

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

**"МИРЭА - Российский технологический университет"**

**РТУ МИРЭА**

Институт информационных технологий (ИТ)

Кафедра математического обеспечения и стандартизации информационных технологий (МОСИТ)

**ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ №3**

**по дисциплине**

**«Структуры и алгоритмы обработки данных»**

Тема. Стандартные типы данных языка программирования для представления

многоэлементных однородных структур данных задачи, с бинарными отношениями между элементами

|  |  |  |
| --- | --- | --- |
| Выполнил студент группы ИКБО-06-22 |  | Кликушин В.И. |
| Принял старший преподаватель |  | Скворцова Л.А. |

Москва 2023

ОГЛАВЛЕНИЕ

[1 ЗАДАНИЕ №1 3](#_Toc131350052)

[1.1 УСЛОВИЕ ЗАДАЧИ И ЗАДАНИЕ ВАРИАНТА 3](#_Toc131350053)

[1.2 АТД ЗАДАЧИ 3](#_Toc131350054)

[1.3 ТАБЛИЦА ТЕСТОВ 4](#_Toc131350055)

[1.4 КОД ПРОЕКТА 4](#_Toc131350056)

[1.4.1 Код файла statATD.h 4](#_Toc131350057)

[1.4.2 Код файла statATD.cpp 5](#_Toc131350058)

[1.4.3 Код файла siaod\_3.cpp 6](#_Toc131350059)

[1.5 РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ 8](#_Toc131350060)

[2 ЗАДАНИЕ №2 11](#_Toc131350061)

[2.1 УСЛОВИЕ ЗАДАЧИ И ЗАДАНИЕ ВАРИАНТА 11](#_Toc131350062)

[2.2 АТД ЗАДАЧИ 11](#_Toc131350063)

[2.3 ТАБЛИЦА ТЕСТОВ 12](#_Toc131350064)

[2.4 КОД ПРОЕКТА 12](#_Toc131350065)

[2.4.1 Код файла DynATD.h 12](#_Toc131350066)

[2.4.2 Код файла DynATD.cpp 13](#_Toc131350067)

[2.4.3 Код файла siaod3\_dyn.cpp 15](#_Toc131350068)

[2.5 РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ 17](#_Toc131350069)

[3 ЗАДАНИЕ №3 19](#_Toc131350070)

[3.1 УСЛОВИЕ ЗАДАЧИ И ЗАДАНИЕ ВАРИАНТА 19](#_Toc131350071)

[3.2 МАТЕМАТИЧЕСКАЯ МОДЕЛЬ РЕШЕНИЯ 19](#_Toc131350072)

[3.3 АТД ЗАДАЧИ 20](#_Toc131350073)

[3.4 ТАБЛИЦА ТЕСТОВ 22](#_Toc131350074)

[3.5 КОД ПРОЕКТА 23](#_Toc131350075)

[3.6 РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ 27](#_Toc131350076)

[4 ВЫВОДЫ 28](#_Toc131350077)

[5 ИСПОЛЬЗУЕМЫЕ ИСТОЧНИКИ 29](#_Toc131350078)

# 1 ЗАДАНИЕ №1

## 1.1 УСЛОВИЕ ЗАДАЧИ И ЗАДАНИЕ ВАРИАНТА

Разработать АТД задачи варианта по управлению многомерными данными и реализовать на статическом многомерном массиве.

Задание варианта: Дана квадратная матрица. Найти минимальное значение среди элементов, расположенных под главной диагональю.

Требования к выполнению: Разработать АТД для задачи варианта; включить к описание АТД операции заполнения массива: с клавиатуры, датчиком случайных чисел; вывода массива на экран построчно; операции для решения задачи варианта; выполнить реализацию АТД; разработать и реализовать основную задачу варианта как отдельную функцию, возможно с параметрами; управление вычислительным процессом через диалоговый интерфейс – текстовое меню.

