6](#_Toc72445052)

[9. Результаты работы программы 7](#_Toc72445053)

[10. Исходный текст программы 9](#_Toc72445054)

# 1 Постановка задачи

В трехмерном пространстве задан массив точек (тройками значений X, Y,Z) и сфера (центр и радиус). Напишите программу, выводящую точки (их координаты), которые попадают в заданную пользователем сферу.

# 2. Исходные данные

В качестве исходных данных программа использует количество точек (размер массива), координаты точек, радиус сферы и координаты центра сферы. Исходные данные могут быть введены пользователем, случайно сгенерированы или введены из файла (сначала количество точек, дальше координаты точек по *x, y, z*, затем радиус сферы и координаты центра сферы).

# 3. Особые ситуации

Необходимо рассмотреть следующие особые ситуации:

– Если пользователь ввел количество точек больше или равно 0, пользователю предлагается ввести значение количества точек заново.

– Если пользователь ввел радиус сферы больше или равно 0, пользователю предлагается ввести значение количества точек заново.

– Если пользователь выбрал ввод данных из файла, в котором записаны некорректные исходные данные, то их считывание невозможно, в таком случае пользователю предлагается ввести название файла и путь к нему заново.

– Если пользователь выбрал запись данных или результата в файл, который уже существует, то в таком случае пользователю предлагается либо ввести новое название файла и путь к нему, либо перезаписать данный файл.

# 4. Математические методы и алгоритмы решения задач

Согласно постановке задачи программа должна работать с точками и сферой в пространстве. Для того, чтобы точка попадала в сферу, необходимо чтобы выполнялось неравенство (1).

(1)

Где– координаты точки,

– координаты центра сферы,

– радиус сферы.

Анализируя массив, мы подставляем точки в данное неравенство, и если оно верно, то точка находится внутри сферы или на ее поверхности.

У нас также есть дополнительный булевый массив, в котором если условие выполняется, то ячейка с этим индексом точки равна единице, в противном случае – нулю. После по этим ячейкам выводится результат программы.

# 5. Форматы представления данных

Таблица 1 – Переменные, используемые в программе

|  |  |  |
| --- | --- | --- |
| **Имя** | **Тип** | **Описание** |
| input | int | Выбор варианта ввода данных |
| check | bool | Проверка корректности ввода |
| end | bool | Повтор программы |
| Path | string | Ввод имени файла и пути к нему |
| size | int | Обозначает размер массива (количество точек) |
| mode | int | Режим чтения/ записи файла |
| radius | int | Радиус сферы |
| output\_file | fstream | Чтение из файла |
| result\_file | ofstream | Запись в файл |

Также используются следующие константы

Таблица 2 – Константы, используемые в программе

|  |  |  |
| --- | --- | --- |
| **Имя** | **Тип** | **Описание** |
| MIN\_SIZE  MAX\_SIZE  MIN\_POINT  MAX\_POINT  MIN\_RADIUS  MAX\_RADIUS | const int | Определяет границы случайной генерации |

Для задания координат точки на плоскости используется структура *Point*, у которой 3 переменные *double x, double y, double z*. Также используется 3 класса: *arr\_point* – для создания массива точек, *res\_points\_class* – для создания массива попадания (булевый), *output\_file\_points* – для создания массива считывания данных из файла.

# 6. Структура программы

В ходе выполнения работы было принято решение разбить программу на модули:

Таблица 3 – Функции, составляющие программу

|  |  |  |
| --- | --- | --- |
| **Имя модуля** | **Функции, входящие в состав модуля** | **Описание** |
| Main | main | Основной модуль |
| Interface | get\_double, get\_int, get\_point, size\_arr, output\_point, greetings,  input\_types,show\_or\_try, create\_file,end\_program, output\_arr,res\_output\_arr, saving\_results,save\_data | Ввод и вывод исходных данных, подключение файла, общие функции |
| Points\_array | check\_inclusion, check | Вычисление результатов |

# 7. Описание хода выполнения лабораторной работы

1. Сначала для явного преобразования использовалась конвертация *C-style* (с помощью оператора ()), что могло привести к ошибке. Было принято решение заменить ее на оператор *static\_cast.*

2. Радиус и размер массива не может быть меньше или равно нулю, поэтому каждый ввод данных (из файла или с клавиатуры) обрабатывается на это условие.

3. Программа после запуска выдавала одни и те же результаты, хотя в коде использовался вызов функции *rand*, возвращающей случайное число. После изучения справочной системы выяснилось, что необходимо использовать функцию *srand* для начальной инициализации генератора случайных чисел. После этого программа стала работать правильно.

4. Так как количество данных в файле больше, чем просто координаты точек, было принято решение создать дополнительный класс массива, который хранит все данные, а потом по индексу присваивать нужные нам элементы.

5. В программе использовались глобальные переменные. Чтобы не было никаких ошибок, мы заменили их на локальные в качестве аргументов к функциям.

# 8. Блок-схема алгоритма решения задачи

На рисунке ниже предоставлена блок-схема алгоритма решения задачи.

