Минобрнауки России

федеральное государственное бюджетное образовательное учреждение высшего образования

«Санкт-Петербургский государственный технологический институт

(технический университет)»

­­­­

**Дисциплина «Программирование»**

Отчёт по лабораторной работе №2.

Использование массивов

Формирование матрицы из точек, заданных пользователем, которые находятся в сфере, заданной пользователем.

Преподаватели: Корниенко Иван Григорьевич.

Федин Алексей Константинович

Исполнил студент 405 группы: Емонаков Константин Юрьевич

Санкт-Петербург

2021

**Постановка задачи**

Пользователем задается количество точек в трехмерном пространстве и их координаты. Также, пользователь вводит координаты центра сферы и ее радиус. Формируется результирующая матрица, содержащая координаты точек, которые лежат в заданной сфере. Затем выводится на экран.

При выполнении лабораторной работы необходимо выделить массив требуемого размера, запросив его у пользователя при запуске программы или считав из файла. В программе должны быть предусмотрены три варианта заполнения исходного массива: пользователем с клавиатуры, из файла и случайными числами. В работе должна присутствовать возможность сохранения введенных пользователем с клавиатуры данных, сохранения результата работы программы и модульного тестирования, также должны быть использованы методы вывода на экран с использованием различных цветов шрифта.

**Исходные данные**

В качестве исходных данных программа использует: пользовательский ввод через консоль количества точек n, элементов матрицы; текстовые файлы формата «txt», координат центра сферы и ее радиус.

Координаты каждой из точек, количество точек (n) и координат центра (x, y, z) сферы и ее радиус (radius) являются целочисленным типом данных (тип int в C++).

**Особые ситуации**

Необходимо рассмотреть следующие особые ситуации:

- Некорректный ввод количества точек n (не может быть меньше 1), элементов матрицы и элементов искомого массива или пункта пользовательского меню.

- Отсутствие ожидаемых программой файлов на чтение или содержание некорректных данных внутри существующих файлов.

- Запись работы программы в уже существующий файл или создание недопустимого файла, а также проверка атрибутов существующего файла (Атрибут «Только для чтения»).

**Математические методы и алгоритмы решения задач**

В поставленной задаче формирования новой матрицы, содержащей координаты точек, которые лежат в сфере, реализуется при помощи цикла, в котором проверяется главное условие – (где – координаты точек, а – координаты сферы).

**Форматы представления данных**

Формат внешних файлов, из которых производится взятие данных – строго текстовый формат txt. Внутри читаемого txt файла для корректной работы должны присутствовать координаты точек в строго данном порядке. Все значения должны быть разделены символом пробела.

Формат файлов для записи выбирается пользователем, но для корректного чтения рекомендуется использовать только текстовый формат txt, иначе чтение сторонним программным обеспечением будет некорректным.

Таблица 1 – Основные переменные программы

|  |  |  |
| --- | --- | --- |
| **Имя** | **Тип** | **Описание** |
| n | int | Количество точек в матрице |
| NUMBEROFCOORDINATES | int | Количество координат, которые имеет каждая точка в матрице |
| radius | int | Радиус сферы |
| x, y, z | int | Координаты центра сферы |
| arrayofdots | int\*\* | Матрица с заданными точками |
| Arrayofdotsincircle | int\*\* | Матрица с точками, лежащими в сфере |

**Структура программы**

Программа разбита на 11 модулей: cpp и header файлы задания данных сферы и определения лежат ли точки в сфере, cpp и header файлы работы с внешними файлами, cpp и header файлы задания массивов, cpp и header файлы основного интерфейса программы, cpp и header файлы вывода массивов и модульных тестов и основной main.cpp файл для запуска программы

Основная последовательность работы программы – цикл ожидания решения пользователя. Внутри цикла программа ожидает пользовательский выбор режима работы: ввод данных вручную, считывание данных из файла или случайное заполнение матрицы. Если данные для работы были получены корректно, запускается алгоритм создания результирующей матрицы. На экран выводится результирующая матрица. Далее пользователь выбирает способ сохранения входных и выходных данных, после чего цикл повторяется до тех пор, пока пользователь не закроет программу через пункт основного меню или с помощью средств операционной системы.

Модули circle.h и circle.cpp содержат алгоритмы решения поставленной задачи, реализуемый в функциях:

Таблица 2 – Функции пользовательского интерфейса

|  |  |
| --- | --- |
| **Имя** | **Описание** |
| void CircleData(int& radius, int& x, int& y, int& z, int n) | Возвращает тип void. Принимает параметры сферы и количество точек в массиве. Запрашивает у пользователя радиус и координаты центра сферы. |
| int HowMany(int\*\* arrayofdots, int n, int radius, int x, int y, int z) | Возвращает тип int. Принимает матрицу, размер массива, радиус и координаты сферы . Считает какое количество точек, содержащихся в матрице, лежат в сфере. |
| void IsInCircle(int\*\* arrayofdots, int\*\* arrayofdotsincircle, int n, int counting, int radius, int x, int y, int z) | Возвращает тип void. Принимает матрицы 1 и 2, их размеры, а также радиус и координаты центра сферы. Записывает во 2ю матрицу точки, содержащиеся в 1й матрице, которые лежат в заданной сфере. |

Модули CreateArray.h и CreateArray.cpp cодержат алгоритмы создания и заполения матриц, реализуемые в функциях.

Таблица 3 – Функции алгоритма

|  |  |
| --- | --- |
| **Имя** | **Описание** |
| void CreateArray(int\*\* arrayofdots, int n) | Возвращает тип void. Принимает матрицу и ее размер. Создает динамическую матрицу. |
| void FillArray(int\*\* arrayofdots, int n) | Возвращает тип void. Принимает матрицу и ее размер. Заполняет матрицу одним из трех способов на выбор (с клавиатуры, из файла, случайными числами). |

Модули File.h и File.cpp содержат интерфейс работы с файлами, методы чтения данных из файла и записи результатов работы программы в файл:

Таблица 4 – Функции работы с файлами

|  |  |
| --- | --- |
| **Имя** | **Описание** |
| void ReadFile(int\*\* arrayofdots, int\*\* arrayofdotsincircle, int radius, int x, int y, int z, int n, int counting) | Возвращает тип void. Принимает две квадратные матрицы, их размеры, а также радиус и координаты сферы. Позволяет пользователю сохранить данные в файл. |