## 1.2 АТД ЗАДАЧИ

АТД statATD

{

Данные

n – количество элементов множества

X – список значений элементов множества, реализованный на статическом двумерном массиве

statATD – тип множества

typeitem – тип элементов множества

Операции

1. Заполнение двумерного массива значениями элементов множества с клавиатуры

Предусловие. Data – ссылка на множество типа StatATD

Постусловие. Заполненный массив data из n\*n элементов. Нет возвращаемого значения

**input\_array(StatATD& data);**

1. Заполнение двумерного массива значениями элементов множества случайными значениями

Предусловие. Data – ссылка на множество типа StatATD

Постусловие. Заполненный массив data из n\*n элементов. Нет возвращаемого значения

**random\_input\_array(StatATD& data);**

1. Вывод значения элементов множества в консоль

Предусловие. Data – ссылка на множество типа StatATD

Постусловие. Вывод значений массива. Нет возвращаемого значения

**output\_array(StatATD& data);**

1. Поиск минимального значения среди элементов, расположенных под главной диагональю

Предусловие. Data – ссылка на множество типа StatATD

Постусловие. Возвращает целочисленное значение, соответствующее минимальному найденному элементу

**int find\_min\_elem(StatATD& data);**

## 1.3 ТАБЛИЦА ТЕСТОВ

|  |  |  |
| --- | --- | --- |
| Поиск минимального значения среди элементов, расположенных под главной диагональю  find\_min\_elem(StatATD& data) | | |
| Номер теста | Входные данные | Эталон результата |
| 1 | data.n = 1  data.X = {{111}} | Вывод сообщения: «Под главной диагональю нет элементов – матрица единичного размера» |
| 2 | data.n = 2  data.X = {{2, 3}, {2, 3}} | mini = 2 |
| 3 | data.n = 2  data.X = {{1, 1}, {777, 1}} | mini = 2 |
| 4 | data.n = 3  data.X = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}} | mini = 4 |
| 5 | data.n = 3  data.X = {{10, 3, 5}, {17, 4, 4}, {35, 333, 1}} | mini = 17 |

## 1.4 КОД ПРОЕКТА

### 1.4.1 Код файла statATD.h

#pragma once

#include <iostream>

using namespace std;

const unsigned int N = 100;

typedef int typeitem;

struct StatATD

{

int n = 0;

typeitem X[N][N];

StatATD();

};

void input\_array(StatATD& data);

void random\_input\_array(StatATD& data);

void output\_array(StatATD& data);

int find\_min\_elem(StatATD& data);

### 1.4.2 Код файла statATD.cpp

#include “StatATD.h”

#include <ctime>

StatATD::StatATD()

{

n = 0;

}

void input\_array(StatATD& data)

{

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

{

for (int j = 0; j < data.n; j++)

{

cin >> data.X[i][j];

}

}

}

void random\_input\_array(StatATD& data)

{

srand(time(NULL));

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

{

for (int j = 0; j < data.n; j++)

{

data.X[i][j] = rand() % 100;

}

}

}

void output\_array(StatATD& data)

{

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

{

for (int j = 0; j < data.n; j++)

{

cout << data.X[i][j] << “ “;

}

cout << ‘\n’;

}

}

int find\_min\_elem(StatATD& data)

{

int mini = INT\_MAX;

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

{

for (int j = 0; j < data.n; j++)

{

if (i > j)

{

if (mini > data.X[i][j])

{

mini = data.X[i][j];

}

}

}

}

return mini;

}

### 1.4.3 Код файла siaod\_3.cpp

#include <iostream>

#include “StatATD.h”

using namespace std;

int main()

{

system(“chcp 1251”);

int num,n;

StatATD data;

while (true)

{

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

cout << « Меню\n»;

cout << « 1. Заполнить матрицу с клавиатуры\n»;

cout << « 2. Заполнить матрицу случайными числами\n»;

cout << « 3. Вывести матрицу\n»;

cout << « 4. Найти минимальное значение среди элементов, расположенных под главной диагональю матрицы\n»;

cout << « 5. Завершение работы программы\n»;

cin >> num;

switch (num)