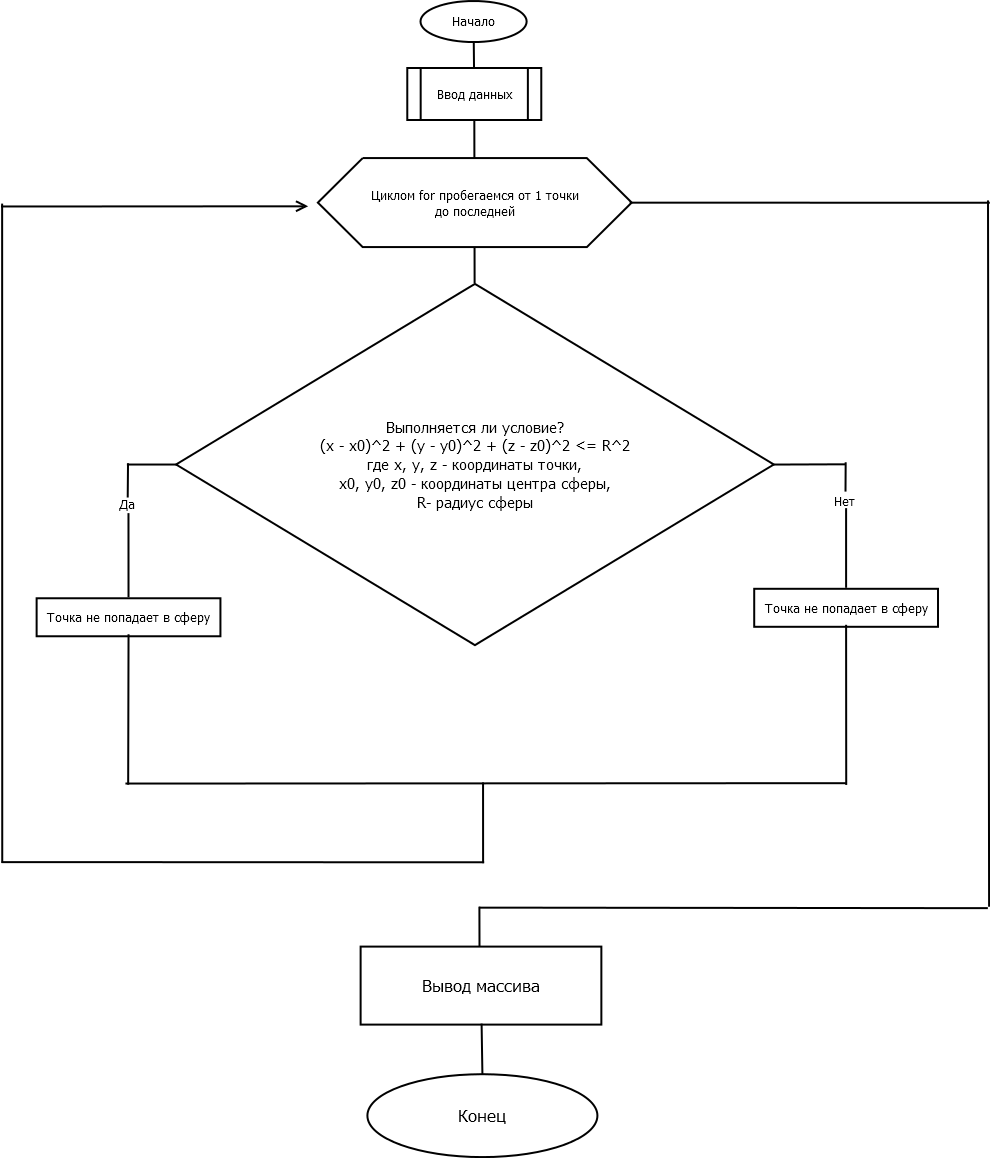


Рисунок 1– Блок-схема алгоритма решения задачи

# 9. Результаты работы программы

В результате вычислений программа выводит координаты точек пересечения прямоугольника и отрезка и их количество, или же сообщение что таких точек не найдено, либо их бесконечное множество. Ниже, на рисунках 2 - 6 приведены примеры результатов работы программы:

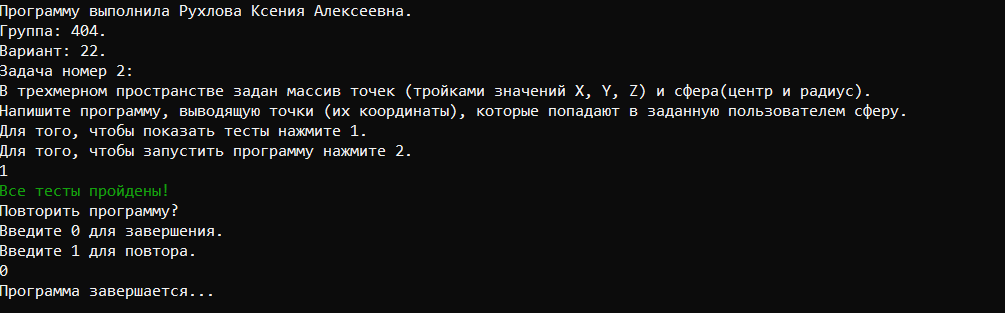


Рисунок 2 – Экранная копия результата работы разработанной программы

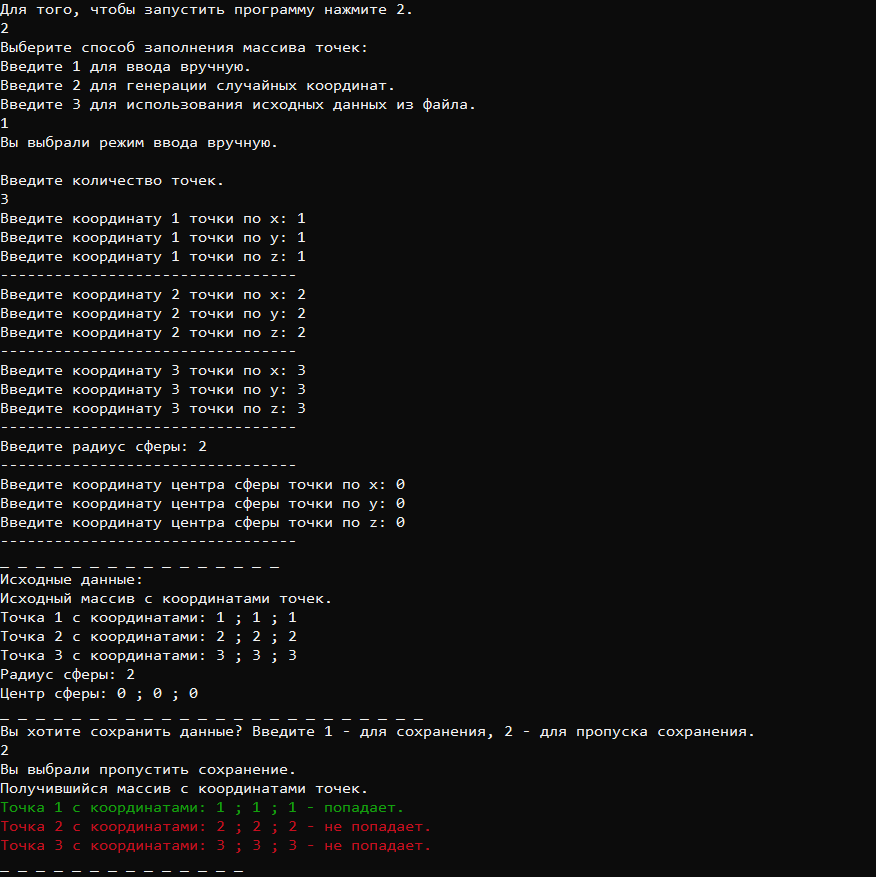


Рисунок 3 – Экранная копия результата работы разработанной программы

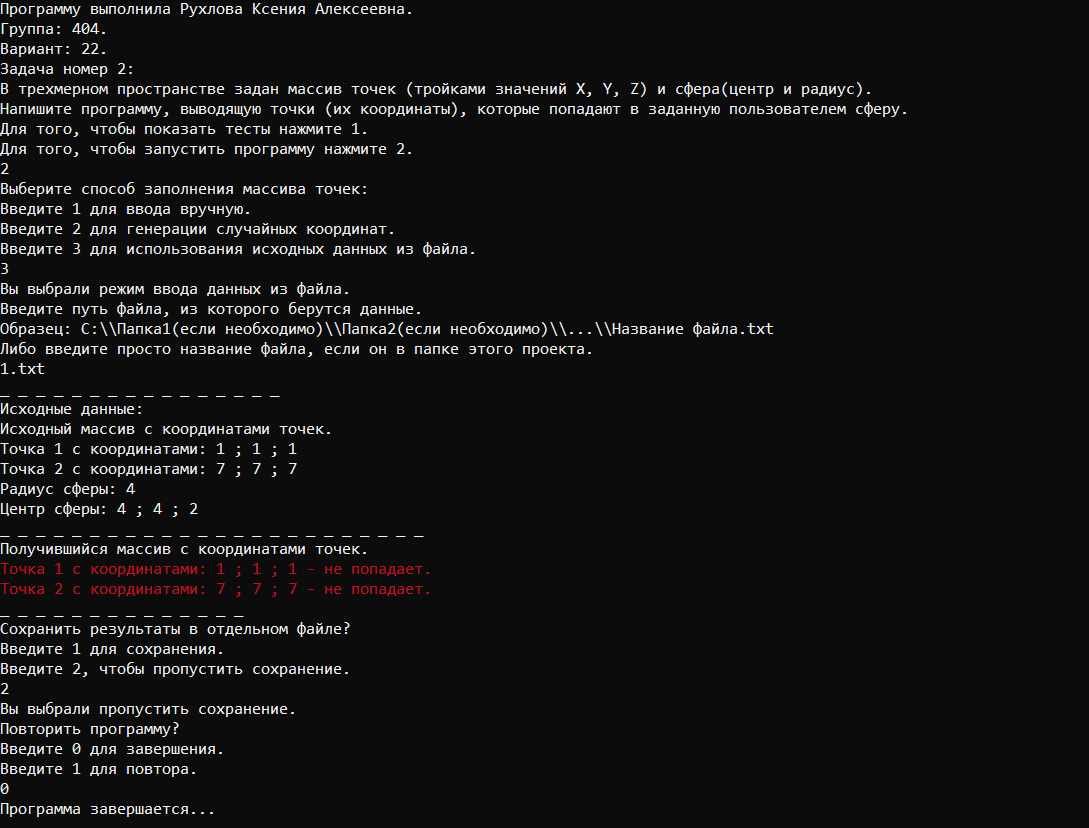


Рисунок 4 – Экранная копия результата работы разработанной программы

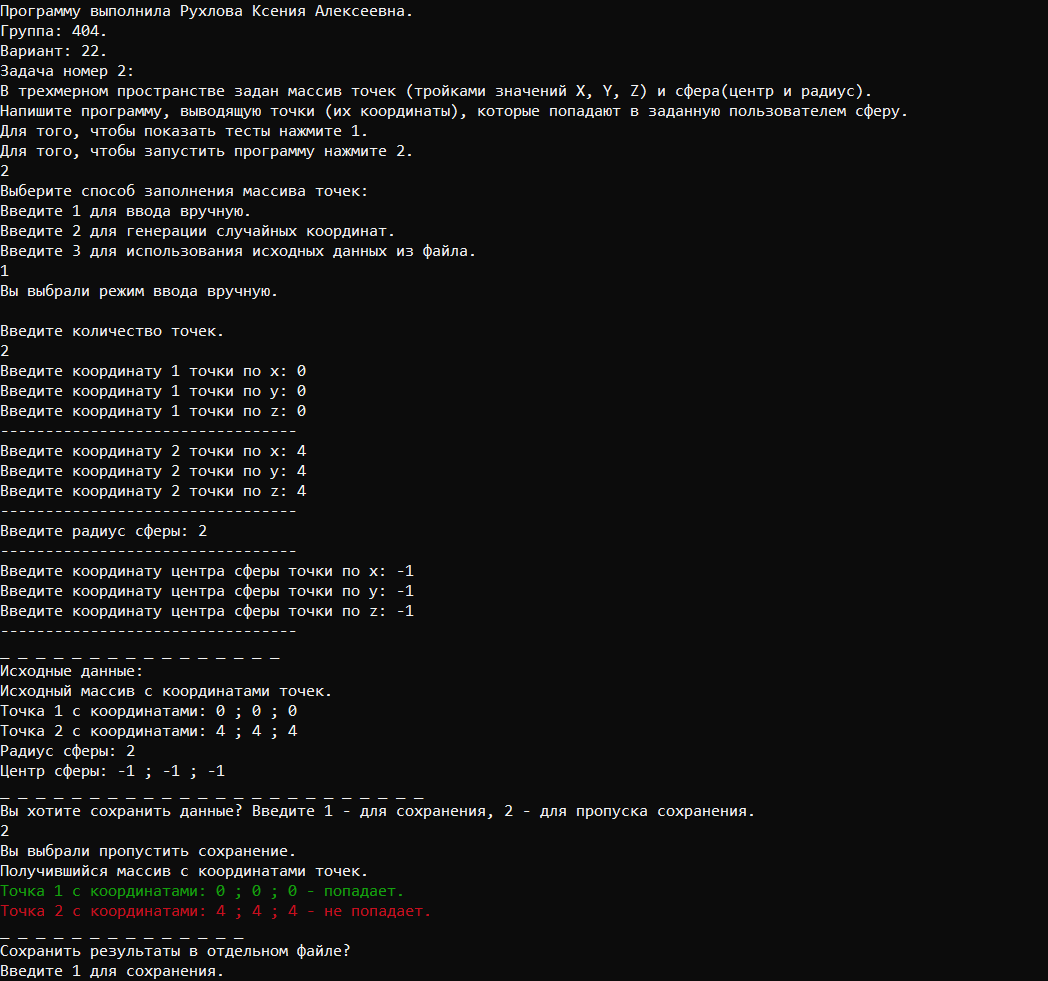


Рисунок 5 – Экранная копия результата работы, разработанной программ

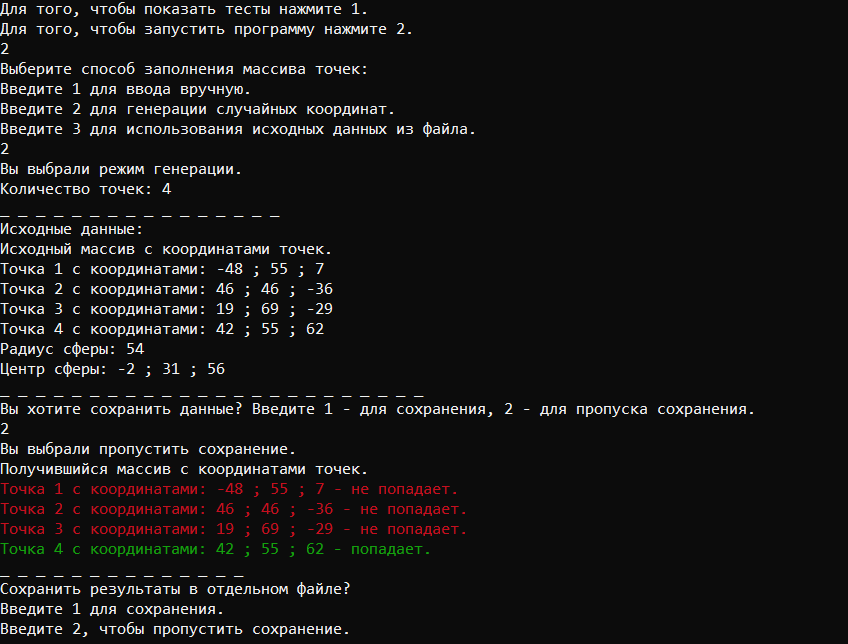


Рисунок 6 – Экранная копия результата работы, разработанной программ

# 10. Исходный текст программы

[Начало программы ---]

// Лабораторная работа № 2.

// Использование языка С++ для использования массивов

// Нахождение точек, попадающих в сферу

// Студентка группы 404, Рухлова Ксения Алексеевна. 2021 год

[Начало модуля main ---]

//Main.cpp

#include <iostream>

#include <ctime>

#include <Windows.h>

#include "Points\_array.h"

#include "Interface.h"

#include "Test.h"

using namespace std;

int main()

{

HANDLE handle;

handle = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleTextAttribute(handle, FOREGROUND\_RED | FOREGROUND\_GREEN | FOREGROUND\_BLUE | FOREGROUND\_INTENSITY);

setlocale(LC\_ALL, "Russian");

greetings();

bool end = false;

Points\_class point\_arr;

res\_points\_class res\_point\_arr;

int size = 0, mode = 0, radius = 0;

Point center;

srand (static\_cast<unsigned char>(time(NULL)));

do {

if (show\_or\_try()) run\_tests();

else {

input\_types(point\_arr, res\_point\_arr, size, mode, radius, center);

output\_arr(point\_arr, radius, center);

if (mode != Actions::FILE\_INPUT)

save\_data(point\_arr, radius, center, res\_point\_arr, size);

check\_inclusion(point\_arr, radius, center, res\_point\_arr);

res\_output\_arr(point\_arr, res\_point\_arr, handle);

saving\_results(point\_arr, res\_point\_arr);

}

end = end\_program();

} while (end != true);

}

[--- Конец модуля Main]

[Начало модуля Interface ---]

//Interface.h

#pragma once

#include <Windows.h>