Модули Input.h и Input.cpp содержат интерфейсы меню и пользовательского ввода, реализуемые в функциях:

Таблица 5 – Функции работы с файлами

|  |  |
| --- | --- |
| **Имя** | **Описание** |
| void Iam() | Возвращает тип void и не принимает аргументы. Выводит данные о программе. |
| bool Menu() | Возвращает тип bool и не принимает аргументы. Позволяет пользователю выбрать способ ввода данных, выйти из программы после ее окончания, вызвать функцию Iam |
| int input() | Возвращает тип int и не принимает аргументы. Проверяет корректность пользовательского ввода и задает количество точек. |

Модули structure.h и structure.cpp содержат алгоритмы вывода массивов, а также юнит тесты, реализуемые в функциях:

Таблица 6 – Функции работы с файлами

|  |  |
| --- | --- |
| **Имя** | **Описание** |
| string ToLower(string line) | Возвращает тип string. Принимает строку – путь к файлу. Приводит все введенные буквы к строчному ввиду. |
| void ShowFirstArray(int\*\* arrayofdots, int n) | Возвращает тип void. Принимает исходную матрицу и ее размер. Выводит матрицу в консоль. |
| void ShowSecondArray(int\*\* arrayofdotsincircle, int counting) | Возвращает тип void. Принимает результирующую матрицу и ее размер. Выводит матрицу в консоль красным цветом. |
| bool IsTxt(string pathinput) | Возвращает тип bool. Принимает строку – пользовательский ввод файла. Проверяет, есть ли в конце ввода “.txt”. |
| bool IsInputOk(string pathinput) | Возвращает тип bool. Принимает строку – пользовательский ввод. Проверяет является ли пользовательский ввод корректным. |
| void IsTestOk() | Возвращает тип void и не принимает аргументы. Юнит тесты. |

**Блок-схемы алгоритмов программы**



Рисунок 1 - Блок-схема IsInCircle - основного алгоритма программы

**Описание хода выполнения лабораторной работы**

В ходе лабораторной работы было создано решение (Solution) интегрированной среде разработки Microsoft Visual Studio C++ 2017. В нём был создан проект.

Программа была разбита на 5 условные части: модуль работы с файлами, модуль создания матрицы, модуль пользовательского интерфейса, модуль обработки пользовательского ввода и основной модуль с циклом работы программы. Далее все модули, кроме основного, были разбиты на две части: заголовочный файл и файл с описанием функций и методов пользовательских классов.

После написания основного цикла работы программы, были созданы функции пользовательского ввода, чтения данных из файла. Далее был создан алгоритм для создания результирующей матрицы. После был создан алгоритм печати полученных данных на экран и использованием цветного вывода в консоль и печати во внешний файл.

В ходе работы над проектом были учтены и обработаны ошибки ввода некорректных данных, а также некорректное чтение файлов, создание файлов с запрещённым в OS Windows именем и проверку атрибутов существующего файла.

Также в код программы были включены модульные тесты, предназначенные для тестирования основного модуля программы – определения пересечения отрезков и поиска координат точки их пересечения. Модульные тесты запускаются автоматически при выборе пользователем соответствующего пункта меню. Данные для тестирования берутся из заранее подготовленных файлов. В случае удачного прохождения тестов на экран выводится сообщение об успешном их выполнении, иначе - сообщение о неудаче в ходе тестирования.

**Результаты работы программы**

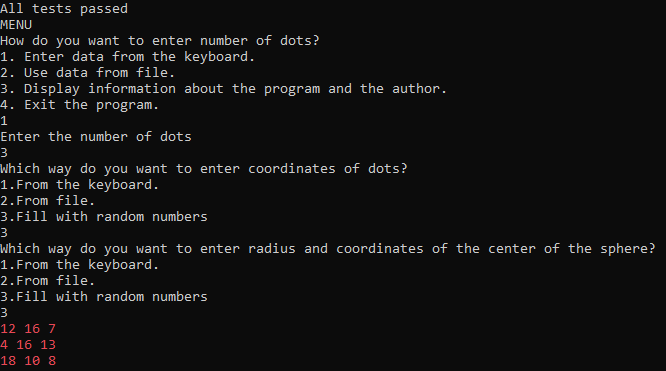
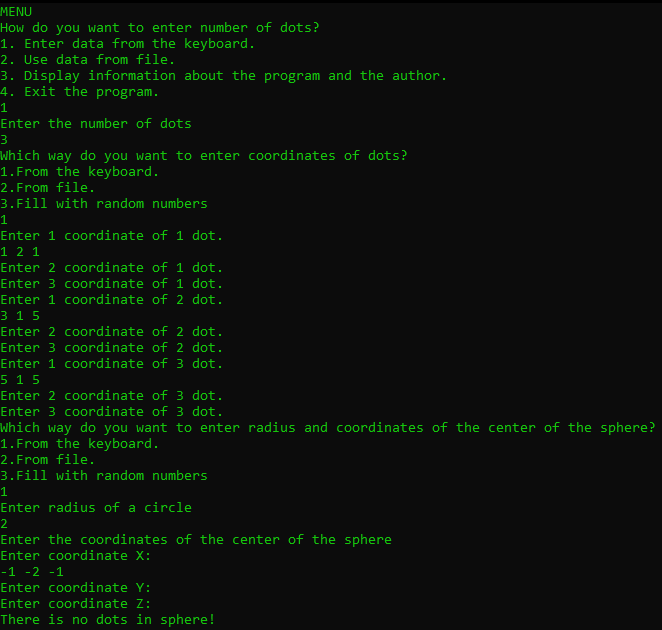
В результате работы программа выводит ответ в виде одной измененной матрицы.