{

case 1:

cout << « Введите размер матрицы: «;

cin >> n;

data.n = n;

if (n == 0 or n > N)

{

cout << « Матрица некоректного размера\n»;

}

else

{

input\_array(data);

}

break;

case 2:

cout << « Введите размер матрицы: «;

cin >> n;

data.n = n;

if (n == 0 or n > N)

{

cout << « Матрица некоректного размера\n»;

}

else

{

random\_input\_array(data);

}

break;

case 3:

if (data.n == 0)

{

cout << « Матрица нулевого размера\n»;

}

else

{

cout << “Data:\n”;

output\_array(data);

}

break;

case 4:

if (data.n == 0)

{

cout << « Матрица нулевого размера\n»;

}

else if (data.n == 1)

{

cout << « Под диагональю нету элементов – матрица размера 1\n»;

}

else

{

cout << “ Min Value is “<< find\_min\_elem(data) << endl;

}

break;

case 5:

return 0;

default:

cout << “ Нет такого пункта\n”;

}

}

return 0;

}

## 1.5 РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 1 – Результат теста 1.1

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 2 – Результат тестирования 1.2

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 3 – Результат тестирования 1.3

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 4 – Результат тестирования 1.4

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 5 – Результат тестирования 1.5

# 2 ЗАДАНИЕ №2

## 2.1 УСЛОВИЕ ЗАДАЧИ И ЗАДАНИЕ ВАРИАНТА

Разработать АТД задачи варианта по управлению многомерными данными и реализовать на динамическом многомерном массиве.

Задание варианта: Дана квадратная матрица. Найти минимальное значение среди элементов, расположенных под главной диагональю.

Требования к выполнению: Включить в АТД задачи варианта дополнительную операцию создания многомерного массива заданного размера; двумерный массив определить как двойной указатель; для управления динамической памятью использовать аппарат языка С++; ввод размеров массива для хранения данных выполнить в основной программе (функция main).

## 2.2 АТД ЗАДАЧИ

АТД DynATD

{

Данные

n – количество элементов множества

X – список значений элементов множества, реализованный на динамическом двумерном массиве

DynATD – тип множества

typeitem – тип элементов множества

Операции

1. Заполнение двумерного массива значениями элементов множества с клавиатуры

Предусловие. data – ссылка на множество типа DynATD

Постусловие. Заполненный массив data из n\*n элементов. Нет возвращаемого значения

**input\_array(DynATD& data);**

1. Заполнение двумерного массива значениями элементов множества случайными значениями

Предусловие. data – ссылка на множество типа DynATD

Постусловие. Заполненный массив data из n\*n элементов. Нет возвращаемого значения

**random\_input\_array(DynATD& data);**

1. Вывод значения элементов множества в консоль

Предусловие. data – ссылка на множество типа DynATD

Постусловие. Вывод значений массива. Нет возвращаемого значения

**output\_array(DynATD& data);**

1. Поиск минимального значения среди элементов, расположенных под главной диагональю

Предусловие. data – ссылка на множество типа DynATD

Постусловие. Возвращает целочисленное значение, соответствующее минимальному найденному элементу

**int find\_min\_elem(DynATD& data);**

1. Создание динамического двумерного массива

Предусловие. data – ссылка на множество типа DynATD

Постусловие. Создает динамический размер заданной размерности. Нет возвращаемого значения