class res\_points\_class;

class Points\_class;

struct Point;

class output\_file\_points {

private:

double\* file\_arr\_point;

int file\_arr\_size;

public:

double& operator[] (const int index) {

return file\_arr\_point[index];

}

int get\_size() {

return file\_arr\_size;

}

void set\_size(int size) {

file\_arr\_size = size;

file\_arr\_point = new double[file\_arr\_size];

}

~output\_file\_points() {

delete[] file\_arr\_point;

}

};

enum Actions {

END, CONTINUE,

MANUAL = 1, GENERATE, FILE\_INPUT,

YES = 1, NO,

QUANTITY\_COORDINATES, SIZE\_BYTE = 1,

NOT\_FALL = 0, FALL,

DATA, RESULT, READ\_POINTS,

TEST = 1, TRY,

REWRITE = 0, REPEAT,

CENTER\_POINT = -1

};

int get\_int();

double get\_double();

void greetings();

Point get\_point();

void input\_types(Points\_class& arr\_point, res\_points\_class& res\_arr\_point, int& size, int& mode, int& radius, Point& center);

bool show\_or\_try();

bool end\_program();

void size\_arr(int& size, Points\_class& point\_arr, res\_points\_class& res\_point\_arr);

void output\_point(Point point);

void output\_arr(Points\_class& point\_arr, int radius, Point center);

void res\_output\_arr(Points\_class& point\_arr, res\_points\_class& res\_point\_arr, HANDLE handle);

void create\_file(int mode, Points\_class& arr\_point, res\_points\_class& res\_arr\_point, int& size, int& radius, Point& center);

void saving\_results(Points\_class& point\_arr, res\_points\_class& res\_point\_arr);

void save\_data(Points\_class& point\_arr, int radius, Point center, res\_points\_class& res\_t, int size);

//Interface.cpp

#include <iostream>

#include <fstream>

#include <string>

#include <limits>

#include <filesystem>

#include <Windows.h>

#include "Interface.h"

#include "Points\_array.h"

using namespace std;

using namespace filesystem;

double get\_double() {

double input = 0;

cin >> input;

while (cin.fail()) {

cin.clear();

cin.ignore(INT\_MAX, '\n');

cout << "Введите число." << endl;

cin >> input;

}

return input;