Рисунок 2 – Решение при вводе данных случайным образом

Рисунок 3 - Решение при вводе данных из файла

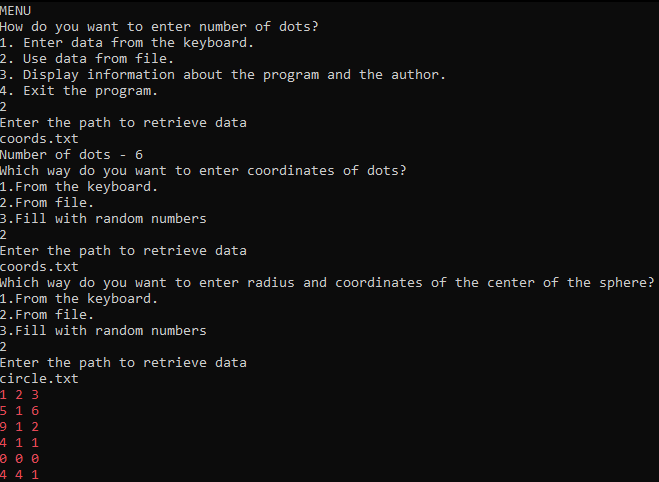


Рисунок 4 – Решение при вводе данных с клавиатуры

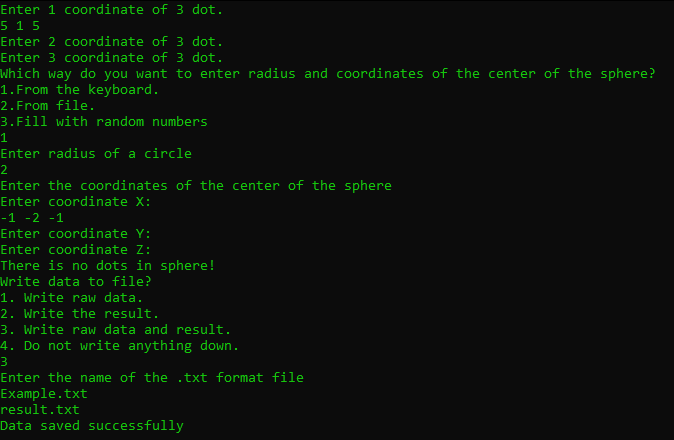


Рисунок 5 – Сохранение файлов в файл

**Исходный текст программы**

[Начало программы ---]

[Начало circle.h --- ]

#pragma once

void CircleData(int& radius, int& x, int& y, int& z, int n);

int HowMany(int\*\* arrayofdots, int n, int radius, int x, int y, int z);

void IsInCircle(int\*\* arrayofdots, int\*\* arrayofdotsincircle, int n, int counting, int radius, int x, int y, int z);

[Конец circle.h --- ]

[Начало circle.cpp --- ]

#include <fstream>

#include <iostream>

#include <ctime>

#include "Input.h"

#include "Circle.h"

#include <cmath>

#include "CreateArray.h"

#include "Structure.h"

void CircleData(int& radius, int& x, int& y, int& z, int n)

{

ifstream ifoutcircle;

int answer;

cout << "Which way do you want to enter radius and coordinates of the center of the sphere?" << endl;

cout << "1.From the keyboard." << endl;

cout << "2.From file." << endl;

cout << "3.Fill with random numbers" << endl;

while (true)

{

cin >> answer;

if (cin.fail() || (cin.peek() != '\n'))

{

cout << "Please enter a valid value!" << endl;

cin.clear();

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

}

else

break;

}

switch (answer)

{

case First:

{

while (true)

{

cout << "Enter radius of a circle" << endl;

cin >> radius;

if (cin.fail() || radius <= 0 || (cin.peek() != '\n'))

{

cout << "Please enter a valid value! \n";

cin.clear();

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

}

else

break;

}

cout << "Enter the coordinates of the center of the sphere" << endl;

while (true)

{

cout << "Enter coordinate X: " << endl;

cin >> x;

if (cin.fail() || (cin.peek() != '\n'))

{

cout << "Please enter a valid value! \n";

cin.clear();

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

}

else

break;

}

while (true)

{

cout << "Enter coordinate Y: " << endl;

cin >> y;

if (cin.fail() || (cin.peek() != '\n'))

{

cout << "Please enter a valid value! \n";

cin.clear();

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

}

else

break;

}

while (true)

{

cout << "Enter coordinate Z: " << endl;

cin >> z;

if (cin.fail() || (cin.peek() != '\n'))

{

cout << "Please enter a valid value! \n";

cin.clear();

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

}

else

break;

}

break;

}

case Second:

{

while (true)

{

string linetocircle;

cout << "Enter the path to retrieve data" << endl;

cin >> linetocircle;

ifoutcircle.open(linetocircle, ios::in);

if (!IsTxt(linetocircle)) {

cout << "Dont fotget to put .txt at the end" << endl;

cin.clear();

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

}

else if (ifoutcircle.is\_open())

{

int howmanysimbols = 0;

int ch;

while (ifoutcircle >> ch)

{

howmanysimbols++;

}

ifoutcircle.close();

ifoutcircle.open(linetocircle);

if ((howmanysimbols != numberofcirclecoords) && (howmanysimbols != 1 + n\*NUMBEROFCOORDINATES + numberofcirclecoords))

{

cout << "Incorrect data in the file! Upload new file." << endl;

cout << "Array contains invalid amount of numbers" << endl;

ifoutcircle.close();

}

else if (howmanysimbols == numberofcirclecoords)

{

ifoutcircle >> radius;

ifoutcircle >> x;

ifoutcircle >> y;

ifoutcircle >> z;

ifoutcircle.close();

howmanysimbols = 0;

break;

}

else if (howmanysimbols == 1 + (n \* NUMBEROFCOORDINATES) + numberofcirclecoords)

{

int time;

ifoutcircle >> time;

for (int i = 0; i < n; i++)

{

for (int j = 0; j < NUMBEROFCOORDINATES; j++)

{

ifoutcircle >> time;

}

}

ifoutcircle >> radius;

ifoutcircle >> x;

ifoutcircle >> y;

ifoutcircle >> z;

ifoutcircle.close();

howmanysimbols = 0;

break;

}

}

else if (!ifoutcircle.is\_open())

{

cout << "The file cannot be opened!" << endl;

}

}

break;

}

case Third:

{

srand((unsigned int)time(NULL));

radius = rand() % random;

x = rand() % random;

y = rand() % random;

z = rand() % random;

break;

}

default:

{

cout << "Invalid enter." << endl;

CircleData(radius, x, y, z, n);

break;

}

}

}

int HowMany(int\*\* arrayofdots, int n, int radius, int x, int y, int z) {

int count = 0;

for (int i = 0; i < n; i++)

{

if ((pow((arrayofdots[i][0] - x), 2) + pow((arrayofdots[i][1] - y), 2) + pow((arrayofdots[i][2] - z), 2)) <= (pow(radius, 2))) {

count++;

}

}

return count;

}

void IsInCircle(int\*\* arrayofdots, int\*\* arrayofdotsincircle, int n, int counting, int radius, int x, int y, int z) {

int j = 0;

for (int i = 0; i < n; i++)