**void create\_array(DynATD& data);**

## 2.3 ТАБЛИЦА ТЕСТОВ

|  |  |  |
| --- | --- | --- |
| Поиск минимального значения среди элементов, расположенных под главной диагональю  find\_min\_elem(DynATD& data) | | |
| Номер теста | Входные данные | Эталон результата |
| 1 | data.n = 1  data.X = {{3}} | Вывод сообщения: «Под главной диагональю нет элементов – матрица единичного размера» |
| 2 | data.n = 2  data.X = {{11, 14}, {76, 1}} | mini = 76 |
| 3 | data.n = 2  data.X = {{5, 5}, {5, 5}} | mini = 5 |
| 4 | data.n = 3  data.X = {{12,13,14}, {6, 45, 11}, {643, 13, 14}} | mini = 6 |
| 5 | data.n = 3  data.X = {{0, 0, 0}, {567, 0, 0}, {878, 999, 0}} | mini = 567 |

## 2.4 КОД ПРОЕКТА

### 2.4.1 Код файла DynATD.h

#pragma once

#include <iostream>

using namespace std;

typedef int typeitem;

struct DynATD

{

int n;

typeitem \*\*X = nullptr;

DynATD();

~DynATD();

};

void create\_array(DynATD& data);

void input\_array(DynATD& data);

void random\_input\_array(DynATD& data);

void output\_array(DynATD& data);

int find\_min\_elem(DynATD& data);

### 2.4.2 Код файла DynATD.cpp

#include "DynATD.h"

#include <ctime>

DynATD::DynATD()

{

n = 0;

}

DynATD :: ~DynATD()

{

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

{

if (X[i] != nullptr)

{

delete[] X[i];

}

}

if (X != nullptr)

{

delete[] X;

}

}

void create\_array(DynATD& data)

{

data.X = new typeitem \* [data.n];

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

{

data.X[i] = new typeitem[data.n];

}

}

void input\_array(DynATD& data)

{

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

{

for (int j = 0; j < data.n; j++)

{

cin >> data.X[i][j];

}

}

}

void random\_input\_array(DynATD& data)

{

srand(time(NULL));

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

{

for (int j = 0; j < data.n; j++)

{

data.X[i][j] = rand() % 100;

}

}

}

void output\_array(DynATD& data)

{

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

{

for (int j = 0; j < data.n; j++)

{

cout << data.X[i][j] << " ";

}

cout << '\n';

}

}

int find\_min\_elem(DynATD& data)

{

int mini = INT\_MAX;

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

{

for (int j = 0; j < data.n; j++)

{

if (i > j)

{

if (mini > data.X[i][j])

{

mini = data.X[i][j];

}

}

}

}

return mini;

}

### 2.4.3 Код файла siaod3\_dyn.cpp

#include <iostream>

#include "DynATD.h"

using namespace std;

int main()

{

system("chcp 1251");

int num, n;

DynATD data;

while (true)

{

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

cout << " Меню\n";

cout << " 1. Заполнить матрицу с клавиатуры\n";

cout << " 2. Заполнить матрицу случайными числами\n";

cout << " 3. Вывести матрицу\n";

cout << " 4. Найти минимальное значение среди элементов, расположенных под главной диагональю матрицы\n";

cout << " 5. Завершение работы программы\n";

cin >> num;

switch (num)

{

case 1:

cout << " Введите размер матрицы: ";

cin >> n;

data.n = n;

if (n == 0)

{

cout << " Матрица некоректного размера\n";

}

else

{

create\_array(data);

input\_array(data);

}

break;

case 2:

cout << " Введите размер матрицы: ";

cin >> n;

data.n = n;

if (n == 0)

{

cout << " Матрица некоректного размера\n";

}

else

{

create\_array(data);

random\_input\_array(data);

}

break;

case 3:

if (data.n == 0)

{

cout << " Матрица нулевого размера\n";

}

else

{

cout << "Data:\n";

output\_array(data);

}

break;

case 4:

if (data.n == 0)

{

cout << " Матрица нулевого размера\n";

}

else if (data.n == 1)

{

cout << " Под диагональю нету элементов - матрица размера 1\n";

}

else

{

cout << " Min Value is " << find\_min\_elem(data) << endl;

}

break;

case 5:

return 0;

default:

cout << " Нет такого пункта\n";