}

int get\_int() {

int input = 0;

cin >> input;

while (cin.fail()) {

cin.clear();

cin.ignore(INT\_MAX, '\n');

cout << "Введите число." << endl;

cin >> input;

}

return input;

}

Point get\_point(const int index) {

Point point;

string name\_point;

if (index == CENTER\_POINT) name\_point = "центра сферы";

else name\_point = to\_string(index + 1);

cout << "Введите координату " << name\_point << " точки по x: ";

point.x = get\_double();

cout << "Введите координату " << name\_point << " точки по y: ";

point.y = get\_double();

cout << "Введите координату " << name\_point << " точки по z: ";

point.z = get\_double();

cout << "---------------------------------" << endl;

return point;

}

void user\_input(int& size, Points\_class& arr\_point, int& radius, Point& center, res\_points\_class& res\_point\_arr) {

cout << "Введите количество точек." << endl;

while (true) {

size = get\_int();

if (size <= 0) {

cout << "Количесвто точек должно быть больше нуля." << endl;

}

else break;

}

arr\_point.set\_size(size);

res\_point\_arr.set\_size(size);

for (int index = 0; index < size; index++) {

Point new\_point = get\_point(index);

arr\_point[index] = new\_point;

}

cout << "Введите радиус сферы: ";

while (true) {

radius = get\_int();

if (radius <= 0) {

cout << "Радиус должен быть больше нуля." << endl;

}

else break;

}

cout << "---------------------------------" << endl;

center = get\_point(CENTER\_POINT);

}

void generation\_input(int& size, Points\_class& arr\_point, int& radius, Point& center, res\_points\_class& res\_point\_arr) {

const int MIN\_SIZE = 1;

const int MAX\_SIZE = 5;

const int MIN\_POINT = -50;

const int MAX\_POINT = 100;

const int MIN\_RADIUS = 50;

const int MAX\_RADIUS = 100;

size = rand() % MAX\_SIZE + MIN\_SIZE;

cout << "Количество точек: " << size << endl;

arr\_point.set\_size(size);

res\_point\_arr.set\_size(size);

for (int index = 0; index < arr\_point.get\_size(); index++) {

arr\_point[index].x = (static\_cast <double> (rand() % (MAX\_POINT - MIN\_POINT)) + MIN\_POINT);

arr\_point[index].y = (static\_cast <double>(rand() % (MAX\_POINT - MIN\_POINT)) + MIN\_POINT);

arr\_point[index].z = (static\_cast <double>(rand() % (MAX\_POINT - MIN\_POINT)) + MIN\_POINT);

}

radius = ((rand() % (MAX\_RADIUS - MIN\_RADIUS)) + MIN\_RADIUS);

center.x = (static\_cast <double>(rand() % (MAX\_POINT - MIN\_POINT)) + MIN\_POINT);

center.y = (static\_cast <double>(rand() % (MAX\_POINT - MIN\_POINT)) + MIN\_POINT);

center.z = (static\_cast <double>(rand() % (MAX\_POINT - MIN\_POINT)) + MIN\_POINT);

}

void output\_point(Point point, const int index) {

cout << "Точка " << index << " с координатами: " << point.x << " ; " << point.y << " ; " << point.z;

}

void output\_arr(Points\_class& point\_arr, int radius, Point center) {

cout << "\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ " << endl;

cout << "Исходные данные: " << endl;

cout << "Исходный массив с координатами точек." << endl;

for (int index = 0; index < point\_arr.get\_size(); index++) {

output\_point(point\_arr[index], index + 1);

cout << endl;

}

cout << "Радиус сферы: " << radius << endl;

cout << "Центр сферы: " << center.x << " ; " << center.y << " ; " << center.z << endl;

cout << "\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_" << endl;

}

void res\_output\_arr(Points\_class& point\_arr, res\_points\_class& res\_point\_arr, HANDLE handle) {

cout << "Получившийся массив с координатами точек." << endl;

for (int index = 0; index < res\_point\_arr.get\_size(); index++) {

if (res\_point\_arr[index] == FALL) {

SetConsoleTextAttribute(handle, FOREGROUND\_GREEN);

output\_point(point\_arr[index], index + 1);

cout << " - попадает." << endl;

}

else {

SetConsoleTextAttribute(handle, FOREGROUND\_RED);

output\_point(point\_arr[index], index + 1);

cout << " - не попадает." << endl;

}

}

SetConsoleTextAttribute(handle, FOREGROUND\_RED | FOREGROUND\_GREEN | FOREGROUND\_BLUE | FOREGROUND\_INTENSITY);

cout << "\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_" << endl;

}

bool show\_or\_try(void) {

cout << "Для того, чтобы показать тесты нажмите 1." << endl;

cout << "Для того, чтобы запустить программу нажмите 2." << endl;

int mode = 0;

while (!((mode == TEST) || (mode == TRY))) {

mode = get\_int();

switch (mode) {

case TEST: return true;

break;

case TRY: return false;

break;

default: cout << "Введите либо 1, либо 2." << endl;

}

}

return true;

}

void create\_file(int mode, Points\_class& point\_arr, res\_points\_class& res\_point\_arr, int& size, int& radius, Point& center) {

if (mode == READ\_POINTS) {

bool repeat = true;

do {

repeat = false;

cout << "Введите путь файла, из которого берутся данные." << endl

<< "Образец: C:\\\\Папка1(если необходимо)\\\\Папка2(если необходимо)\\\\...\\\\Название файла.txt " << endl

<< "Либо введите просто название файла, если он в папке этого проекта."