{

if ((pow((arrayofdots[i][0] - x), 2) + pow((arrayofdots[i][1] - y), 2) + pow((arrayofdots[i][2] - z), 2)) <= (pow(radius, 2)))

{

while (j < counting)

{

arrayofdotsincircle[j][0] = arrayofdots[i][0];

arrayofdotsincircle[j][1] = arrayofdots[i][1];

arrayofdotsincircle[j][2] = arrayofdots[i][2];

j+=1;

break;

}

}

}

}

[Конец circle.cpp --- ]

[Начало CreateArray.h --- ]

#pragma once

enum answercreate { First = 1, Second, Third };

void CreateArray(int\*\* arrayofdots, int n);

void FillArray(int\*\* arrayofdots, int n, int previouschoice);

const int NUMBEROFCOORDINATES = 3; //Количество координат точки

[Конец CreateArray.h --- ]

[Начало CreateArray.cpp --- ]

#include "CreateArray.h"

#include "Input.h"

#include <fstream>

#include <iostream>

#include <ctime>

#include "Structure.h"

using namespace std;

ifstream ifoutcreatearray;

void CreateArray(int\*\* array, int n)//Создаем массив

{

for (int i = 0; i < n; i++)

{

array[i] = new int[NUMBEROFCOORDINATES];

}

}

void FillArray(int\*\* arrayofdots, int n, int previouschoice)//Заполняем

{

int answered = NULL;

cout << "Which way do you want to enter coordinates of dots?" << endl;

cout << "1.From the keyboard." << endl;

cout << "2.Fill with random numbers" << endl;

int element = NULL;

if (previouschoice == 1) {

while (true)

{

cin >> answered;

if (cin.fail() || (cin.peek() != '\n') || answered != First && answered != Second)

{

cout << "Please enter a valid value!" << endl;

cin.clear();

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

}

else

break;

}

}

else

{

cout << "3.From file." << endl;

while (true)

{

cin >> answered;

if (cin.fail() || (cin.peek() != '\n') || answered != First && answered != Second && answered != Third)

{

cout << "Please enter a valid value!" << endl;

cin.clear();

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

}

else

break;

}

}

switch (answered)

{

case First:

{

for (int i = 0; i < n; i++)

{

for (int j = 0; j < NUMBEROFCOORDINATES; j++)

{

cout << "Enter "<< j+1 << " coordinate of " << i+1 << " dot." << endl;

while (true)

{

cin >> element;

if (cin.fail() || (cin.peek() != '\n'))

{

cout << "Invalid value entered!" << endl;

cout << "Please enter a number in betwen of -2147483648 and 2147483647" << endl;

cin.clear();

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

}

else

{

arrayofdots[i][j] = (int) element;

break;

}

}

}

}

break;

}

case Second:

{

srand((unsigned int)time(NULL));

for (int i = 0; i < n; i++)

{

for (int j = 0; j < NUMBEROFCOORDINATES; j++)

{

arrayofdots[i][j] = rand() % random;

}

}

break;

}

case Third:

{

while (true)

{

string line;

cout << "Enter the path to retrieve data" << endl;

cin >> line;

ifoutcreatearray.open(line, ios::in);

if (!IsTxt(line)) {

cout << "Dont fotget to put .txt at the end" << endl;

cin.clear();

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

}

else if (ifoutcreatearray.is\_open() && IsInputOk(line))

{

int counercreate = 0;

int ch;

while (ifoutcreatearray >> ch)

{

counercreate++;

}

ifoutcreatearray.close();

ifoutcreatearray.open(line);

if (counercreate != n \* NUMBEROFCOORDINATES && counercreate != n \* NUMBEROFCOORDINATES + 1 && counercreate != n \* NUMBEROFCOORDINATES + 5)

{

cout << "Incorrect data in the file! Upload new file." << endl;

cout << "Array contains invalid amount of numbers" << endl;

ifoutcreatearray.close();

}

else if (counercreate == n \* NUMBEROFCOORDINATES)

{

for (int i = 0; i < n; i++)

{

for (int j = 0; j < NUMBEROFCOORDINATES; j++)

{

ifoutcreatearray >> arrayofdots[i][j];

}

}

ifoutcreatearray.close();

counercreate = 0;

break;

}

else if (counercreate == (n \* NUMBEROFCOORDINATES) + 1 || counercreate == (n \* NUMBEROFCOORDINATES) + 5)

{

int time;

ifoutcreatearray >> time;

for (int i = 0; i < n; i++)

{

for (int j = 0; j < NUMBEROFCOORDINATES; j++)

{

ifoutcreatearray >> arrayofdots[i][j];

}

}

ifoutcreatearray.close();

counercreate = 0;

break;

}

}

else if (!ifoutcreatearray.is\_open() || !IsInputOk(line))

{

cout << "The file cannot be opened!" << endl;

}

}

}

}

}

[Конец CreateArray.cpp --- ]

[Начало File.h]

#pragma once

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

void ReadFile(int\*\* arrayofdots, int\*\* arrayofdotsincircle, int radius, int x, int y, int z, int n, int counting);

enum question { SaveInput = 1, SaveResalt, SaveBoth, Nothing };

enum file { Yes = 1, No };

[Конец File.h --- ]

[Начало File.cpp ---]

#define \_USE\_MATH\_DEFINES

#include <iostream>

#include <fstream>

#include <string>

#include "Input.h"

#include "CreateArray.h"

#include "File.h"

#include "Structure.h"

#include <math.h>

#include <cmath>

#include "Circle.h"

#include <filesystem>

using namespace std::experimental::filesystem::v1;

using namespace std;

void ReadFile(int\*\* arrayofdots, int\*\* arrayofdotsincircle, int radius, int x, int y, int z, int n, int counting)

{

ofstream fout;

int answering;

bool checkfile;

string pathinput;

cout << "Write data to file?" << endl;

cout << "1. Write only raw data. " << endl;

cout << "2. Write only the result." << endl;

cout << "3. Write raw data and result." << endl;

cout << "4. Do not write anything down." << endl;

while (true)

{

cin >> answering;

if (cin.fail() || (cin.peek() != '\n') || (answering != SaveInput && answering != SaveResalt && answering != SaveBoth && answering != Nothing))

{

cout << "Please enter a valid value!" << endl;

cin.clear();

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

}

else

break;

}

switch (answering)

{

case SaveInput:

{

while (!fout.is\_open())