}

}

return 0;

}

## 2.5 РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 6 – Результат теста 2.1

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 7 – Результат тестирования 2.2

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 8 – Результат тестирования 2.3

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 9 – Результат тестирования 2.4

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 10 – Результат тестирования 2.5

# 3 ЗАДАНИЕ №3

## 3.1 УСЛОВИЕ ЗАДАЧИ И ЗАДАНИЕ ВАРИАНТА

Задание варианта: Дано 3n точек на плоскости, причем никакие три из них не лежат на одной прямой. Построить множество n треугольников с вершинами в этих точках так, чтобы никакие два треугольника не пересекались и не содержали друг друга.

Требования по выполнению: Описать математическую модель решения задачи; определить структуру для хранения данных задачи; определить АТД задачи; структуру хранения данных реализовать на основе шаблона, размеры определить при вводе с клавиатуры; разработать программу, управляющую процессом посредством текстового меню; определить объем памяти, требующийся программе для хранения данных.

## 3.2 МАТЕМАТИЧЕСКАЯ МОДЕЛЬ РЕШЕНИЯ

В ходе решения задачи были задействованы следующие формулы:

* + 1. Проверка, лежат ли три данных точки на одной прямой:
    2. Проверка треугольников на пересечение или нахождение одного треугольника внутри другого: для каждой пары сравниваются каждая из трех сторон одного треугольника с тремя остальными сторонами другого. Рассматриваются частные случаи, когда прямые параллельны и перпендикулярны. Составляются уравнения двух прямых, на которых лежат отрезки. Находится коэффициент k для прямых, а также коэффициент b:

k = tan(a) = ; y = kx + b

* + 1. Если пересечения отрезков не найдено, проверяется лежит ли треугольник внутри второго треугольника. Используется метод геометрического луча. Из данной точки пускается луч по какой-либо оси в каком-либо направлении. Затем проверяются пересечения со сторонами многоугольника и ведётся подсчёт пересечений. Таким образом если количество пересечений чётное, то значит точка лежит вне многоугольника, если же кол-во нечётное, то значит точка лежит внутри.

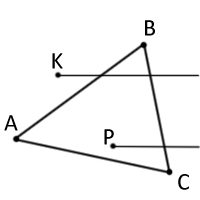


Рисунок 11 – Метод геометрического луча

* + 1. Количество уникальных треугольников можно рассчитать с помощью следующей формулы сочетаний:

## 3.3 АТД ЗАДАЧИ

АТД Point

{

Данные

x – абсцисса точки, число с плавающей точкой

y – ордината точки, число с плавающей точкой

Операции

1. Заполнение вектора точек Point с клавиатуры

Предусловие. data – ссылка на вектор из элементов типа Point, N – число элементов в векторе.

Постусловие. Заполненный вектор data из N элементов. Нет возвращаемого значения.

**input\_points(vector <Point>& data,int N);**

1. Вывод вектора точек

Предусловие. data – вектор из элементов типа Point

Постусловие. Выводит точки из вектора в консоль. Нет возвращаемого значения.

**output\_points(vector <Point> data, int N);**

1. Проверка, лежат ли три точки на одной прямой

Предусловие. a, b, c – элементы типа Point.

Постусловие. Возвращает булево значение: 1 – если точки лежат на одной прямой, иначе 0.

**on\_the\_line(Point a, Point b, Point c);**

1. Проверка, лежат ли какие – то три точки из данного множества на одной прямой

Предусловие. data – вектор из элементов типа Point.

Постусловие. Возвращает булево значение: 1 – если найдены три точки, которые лежат на одной прямой, иначе 0.