<< endl;

string name = "";

cin >> name;

fstream output\_file(name);

output\_file >> size;

output\_file\_points file\_points;

int AMOUNT\_DATA = QUANTITY\_COORDINATES \* (size - CENTER\_POINT) + SIZE\_BYTE;

file\_points.set\_size(AMOUNT\_DATA);

for (int index = 0; index < AMOUNT\_DATA; index++) {

if (!(output\_file >> file\_points[index])) {

repeat = true;

}

}

if (repeat) cout << "Некорректные данные." << endl;

else {

if (size <= 0) {

cout << "Количество точек должно быть больше нуля." << endl;

repeat = true;

}

if (!(repeat)) {

for (int index = 0; index < AMOUNT\_DATA; index++) {

output\_file >> file\_points[index];

}

int file\_index = 0;

point\_arr.set\_size(size);

res\_point\_arr.set\_size(size);

for (int index = 0; index < size; index++) {

point\_arr[index].x = file\_points[file\_index];

point\_arr[index].y = file\_points[file\_index + 1];

point\_arr[index].z = file\_points[file\_index + 2];

file\_index += QUANTITY\_COORDINATES;

}

radius = (int)(file\_points[size\*QUANTITY\_COORDINATES]);

if (radius <= 0) {

cout << "Радиус сферы должен быть больше нуля." << endl;

repeat = true;

}

center.x = file\_points[AMOUNT\_DATA - 3];

center.y = file\_points[AMOUNT\_DATA - 2];

center.z = file\_points[AMOUNT\_DATA - 1];

output\_file.close();

}

}

} while (repeat);

}

else {

bool repeat = true;

do {

repeat = false;

cout << "Введите путь файла, в который запишутся данные." << endl << "Образец: C:\\\\Папка1(если необходимо)\\\\"

"Папка2(если необходимо)\\\\...\\\\Название файла.txt " << endl << "Либо введите просто название файла, тогда он будет в папке этого проекта."

<< endl;

string name = "";

cin >> name;

if (ifstream(name)) {

int rewrite = 0;

cout << "Файл уже существует." << endl;

cout << "Введите 0 для того, чтобы перезаписать существующий файл." << endl;

cout << "Введите 1 для того, чтобы повторить ввод." << endl;

while (true) {

rewrite = get\_int();

if (rewrite == REWRITE) {

cout << "Вы выбрали опцию перезаписать файл." << endl;

break;

}

else if (rewrite == REPEAT) {

cout << "Вы выбрали опцию повторить ввод." << endl;

repeat = true;

break;

}

else cout << "Введите 0, либо 1" << endl;

}

}

if (!(repeat)) {

ofstream result\_file(name, ofstream::app);

error\_code ec;

if (!is\_regular\_file(name, ec)) {

cout << "Адрес содержит недопустимые значения. Повторите ввод." << endl;

ec.clear();

repeat = true;

}

if (!(repeat)) {

result\_file.clear();

switch (mode) {

case DATA:

result\_file << size << " ";

for (int index = 0; index <= size; index++) {

if (index != size)

result\_file << point\_arr[index].x << " " << point\_arr[index].y << " " << point\_arr[index].z << " ";

else result\_file << radius << " " << center.x << " " << center.y << " " << center.z << " ";

}

break;

case RESULT:

result\_file

<< "Результат программы. " << endl

<< "Получившийся массив с координатами точек:" << endl;

for (int index = 0; index < res\_point\_arr.get\_size(); index++) {

if (res\_point\_arr[index] == FALL) {

result\_file << "Точка " << index + 1 << "с координатами: " << point\_arr[index].x << " ; " << point\_arr[index].y << " ; " << point\_arr[index].z

<< " - попадает." << endl;

}

else {

result\_file << "Точка " << index + 1 << " с координатами: " << point\_arr[index].x << " ; " << point\_arr[index].y << " ; " << point\_arr[index].z

<< " - не попадает." << endl;

}

}

cout << "\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_" << endl;

break;

}

result\_file.close();

}

}

} while (repeat);

}

}

void greetings() {

cout << "Программу выполнила Рухлова Ксения Алексеевна." << endl << "Группа: 404." << endl << "Вариант: 22." << endl <<

"Задача номер 2:" << endl <<

"В трехмерном пространстве задан массив точек (тройками значений X, Y, Z) и сфера(центр и радиус). " << endl

<< "Напишите программу, выводящую точки (их координаты), которые попадают в заданную пользователем сферу."

<< endl;

}

void input\_types(Points\_class& arr\_point, res\_points\_class& res\_arr\_point, int& size, int& mode, int& radius, Point& center) {

cout << "Выберите способ заполнения массива точек:" << endl << "Введите 1 для ввода вручную." << endl

<< "Введите 2 для генерации случайных координат." << endl << "Введите 3 для использования исходных данных из файла." << endl;

bool repeat = true;

do {

repeat = false;

int menu\_type = get\_int();

mode = menu\_type;

switch (menu\_type) {

case MANUAL:

cout << "Вы выбрали режим ввода вручную." << endl;

cout << endl;

user\_input(size, arr\_point, radius, center, res\_arr\_point);

break;

case GENERATE:

cout << "Вы выбрали режим генерации." << endl;

generation\_input(size, arr\_point, radius, center, res\_arr\_point);

break;

case FILE\_INPUT:

cout << "Вы выбрали режим ввода данных из файла." << endl;

create\_file(READ\_POINTS, arr\_point, res\_arr\_point, size, radius, center);

break;

default:

cout << "Введите либо 1, либо 2, либо 3." << endl;

repeat = true;

}

} while (repeat);

}

bool end\_program() {

cout << "Повторить программу? " << endl << "Введите 0 для завершения." << endl << "Введите 1 для повтора." << endl;

bool repeat = true;

do {

repeat = false;

int end = get\_int();

switch (end) {

case END:

cout << "Программа завершается..." << endl;

break;

case CONTINUE:

cout << "Вы выбрали повторить программу: " << endl;

break;

default:

cout << "Введите либо 0, либо 1." << endl;

repeat = true;

}

return !(end);

} while (repeat);

}

void saving\_results(Points\_class& point\_arr, res\_points\_class& res\_point\_arr) {

cout << "Сохрaнить результаты в отдельном файле?" << endl << "Введите 1 для сохранения." << endl << "Введите 2, чтобы пропустить сохранение." << endl;

int size\_t = 0, radius\_t = 0;

Point center\_t;

while (true) {

int save = get\_int();

if (save == YES) {

create\_file(RESULT, point\_arr, res\_point\_arr, size\_t, radius\_t, center\_t);

break;

}

else if (save == NO) {

cout << "Вы выбрали пропустить сохранение." << endl;

break;

}

else cout << "Введите либо 1, либо 2." << endl;

}

}

void save\_data(Points\_class& point\_arr, int radius, Point center, res\_points\_class& res\_t, int size) {

cout << "Вы хотите сохранить данные? Введите 1 - для сохранения, 2 - для пропуска сохранения." << endl;

while (true) {

int save = get\_int();

if (save == YES) {

create\_file(DATA, point\_arr, res\_t, size, radius, center);

cout << "Данные сохранены." << endl;

break;