{

checkfile = true;

cout << "Enter the name of the .txt format file" << endl;

cout << "Example.txt" << endl;

cin >> pathinput;

if(!IsInputOk(pathinput))

{

cout << "This name is reserved by Windows. Enter another" << endl;

cin.clear();

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

}

else if (!IsTxt(pathinput)) {

cout << "Dont fotget to put .txt at the end" << endl;

cin.clear();

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

}

else

{

if (exists(pathinput))

{

cout << "A file with this name exists. Do you want to overwrite the file? " << endl;

cout << "1. Yes." << endl;

cout << "2. No." << endl;

while (true)

{

cin >> answering;

if (cin.fail())

{

cout << "Enter the correct value!";

cin.clear();

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

}

else

break;

}

switch (answering)

{

case Yes:

{

break;

}

case No:

{

checkfile = false;

break;

}

}

}

if (checkfile != false)

{

fout.open(pathinput);

}

if (fout.is\_open() && checkfile != false)

{

fout << n << endl;

for (int i = 0; i < n; i++)

{

for (int j = 0; j < NUMBEROFCOORDINATES; j++)

{

fout << arrayofdots[i][j] << " ";

}

fout << "\n";

}

fout << radius << "\n" << x << " " << y << " " << z << " " << endl;

cout << "Data saved successfully" << endl;

fout.close();

break;

}

else if (checkfile != false)

{

cout << "The file has the wrong extension, please try again" << endl;

cout << "The file must have the .txt extension" << endl;

fout.close();

}

}

}

break;

}

case SaveResalt:

{

while (!fout.is\_open())

{

checkfile = true;

cout << "Enter the name of the .txt format file" << endl;

cout << "Example.txt" << endl;

cin >> pathinput;

if (!IsInputOk(pathinput)) {

cout << "This name is reserved by Windows. Enter another" << endl;

cin.clear();

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

}

else if (!IsTxt(pathinput)) {

cout << "Dont fotget to put .txt at the end" << endl;

cin.clear();

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

}

else

{

if (exists(pathinput))

{

cout << "A file with this name exists. Do you want to overwrite the file? " << endl;

cout << "1. Yes." << endl;

cout << "2. No." << endl;

while (true)

{

cin >> answering;

if (cin.fail())

{

cout << "Please enter a valid value!";

cin.clear();

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

}

else

break;

}

switch (answering)

{

case Yes:

{

break;

}

case No:

{

checkfile = false;

break;

}

}

}

if (checkfile != false)

{

fout.open(pathinput);

}

if (fout.is\_open() && checkfile != false)

{

fout << "Dots in sphere:" << endl;

if (counting >= 1) {

for (int i = 0; i < counting; i++)

{

fout << "Dot number " << i + 1 << endl;

fout << arrayofdotsincircle[i][0] << " " << arrayofdotsincircle[i][1] << " " << arrayofdotsincircle[i][2] << endl;

}

}

else

cout << "There is No dots in sphere!" << endl;

cout << "Data saved successfully" << endl;

fout.close();

break;

}

else if (checkfile != false)

{

cout << "The file has the wrong extension, please try again" << endl;

cout << "The file must have the .txt extension" << endl;

fout.close();

}

}

}

break;

}

case SaveBoth:

{

while (true) {

cout << "How do you want save raw data and result?" << endl;

cout << "1. In one file." << endl;

cout << "2. In different files" << endl;

cin >> answering;

switch (answering)

{

case First:

{

while (!fout.is\_open())

{

checkfile = true;

cout << "Enter the name of the .txt format file" << endl;

cout << "Example.txt" << endl;

cin >> pathinput;

if (!IsInputOk(pathinput))

{

cout << "This name is reserved by Windows. Enter another" << endl;

cin.clear();

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

}

else if (!IsTxt(pathinput)) {

cout << "Dont fotget to put .txt at the end" << endl;

cin.clear();

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

}

else

{

if (exists(pathinput))

{

cout << "A file with this name exists. Do you want to overwrite the file? " << endl;

cout << "1. Yes." << endl;

cout << "2. No." << endl;

while (true)

{

cin >> answering;

if (cin.fail())

{

cout << "Please enter a valid value!";

cin.clear();

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

}

else

break;

}

switch (answering)

{

case Yes:

{

break;

}

case No:

{

checkfile = false;

break;

}

}

}

if (checkfile != false)

{

fout.open(pathinput);

}

if (fout.is\_open() && checkfile != false)

{

fout << "Radius and coordinates of circle:" << endl;

fout << radius << "\n" << x << " " << y << " " << z << " " << endl;

fout << "Coordintes of entered dots:" << endl;

for (int i = 0; i < n; i++)

{

fout << "Dot number " << i + 1 << endl;

for (int j = 0; j < NUMBEROFCOORDINATES; j++)

{

fout << arrayofdots[i][j] << " " << endl;

}

}

fout << "Dots in sphere:" << endl;

for (int i = 0; i < counting; i++)

{

fout << "Dot number " << i + 1 << endl;

fout << arrayofdotsincircle[i][0] << " " << arrayofdotsincircle[i][1] << " " << arrayofdotsincircle[i][2] << endl;

}

cout << "Data saved successfully" << endl;

fout.close();

break;

}

else if (checkfile != false)

{

cout << "The file has the wrong extension, please try again" << endl;

cout << "The file must have the .txt extension" << endl;

fout.close();

}

}

}

break;

}

case Second:

{

while (!fout.is\_open())

{

checkfile = true;

cout << "Enter the name of the .txt format file to save raw data." << endl;

cout << "Example.txt" << endl;

cin >> pathinput;

if (!IsInputOk(pathinput))

{

cout << "This name is reserved by Windows. Enter another" << endl;

cin.clear();

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

}

else if (!IsTxt(pathinput)) {

cout << "Dont fotget to put .txt at the end" << endl;

cin.clear();

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

}

else

{

if (exists(pathinput))

{

cout << "A file with this name exists. Do you want to overwrite the file? " << endl;

cout << "1. Yes." << endl;

cout << "2. No." << endl;

while (true)

{

cin >> answering;

if (cin.fail())

{

cout << "Enter the correct value!";

cin.clear();

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

}

else

break;

}

switch (answering)

{

case Yes:

{

break;

}

case No:

{

checkfile = false;

break;

}

}

}

if (checkfile != false)

{

fout.open(pathinput);

}

if (fout.is\_open() && checkfile != false)