**check\_all\_points(vector <Point> data);**

АТД Triangle

{

Данные

A, B, C – три вершины треугольника, элементы типа Point

Операции

1. Генерация всех возможных треугольников из данного множества точек

Предусловие. figures – ссылка на вектор из элементов типа Triangle, data – вектор из элементов типа Point, M – размерность генерации сочетаний из N точек

Постусловие. Заполненный вектор из элементов типа Triangle

**generate\_permutations(vector <Triangle>& figures,vector <Point> data, int M);**

1. Вывод множества треугольников с координатами их вершин

Предусловие. figures – вектор из элементов типа Triangle

Постусловие. Выводит в консоль множество треугольников, задаваемых координатами трех вершин

**output\_permutations(vector <Triangle> figures);**

1. Проверка, пересекаются ли два треугольника, или лежат ли они друг в друге

Предусловие. ABC, CDE – элементы типа Triangle

Постусловие. Возвращает булево значение: 1, если треугольники пересекаются или лежат друг в друге, иначе 0

**compare\_triangles(Triangle ABC, Triangle CDE);**

1. Проверка двух отрезков на пересечение

Предусловие. a, b, c, d – элементы типа Point

Постусловие. Возвращает булево значение: 1, если у отрезков с началом в точке a и концом в точке b и отрезком с началом в точке c и концом в точке d есть общие точки, иначе 0

**check\_segments(Point a, Point b, Point c, Point d);**

1. Проверка, лежит ли первый треугольник внутри второго

Предусловие. ABC, CDE – элементы типа Triangle

Постусловие. Возвращает булево значение: 1, если первый треугольник лежит внутри второго, иначе 0

**check\_inside(Triangle ABC, Triangle CDE);**

1. Создание нового множества треугольников в соответствие с требованиями задачи

Предусловие. figures, new\_set – вектора из элементов типа Triangle

Постусловие. Выводит новое множество треугольников в консоль

**create\_set(vector <Triangle>& figures,vector <Triangle>& new\_set);**

## 3.4 НАБОР ТЕСТОВ

В целях большей наглядности входные и выходные данные были представлены в виде скриншотов координатной плоскости.

1. Тест №1

Входные данные: data = {(-4,0), (0,0), (2,1), (1,4), (-1,4), (0,2)}

Изображение выглядит как календарь

Автоматически созданное описаниеРисунок 11 – Входные данные для теста №3.1

Эталон результата:

Изображение выглядит как диаграмма

Автоматически созданное описание

Рисунок 12 – Первый набор множества для теста №3.1

Изображение выглядит как диаграмма

Автоматически созданное описание

Рисунок 13 – Второй набор множества для теста №3.1

Изображение выглядит как диаграмма

Автоматически созданное описание

Рисунок 14 – Третий набор множества для теста №3.1

Изображение выглядит как диаграмма

Автоматически созданное описание

Рисунок 15 – Четвертый набор множества для теста №3.1

Изображение выглядит как диаграмма

Автоматически созданное описание

Рисунок 16 – Пятый набор множества для теста №3.1

1. Тест №2

Входные данные: data = {(0,0), (2,2), (4,4), (3,1), (1,5), (-3,3)}

Эталон результата: через 1,2,3 точки можно провести прямые – набор не подходит.

Изображение выглядит как диаграмма

Автоматически созданное описание

Рисунок 17 – Эталон результата для теста №3.2

1. Тест №3

Входные данные: data = {(4,3), (5,7), (1,3)}

Эталон результата:

Изображение выглядит как диаграмма

Автоматически созданное описание

Рисунок 18 – Эталон результата для теста №3.3

1. Тест №4

Входные данные: data = {(0,1), (4,1), (2,2), (3,3), (1,3), (2,4)}

Эталон результата:

Изображение выглядит как диаграмма

Автоматически созданное описание

Рисунок 19 – Эталон результата для теста №3.4

Изображение выглядит как диаграмма

Автоматически созданное описание

Рисунок 20 – Эталон результата для теста №3.4

Изображение выглядит как диаграмма

Автоматически созданное описание

Рисунок 21 – Эталон результата для теста №3.4

## 3.5 КОД ПРОЕКТА

Код файла Point.h:

#pragma once

#include <vector>

#include <iostream>

using namespace std;

struct Point

{

double x, y;

Point();

Point(double x1,double y1);

};

void input\_points(vector <Point>& data,int N);

void output\_points(vector <Point> data, int N);

bool on\_the\_line(Point a, Point b, Point c);

bool check\_all\_points(vector <Point> data);

Код файла Point.cpp:

#include "Point.h"

Point::Point()

{

x = 0;

y = 0;

}

Point::Point(double x1, double y1)

{

x = x1;

y = y1;

}

void input\_points(vector <Point>& data,int N)

{

data.resize(N);

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

{

cout << " Введите абсциссу и ординату " << i + 1 << " точки\n";

cin >> data[i].x >> data[i].y;

}

}

void output\_points(vector <Point> data, int N)

{

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

{

cout << '(' << data[i].x << ',' << data[i].y << ')' << endl;

}

}

bool on\_the\_line(Point a, Point b,Point c)

{

bool rule\_1 = (c.x - a.x) / (b.x - a.x) == (c.y - a.y) / (b.y - a.y);

bool rule\_2 = (a.x == b.x and b.x == c.x) or (a.y == b.y and b.y == c.y);

return rule\_1 + rule\_2;

}

bool check\_all\_points(vector<Point> data)

{

for (int i = 0; i < data.size(); i++)

{

for (int j = i+1; j < data.size(); j++)

{

for (int x = j + 1; x < data.size(); x++)

{

if (on\_the\_line(data[i],data[j],data[x]))

{

cout << " Точки " << i + 1 << ", " << j + 1 << ", " << x + 1 << " лежат на одной прямой\n";

return 1;

}

}

}

}

return 0;

}

Код файла Triangle.h:

#pragma once

#include "Point.h"

#include <iostream>

using namespace std;

struct Triangle

{

Point A;

Point B;

Point C;

};

void generate\_permutations(vector <Triangle>& figures,vector <Point> data, int M);

void output\_permutations(vector <Triangle> figures);

bool compare\_triangles(Triangle ABC, Triangle CDE);

bool check\_segments(Point a, Point b, Point c, Point d);

bool check\_inside(Triangle ABC, Triangle CDE);

void create\_set(vector <Triangle>& figures,vector <Triangle>& new\_set, int N);

Код файла Triangle.cpp:

#include "Triangle.h"

#include <algorithm>

#include <iostream>

void generate\_permutations(vector <Triangle>& figures,vector <Point> data, int m)

{

vector <int> temp;

vector<bool> v(data.size());

fill(v.end() - m, v.end(), true);

do {

for (int i = 0; i < data.size(); ++i)

{

if (v[i])

{

temp.push\_back(i);

}

}

} while (next\_permutation(v.begin(), v.end()));

for (int i = 0; i < temp.size(); i +=3)

{

Triangle ABC;

ABC.A = data[temp[i]];

ABC.B = data[temp[i+1]];

ABC.C = data[temp[i+2]];

figures.push\_back(ABC);

}

}

void output\_permutations(vector<Triangle> figures)

{

for (int i = 0; i < figures.size(); i++)

{

cout << "((" << figures[i].A.x << "," << figures[i].A.y << "), (" << figures[i].B.x << "," << figures[i].B.y << "), (" << figures[i].C.x << "," << figures[i].C.y << "))" << endl;

}

}

bool compare\_triangles(Triangle ABC, Triangle CDE)

{

int total = 0;

total += check\_segments(ABC.A, ABC.B, CDE.A, CDE.B);

total += check\_segments(ABC.A, ABC.B, CDE.A, CDE.C);

total += check\_segments(ABC.A, ABC.B, CDE.B, CDE.C);

total += check\_segments(ABC.A, ABC.C, CDE.A, CDE.B);

total += check\_segments(ABC.A, ABC.C, CDE.A, CDE.C);

total += check\_segments(ABC.A, ABC.C, CDE.B, CDE.C);

total += check\_segments(ABC.B, ABC.C, CDE.A, CDE.B);

total += check\_segments(ABC.B, ABC.C, CDE.A, CDE.C);

total += check\_segments(ABC.B, ABC.C, CDE.B, CDE.C);

if (total > 0)