}

else if (save == NO) {

cout << "Вы выбрали пропустить сохранение." << endl;

break;

}

else cout << "Введите либо 1, либо 2." << endl;

}

}

[--- Конец модуля Input]

[ Начало модуля Points\_array ---]

// Points\_array.h

#pragma once

#include "Interface.h"

struct Point

{

double x, y, z;

};

class Points\_class {

private:

Point\* arr\_point;

int arr\_size;

public:

Point& operator[] (const int index) {

return arr\_point[index];

}

int get\_size() {

return arr\_size;

}

void set\_size(int size) {

arr\_size = size;

arr\_point = new Point[arr\_size];

}

~Points\_class() {

delete[] arr\_point;

}

};

class res\_points\_class {

private:

bool\* res\_arr\_point;

int res\_arr\_size;

public:

bool& operator[] (const int index) {

return res\_arr\_point[index];

}

int get\_size() {

return res\_arr\_size;

}

void set\_size(int size) {

res\_arr\_size = size;

res\_arr\_point = new bool[res\_arr\_size];

}

~res\_points\_class() {

delete[] res\_arr\_point;

}

};

void check(Point point, int radius, Point center, res\_points\_class& res\_arr, const int index);

void check\_inclusion(Points\_class& point\_arr, int radius, Point center, res\_points\_class& res\_arr);

// Points\_array.cpp

#include <math.h>

#include "Points\_array.h"

void check(Point point, int radius, Point center, res\_points\_class& res\_arr, const int index) {

if ((pow((point.x - center.x), 2) + pow((point.y - center.y), 2) + pow((point.z - center.z), 2)) <= pow(radius, 2))

res\_arr[index] = FALL;

else res\_arr[index] = NOT\_FALL;

}

void check\_inclusion(Points\_class& point\_arr, int radius, Point center, res\_points\_class& res\_arr) {

for (int index = 0; index < point\_arr.get\_size(); index++) {

check(point\_arr[index], radius, center, res\_arr, index);

}

}

[--- Конец модуля Points\_array]

[Начало модуля Test ---]

// Test.h

#pragma once

#include <Windows.h>

#include <iostream>

using namespace std;

void run\_tests(void);

// Test.cpp

#include "Test.h"

#include "Points\_array.h"

#include "Interface.h"

#include <string>

bool Test1(Point& center, Points\_class& point\_arr, res\_points\_class& res\_point\_arr, res\_points\_class& expected\_res\_point\_arr) {

int quantity = 0, radius = 0;

quantity = 5;

point\_arr.set\_size(quantity);

res\_point\_arr.set\_size(quantity);

expected\_res\_point\_arr.set\_size(quantity);

point\_arr[0].x = 0;

point\_arr[0].y = 0;

point\_arr[0].z = 0;

point\_arr[1].x = 1;

point\_arr[1].y = 1;

point\_arr[1].z = 1;

point\_arr[2].x = 2;

point\_arr[2].y = 2;

point\_arr[2].z = 2;

point\_arr[3].x = 3;

point\_arr[3].y = 3;

point\_arr[3].z = 3;

point\_arr[4].x = 1.5;

point\_arr[4].y = 0;

point\_arr[4].z = 0;

radius = 3;

center.x = 0;

center.y = 0;

center.z = 0;

expected\_res\_point\_arr[0] = 1;

expected\_res\_point\_arr[1] = 1;

expected\_res\_point\_arr[2] = 0;

expected\_res\_point\_arr[3] = 0;

expected\_res\_point\_arr[4] = 1;

int check = 0;

HANDLE handle = GetStdHandle(STD\_OUTPUT\_HANDLE);

check\_inclusion(point\_arr, radius, center, res\_point\_arr);

for (int i = 0; i < res\_point\_arr.get\_size(); i++) {

if (expected\_res\_point\_arr[i] == res\_point\_arr[i])

check++;

}

if (check == res\_point\_arr.get\_size()) {

return true;

}

else {

SetConsoleTextAttribute(handle, FOREGROUND\_RED);

cout << "Тест 1 не пройден!" << endl;

output\_arr(point\_arr, radius, center);

string expected\_result, result;

SetConsoleTextAttribute(handle, FOREGROUND\_RED | FOREGROUND\_GREEN | FOREGROUND\_BLUE | FOREGROUND\_INTENSITY);

for (int i = 0; i < res\_point\_arr.get\_size(); i++) {

if (expected\_res\_point\_arr[i] == 1) expected\_result = " попадает.";

else expected\_result = " не попадает.";

cout << "Ожидалось, что точка " << i + 1 << expected\_result << endl;

if (res\_point\_arr[i] == 1) result = " попадает.";

else result = " не попадает.";

cout << "По прохождению теста точка " << i + 1 << result << endl;

}

return false;

}

}

bool Test2(Point& center, Points\_class& point\_arr, res\_points\_class& res\_point\_arr, res\_points\_class& expected\_res\_point\_arr) {

int quantity = 5, radius = 0;

point\_arr.set\_size(quantity);

res\_point\_arr.set\_size(quantity);

expected\_res\_point\_arr.set\_size(quantity);

point\_arr[0].x = 0;

point\_arr[0].y = 0;

point\_arr[0].z = 0;

point\_arr[1].x = 1;

point\_arr[1].y = 1;

point\_arr[1].z = 1;

point\_arr[2].x = 2;

point\_arr[2].y = 2;

point\_arr[2].z = 2;

point\_arr[3].x = 3;

point\_arr[3].y = 3;

point\_arr[3].z = 3;

point\_arr[4].x = 1.5;

point\_arr[4].y = 0;

point\_arr[4].z = 0;

radius = 4;

center.x = 0;

center.y = 0;

center.z = 0;

expected\_res\_point\_arr[0] = 1;

expected\_res\_point\_arr[1] = 1;

expected\_res\_point\_arr[2] = 1;

expected\_res\_point\_arr[3] = 0;

expected\_res\_point\_arr[4] = 1;

int check = 0;

HANDLE handle = GetStdHandle(STD\_OUTPUT\_HANDLE);

check\_inclusion(point\_arr, radius, center, res\_point\_arr);

for (int i = 0; i < res\_point\_arr.get\_size(); i++) {

if (expected\_res\_point\_arr[i] == res\_point\_arr[i])

check++;