{

fout << n << endl;

for (int i = 0; i < n; i++)

{

for (int j = 0; j < NUMBEROFCOORDINATES; j++)

{

fout << arrayofdots[i][j] << " ";

}

cout << "\n";

}

fout << radius << "\n" << x << " " << y << " " << z << " " << endl;

cout << "Data saved successfully" << endl;

fout.close();

break;

}

else if (checkfile != false)

{

cout << "The file has the wrong extension, please try again" << endl;

cout << "The file must have the .txt extension" << endl;

fout.close();

}

}

}

while (!fout.is\_open())

{

checkfile = true;

cout << "Enter the name of the .txt format file to save the result" << endl;

cout << "Example.txt" << endl;

cin >> pathinput;

if (!IsInputOk(pathinput)) {

cout << "This name is reserved by Windows. Enter another" << endl;

cin.clear();

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

}

else if (!IsTxt(pathinput)) {

cout << "Dont fotget to put .txt at the end" << endl;

cin.clear();

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

}

else

{

if (exists(pathinput))

{

cout << "A file with this name exists. Do you want to overwrite the file? " << endl;

cout << "1. Yes." << endl;

cout << "2. No." << endl;

while (true)

{

cin >> answering;

if (cin.fail())

{

cout << "Please enter a valid value!";

cin.clear();

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

}

else

break;

}

switch (answering)

{

case Yes:

{

break;

}

case No:

{

checkfile = false;

break;

}

}

}

if (checkfile != false)

{

fout.open(pathinput);

}

if (fout.is\_open() && checkfile != false)

{

fout << "Dots in sphere:" << endl;

if (counting >= 1) {

for (int i = 0; i < counting; i++)

{

fout << "Dot number " << i + 1 << endl;

fout << arrayofdotsincircle[i][0] << " " << arrayofdotsincircle[i][1] << " " << arrayofdotsincircle[i][2] << endl;

}

}

else

cout << "There is No dots in sphere!" << endl;

cout << "Data saved successfully" << endl;

fout.close();

break;

}

else if (checkfile != false)

{

cout << "The file has the wrong extension, please try again" << endl;

cout << "The file must have the .txt extension" << endl;

fout.close();

}

}

}

}

default:

{

cout << "Invalid enter." << endl;

cin.clear();

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

}

}

break;

}

}

case Nothing:

{

break;

}

default:

{

cout << "Invalid enter." << endl;

ReadFile(arrayofdots, arrayofdotsincircle, radius, x, y, z, n, counting);

break;

}

}

}

[Конец File.cpp --- ]

[Начало Input.h ---]

#pragma once

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

void ReadFile(int\*\* arrayofdots, int\*\* arrayofdotsincircle, int radius, int x, int y, int z, int n, int counting);

enum question { SaveInput = 1, SaveResalt, SaveBoth, Nothing };

enum file { Yes = 1, No };

[Конец Input.h --- ]

[Начало Input.cpp ---]

#include <iostream>

#include <fstream>

#include <string>

#include "Structure.h"

#include "File.h"

#include "Input.h"

#include "CreateArray.h"

#include "Circle.h"

using namespace std;

void Iam()

{

cout << "Laboratory work 2" << endl;

cout << "An array of points is given in 3D space and sphere(center and radius). Write a program that prints out dots(their coordinates) that fall within a user - defined sphere." << endl;

cout << "Student group 405, Emonakov Konstantin." << endl;

}

bool Menu()

{

int variant;

int n, counter = 0;

cout << "MENU" << endl;

cout << "How do you want to enter number of dots?" << endl;

cout << "1. Enter data from the keyboard." << endl;

cout << "2. Use data from file." << endl;

cout << "3. Display information about the program and the author." << endl;

cout << "4. Exit the program." << endl;

while (true)

{

cin >> variant;

if (cin.fail() || (cin.peek() != '\n') || (variant != Start && variant != UseFile && variant != Information && variant != Finish))

{

cout << "Please enter a valid value!" << endl;

cin.clear();

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

}

else

break;

}

switch ((int) variant)

{

case Start:

{

variant = 0;

return true;

}

case UseFile:

{

while (true)

{

ifstream ifout;

string line;

cout << "Enter the path to retrieve data" << endl;

cin >> line;

ifout.open(line, ios::in);

if (!IsTxt(line)) {

cout << "Dont fotget to put .txt at the end" << endl;

cin.clear();

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

}

else if (ifout.is\_open() && IsInputOk(line))

{

int ch;

while (ifout >> ch)

{

counter++;

}

ifout.close();

ifout.open(line);

ifout >> n;

if (counter == 0)

{

cout << "Incorrect data in the file! Upload new file." << endl;

}

else

{

int radius = 0, x = 0, y = 0, z = 0;

counter = 0;

cout << "Number of dots - " << n << endl;

ifout.close();

int\*\* arrayofdots = new int\*[n];

int previouschoice = 2;

CreateArray(arrayofdots, n);

FillArray(arrayofdots, n, previouschoice);

CircleData(radius, x, y, z, n);

int counting = HowMany(arrayofdots, n, radius, x, y, z);

int \*\*arrayofdotsincircle = new int \*[counting];

CreateArray(arrayofdotsincircle, counting);

IsInCircle(arrayofdots, arrayofdotsincircle, n, counting, radius, x, y, z);

ShowSecondArray(arrayofdotsincircle, counting);

ReadFile(arrayofdots, arrayofdotsincircle, radius, x, y, z, n, counting);

for (int i = 0; i < n; i++)

{

delete[] arrayofdots[i];

}

delete[] arrayofdots;

for (int i = 0; i < counting; i++)

{

delete[] arrayofdotsincircle[i];

}

delete[] arrayofdotsincircle;

bool checkMenuFile = Menu();

if (checkMenuFile == false)

return false;

else

return true;

}

}

if (!IsInputOk(line))

{

cout << "This file reserved by windows!" << endl;

}

else if (!ifout.is\_open())

{

cout << "The file cannot be opened!" << endl;

bool checkMenuFile = Menu();

if (checkMenuFile == false)

return false;

else

return true;

}

}

return false;

}

case Information:

{

Iam();

Menu();

return true;

}

case Finish:

{

cout << "The program has ended. Goodbye." << endl;

return false;

}

default:

{

cout << "Invalid value entered." << endl;

return false;

}

}

}

int input()

{

int n;

while (true)

{

cout << "Enter the number of dots" << endl;

cin >> n;

if (cin.fail() || n < 1 || (cin.peek() != '\n'))

{

cout << "Invalid value entered!" << endl;

cout << "Please enter a number in betwen of 1 and 2147483647" << endl;

cin.clear();

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

}

else

break;

}

return n;

}

[Конец Input.cpp --- ]

[Начало structure.h ---]

#pragma once

#include"Input.h"

void ShowSecondArray(int\*\* arrayofdotsincircle, int counting);

void ShowFirstArray(int\*\* arrayofdots, int n);

void IsTestOk();

bool IsInputOk(string pathinput);

bool IsTxt(string pathinput);

const int random = 20;

const int numberofcirclecoords = 4;

[Конец structure.h --- ]

[Начало structure.cpp ---]

#include <Windows.h>

#include <cstring>

#include "Structure.h"

#include <algorithm>

#include "Input.h"

#include <iostream>

#include "Circle.h"

#include "CreateArray.h"

using namespace std;

string ToLower(string line)

{

transform(line.begin(), line.end(), line.begin(), [](char c) {return static\_cast<char>(tolower(c)); });

return line;

}

void ShowFirstArray(int\*\* arrayofdots, int n)//Выводим на консоль первый массив

{

for (int i = 0; i < n; i++)

{

for (int j = 0; j < NUMBEROFCOORDINATES; j++)

{

cout << arrayofdots[i][j] << "\t";

}

cout << endl;

}

cout << endl;

}

void ShowSecondArray(int\*\* arrayofdotsincircle, int counting)//Выводим на консоль второй массив

{

SetConsoleTextAttribute(GetStdHandle(STD\_OUTPUT\_HANDLE), FOREGROUND\_RED | FOREGROUND\_INTENSITY);

if (counting >= 1) {

cout << "Dots in sphere:" << endl;

cout << "X Y Z" << endl;

for (int i = 0; i < counting; i++)

{

cout << arrayofdotsincircle[i][0] << " ";

cout << arrayofdotsincircle[i][1] << " ";

cout << arrayofdotsincircle[i][2] << " " << endl;

}

}

else

cout << "There is no dots in sphere!" << endl;

SetConsoleTextAttribute(GetStdHandle(STD\_OUTPUT\_HANDLE), FOREGROUND\_GREEN | FOREGROUND\_INTENSITY);

}

bool IsTxt(string pathinput) {

size\_t lenght = pathinput.size();

if ( lenght < 5) {

return false;

}

else {

if (pathinput[lenght-1] != 't' || pathinput[lenght -2] != 'x' || pathinput[lenght -3] != 't' || pathinput[lenght -4] != '.') {

return false;

}

else {

return true;

}

}

}

bool IsInputOk(string pathinput) {

pathinput = ToLower(pathinput);

const char\* a;

const char\* b;

const char\* c;

const char\* d;

const char\* e;

const char\* f;

const char\* g;

const char\* o;

a = strstr(pathinput.c\_str(), "con.txt");

b = strstr(pathinput.c\_str(), "prn.txt");

c = strstr(pathinput.c\_str(), "aux.txt");

d = strstr(pathinput.c\_str(), "nul.txt");

e = strstr(pathinput.c\_str(), ".con.txt");

f = strstr(pathinput.c\_str(), ".prn.txt");

g = strstr(pathinput.c\_str(), ".aux.txt");

o = strstr(pathinput.c\_str(), ".nul.txt");

if (e != NULL || f != NULL || g != NULL || o != NULL)

{

a = NULL;

b = NULL;

c = NULL;

d = NULL;

}

if (pathinput == "con.txt" || pathinput == "prn.txt" || pathinput == "aux.txt" ||

pathinput == "nul.txt" || pathinput == "com1.txt" || pathinput == "com2.txt" ||

pathinput == "com3.txt" || pathinput == "com4.txt" || pathinput == "com5.txt" ||

pathinput == "com6.txt" || pathinput == "com7.txt" || pathinput == "com8.txt" ||

pathinput == "com9.txt" || pathinput == "lpt1.txt" || pathinput == "lpt2.txt" ||

pathinput == "lpt3.txt" || pathinput == "lpt4.txt" || pathinput == "lpt5.txt" ||

pathinput == "lpt6.txt" || pathinput == "lpt7.txt" || pathinput == "lpt8.txt" ||

pathinput == "lpt9.txt" || pathinput == "con" || pathinput == "prn" || pathinput == "aux" ||

pathinput == "nul" || pathinput == "com1" || pathinput == "com2" ||

pathinput == "com3" || pathinput == "com4" || pathinput == "com5.txt" ||

pathinput == "com6" || pathinput == "com7" || pathinput == "com8" ||

pathinput == "com9" || pathinput == "lpt1" || pathinput == "lpt2" ||

pathinput == "lpt3" || pathinput == "lpt4" || pathinput == "lpt5" ||

pathinput == "lpt6" || pathinput == "lpt7" || pathinput == "lpt8" ||

pathinput == "lpt9" || ((e == NULL || f == NULL || g == NULL || o == NULL) && (a != NULL

|| b != NULL || c != NULL || d != NULL)))

{

return false;

}

return true;

}

void IsTestOk() {

int testcount;

int n = 1;

int counting = 3;

int\*\* testarrayofdots = new int\*[n];

CreateArray(testarrayofdots, n);

testarrayofdots[0][0] = 0;

testarrayofdots[0][1] = 0;

testarrayofdots[0][2] = 0;

int radius = 1;

int circleX = 0;

int circleY = 0;

int circleZ = 0;

int\*\* testarrayofdotsincircle = new int\*[counting];

CreateArray(testarrayofdotsincircle, counting);

IsInCircle(testarrayofdots, testarrayofdotsincircle, n, counting, radius, circleX, circleY, circleZ);

testcount = 0;

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

if (testarrayofdots[0][i] == testarrayofdotsincircle[0][i]) {

testcount++;

}

}

for (int i = 0; i < n; i++)

{

delete[] testarrayofdots[i];

}

delete[] testarrayofdots;

for (int i = 0; i < counting; i++)

{

delete[] testarrayofdotsincircle[i];

}

delete[] testarrayofdotsincircle;

if (testcount == 3)

{ //1

n = 1;

counting = 3;

testarrayofdots = new int\*[n];