}

if (check == res\_point\_arr.get\_size()) {

return true;

}

else {

SetConsoleTextAttribute(handle, FOREGROUND\_RED);

cout << "Тест 2 не пройден!" << endl;

output\_arr(point\_arr, radius, center);

string expected\_result, result;

SetConsoleTextAttribute(handle, FOREGROUND\_RED | FOREGROUND\_GREEN | FOREGROUND\_BLUE | FOREGROUND\_INTENSITY);

for (int i = 0; i < res\_point\_arr.get\_size(); i++) {

if (expected\_res\_point\_arr[i] == 1) expected\_result = " попадает.";

else expected\_result = " не попадает.";

cout << "Ожидалось, что точка " << i + 1 << expected\_result << endl;

if (res\_point\_arr[i] == 1) result = " попадает.";

else result = " не попадает.";

cout << "По прохождению теста точка " << i + 1 << result << endl;

}

return false;

}

}

bool Test3(Point& center, Points\_class& point\_arr, res\_points\_class& res\_point\_arr, res\_points\_class& expected\_res\_point\_arr) {

int quantity = 3, radius = 2;

point\_arr.set\_size(quantity);

res\_point\_arr.set\_size(quantity);

expected\_res\_point\_arr.set\_size(quantity);

point\_arr[0].x = 0;

point\_arr[0].y = 0;

point\_arr[0].z = 0;

point\_arr[1].x = -1;

point\_arr[1].y = -1;

point\_arr[1].z = -1;

point\_arr[2].x = 2;

point\_arr[2].y = 2;

point\_arr[2].z = 2;

center.x = 1;

center.y = 1;

center.z = 1;

expected\_res\_point\_arr[0] = 1;

expected\_res\_point\_arr[1] = 0;

expected\_res\_point\_arr[2] = 1;

int check = 0;

HANDLE handle = GetStdHandle(STD\_OUTPUT\_HANDLE);

check\_inclusion(point\_arr, radius, center, res\_point\_arr);

for (int i = 0; i < res\_point\_arr.get\_size(); i++) {

if (expected\_res\_point\_arr[i] == res\_point\_arr[i])

check++;

}

if (check == res\_point\_arr.get\_size()) {

return true;

}

else {

SetConsoleTextAttribute(handle, FOREGROUND\_RED);

cout << "Тест 3 не пройден!" << endl;

output\_arr(point\_arr, radius, center);

string expected\_result, result;

SetConsoleTextAttribute(handle, FOREGROUND\_RED | FOREGROUND\_GREEN | FOREGROUND\_BLUE | FOREGROUND\_INTENSITY);

for (int i = 0; i < res\_point\_arr.get\_size(); i++) {

if (expected\_res\_point\_arr[i] == 1) expected\_result = " попадает.";

else expected\_result = " не попадает.";

cout << "Ожидалось, что точка " << i + 1 << expected\_result << endl;

if (res\_point\_arr[i] == 1) result = " попадает.";

else result = " не попадает.";

cout << "По прохождению теста точка " << i + 1 << result << endl;

}

return false;

}

}

bool Test4(Point& center, Points\_class& point\_arr, res\_points\_class& res\_point\_arr, res\_points\_class& expected\_res\_point\_arr) {

int quantity = 3, radius = 2;

point\_arr.set\_size(quantity);

res\_point\_arr.set\_size(quantity);

expected\_res\_point\_arr.set\_size(quantity);

point\_arr[0].x = 0;

point\_arr[0].y = 0;

point\_arr[0].z = 0;

point\_arr[1].x = -1;

point\_arr[1].y = -1;

point\_arr[1].z = -1;

point\_arr[2].x = 2;

point\_arr[2].y = 2;

point\_arr[2].z = 2;

center.x = 100;

center.y = 100;

center.z = 100;

expected\_res\_point\_arr[0] = 0;

expected\_res\_point\_arr[1] = 0;

expected\_res\_point\_arr[2] = 0;

int check = 0;

HANDLE handle = GetStdHandle(STD\_OUTPUT\_HANDLE);

check\_inclusion(point\_arr, radius, center, res\_point\_arr);

for (int i = 0; i < res\_point\_arr.get\_size(); i++) {

if (expected\_res\_point\_arr[i] == res\_point\_arr[i])

check++;

}

if (check == res\_point\_arr.get\_size()) {

return true;

}

else {

SetConsoleTextAttribute(handle, FOREGROUND\_RED);

cout << "Тест 4 не пройден!" << endl;

output\_arr(point\_arr, radius, center);

string expected\_result, result;

SetConsoleTextAttribute(handle, FOREGROUND\_RED | FOREGROUND\_GREEN | FOREGROUND\_BLUE | FOREGROUND\_INTENSITY);

for (int i = 0; i < res\_point\_arr.get\_size(); i++) {

if (expected\_res\_point\_arr[i] == 1) expected\_result = " попадает.";

else expected\_result = " не попадает.";

cout << "Ожидалось, что точка " << i + 1 << expected\_result << endl;

if (res\_point\_arr[i] == 1) result = " попадает.";

else result = " не попадает.";

cout << "По прохождению теста точка " << i + 1 << result << endl;

}

return false;

}

}

void run\_tests(void) {

HANDLE handle = GetStdHandle(STD\_OUTPUT\_HANDLE);

Point center;

Points\_class point\_arr;

res\_points\_class res\_point\_arr, expected\_res\_point\_arr;

if (Test1(center, point\_arr, res\_point\_arr, expected\_res\_point\_arr) &&

Test2(center, point\_arr, res\_point\_arr, expected\_res\_point\_arr) &&

Test3(center, point\_arr, res\_point\_arr, expected\_res\_point\_arr) &&

Test4(center, point\_arr, res\_point\_arr, expected\_res\_point\_arr)) {

SetConsoleTextAttribute(handle, FOREGROUND\_GREEN);

cout << "Все тесты пройдены!" << endl;

SetConsoleTextAttribute(handle, FOREGROUND\_RED | FOREGROUND\_GREEN | FOREGROUND\_BLUE | FOREGROUND\_INTENSITY);

}

}

[--- Конец модуля Test]

[--- Конец программы]