CreateArray(testarrayofdots, n);

testarrayofdots[0][0] = 2;

testarrayofdots[0][1] = 2;

testarrayofdots[0][2] = 2;

radius = 1;

circleX = 0;

circleY = 0;

circleZ = 0;

testarrayofdotsincircle = new int\*[counting];

CreateArray(testarrayofdotsincircle, counting);

IsInCircle(testarrayofdots, testarrayofdotsincircle, n, counting, radius, circleX, circleY, circleZ);

testcount = 0;

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

if (testarrayofdots[0][i] != testarrayofdotsincircle[0][i]) {

testcount++;

}

}

for (int i = 0; i < n; i++)

{

delete[] testarrayofdots[i];

}

delete[] testarrayofdots;

for (int i = 0; i < counting; i++)

{

delete[] testarrayofdotsincircle[i];

}

delete[] testarrayofdotsincircle;

if (testcount == 3) { //2

n = 1;

counting = 3;

testarrayofdots = new int\*[n];

CreateArray(testarrayofdots, n);

testarrayofdots[0][0] = -2;

testarrayofdots[0][1] = -2;

testarrayofdots[0][2] = -2;

radius = 2;

circleX =-1;

circleY = -1;

circleZ = -1;

testarrayofdotsincircle = new int\*[counting];

CreateArray(testarrayofdotsincircle, counting);

IsInCircle(testarrayofdots, testarrayofdotsincircle, n, counting, radius, circleX, circleY, circleZ);

testcount = 0;

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

if (testarrayofdots[0][i] == testarrayofdotsincircle[0][i]) {

testcount++;

}

}

for (int i = 0; i < n; i++)

{

delete[] testarrayofdots[i];

}

delete[] testarrayofdots;

for (int i = 0; i < counting; i++)

{

delete[] testarrayofdotsincircle[i];

}

delete[] testarrayofdotsincircle;

if (testcount == 3) { //3

n = 3;

counting = 2;

testarrayofdots = new int\*[n];

CreateArray(testarrayofdots, n);

testarrayofdots[0][0] = -2;

testarrayofdots[0][1] = -1;

testarrayofdots[0][2] = -3;

testarrayofdots[1][0] = 2000;

testarrayofdots[1][1] = 2002;

testarrayofdots[1][2] = 2001;

testarrayofdots[2][0] = 3;

testarrayofdots[2][1] = 2;

testarrayofdots[2][2] = 1;

radius = 10;

circleX = 0;

circleY = 0;

circleZ = 0;

testarrayofdotsincircle = new int\*[counting];

CreateArray(testarrayofdotsincircle, counting);

IsInCircle(testarrayofdots, testarrayofdotsincircle, n, counting, radius, circleX, circleY, circleZ);

testcount = 0;

if (testarrayofdotsincircle[0][0] == -2 && testarrayofdotsincircle[0][1] == -1 && testarrayofdotsincircle[0][2] == -3) {

testcount++;

if (testarrayofdotsincircle[1][0] == 3 && testarrayofdotsincircle[1][1] == 2 && testarrayofdotsincircle[1][2] == 1){

testcount++;

}

}

for (int i = 0; i < n; i++)

{

delete[] testarrayofdots[i];

}

delete[] testarrayofdots;

for (int i = 0; i < counting; i++)

{

delete[] testarrayofdotsincircle[i];

}

delete[] testarrayofdotsincircle;

if (testcount == 2) { //4

n = 3;

counting = 1;

testarrayofdots = new int\*[n];

CreateArray(testarrayofdots, n);

testarrayofdots[0][0] = 126;

testarrayofdots[0][1] = 123;

testarrayofdots[0][2] = 111;

testarrayofdots[1][0] = -1;

testarrayofdots[1][1] = 0;

testarrayofdots[1][2] = 0;

testarrayofdots[2][0] = -12;

testarrayofdots[2][1] = 33;

testarrayofdots[2][2] = 11;

radius = 1;

circleX = 0;

circleY = 0;

circleZ = 0;

testarrayofdotsincircle = new int\*[counting];

CreateArray(testarrayofdotsincircle, counting);

IsInCircle(testarrayofdots, testarrayofdotsincircle, n, counting, radius, circleX, circleY, circleZ);

testcount = 0;

if (testarrayofdotsincircle[0][0] == -1 && testarrayofdotsincircle[0][1] == 0 && testarrayofdotsincircle[0][2] == 0) {

testcount++;

}

for (int i = 0; i < n; i++)

{

delete[] testarrayofdots[i];

}

delete[] testarrayofdots;

for (int i = 0; i < counting; i++)

{

delete[] testarrayofdotsincircle[i];

}

delete[] testarrayofdotsincircle;

if (testcount == 1) { //5

cout << "All tests passed" << endl;

}

else

{

cout << "Something went wrong. Tests not passed!" << endl;

}

}

else

{

cout << "Something went wrong. Tests not passed!" << endl;

}

}

else

{

cout << "Something went wrong. Tests not passed!" << endl;

}

}

else

{

cout << "Something went wrong. Tests not passed!" << endl;

}

}

else

{

cout << "Something went wrong. Tests not passed!" << endl;

}

}

[Конец structure.cpp --- ]

[Начало main.cpp ---]

#include <fstream>

#include <iostream>

#include "Structure.h"

#include "Input.h"

#include "CreateArray.h"

#include "File.h"

#include "Circle.h"

using namespace std;

int main()

{

bool check;

int radius = 0, x = 0, y = 0, z = 0;

do

{

IsTestOk();

check = Menu();

if (check == true)

{

int n = input();

int previouschoice = 1;

int\*\* arrayofdots = new int\*[n];

CreateArray(arrayofdots, n);

FillArray(arrayofdots, n, previouschoice);

CircleData(radius, x, y, z, n);

int counting = HowMany(arrayofdots, n, radius, x, y, z);

int\*\* arrayofdotsincircle = new int\*[counting];

CreateArray(arrayofdotsincircle, counting);

IsInCircle(arrayofdots, arrayofdotsincircle, n, counting, radius, x, y, z);

ShowSecondArray(arrayofdotsincircle, counting);

ReadFile(arrayofdots, arrayofdotsincircle, radius, x, y, z, n, counting);

for (int i = 0; i < n; i++)

{

delete[] arrayofdots[i];

}

delete[] arrayofdots;

for (int i = 0; i < counting; i++)

{

delete[] arrayofdotsincircle[i];

}

delete[] arrayofdotsincircle;

}

} while (check == true);

}

[Конец main.cpp --- ]

[Конец программы --- ]