{

return 1;

}

else if (check\_inside(ABC, CDE))

{

return 1;

}

else if (check\_inside(CDE, ABC))

{

return 1;

}

return 0;

}

bool check\_segments(Point a, Point b, Point c, Point d)

{

if (b.x < a.x)

{

Point temp = a;

a = b;

b = temp;

}

if (d.x < c.x)

{

Point temp = c;

c = d;

d = temp;

}

if (b.x < c.x)

{

return 0;

}

else if ((a.x - b.x == 0 ) and (c.x - d.x == 0))

{

if (a.x == c.x)

{

if ( (max(a.y,b.y) < min(c.y,d.y)) or (min(a.y, b.y) > max(c.y, d.y)) )

{

return 1;

}

}

return 0;

}

else if (a.x - b.x == 0)

{

double X = a.x;

double k2 = (c.y - d.y) / (c.x - d.x);

double b2 = c.y - k2 \* c.x;

double Y = k2 \* X + b2;

if (c.x <= X and d.x >= X and min(a.y, b.y) <= Y and max(a.y, b.y) >= Y)

{

return 1;

}

return 0;

}

else if (c.x - d.x == 0)

{

double X = c.x;

double k1 = (a.y - b.y) / (a.x - b.x);

double b1 = a.y - k1 \* a.x;

double Y = k1 \* X + b1;

if (a.x <= X and b.x >= X and min(c.y, d.y) <= Y and max(c.y, d.y) >= Y)

{

return 1;

}

return 0;

}

else

{

double k1 = (a.y - b.y) / (a.x - b.x);

double k2 = (c.y - d.y) / (c.x - d.x);

double b1 = a.y - k1 \* a.x;

double b2 = c.y - k2 \* c.x;

if (k1 == k2)

{

return 0;

}

double X = (b2 - b1) / (k1 - k2);

if ((X < max(a.x, c.x)) or (X > min(b.x, d.x)))

{

return 0;

}

else

{

return 1;

}

}

}

bool check\_inside(Triangle Inside, Triangle Outside)

{

int count = 0;

Point Z;

Z.x = Inside.A.x;

Z.y = Inside.A.y;

Point inf;

inf.x = INT\_MAX;

inf.y = Z.y;

count += check\_segments(Z, inf, Outside.A, Outside.B);

count += check\_segments(Z, inf, Outside.A, Outside.C);

count += check\_segments(Z, inf, Outside.B, Outside.C);

if (count % 2 == 0)

{

return 0;

}

return 1;

}

void create\_set(vector<Triangle>& figures, vector <Triangle>& new\_set,int N)

{

for (int i = 0; i < figures.size(); i++)

{

for (int j = i + 1; j < figures.size(); j++)

{

if (compare\_triangles(figures[i], figures[j]) == 0)

{

new\_set.push\_back(figures[i]);

new\_set.push\_back(figures[j]);

}

}

}

figures = new\_set;

new\_set.clear();

output\_permutations(figures);

}

Код файла Task3.cpp:

#include "Point.h"

#include "Triangle.h"

#include <vector>

#include <iostream>

#include <algorithm>

using namespace std;

int main()

{

system("chcp 1251");

vector <Point> data;

vector <Triangle> figures, new\_set;

int N,num, n = 0;

while (true)

{

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

cout << " Меню\n";

cout << " 1. Задать координаты точек\n";

cout << " 2. Вывести координаты всех точек\n";

cout << " 3. Составить множество треугольников\n";

cout << " 4. Завершение работы программы\n";

cin >> num;

switch (num)

{

case 1:

cout << " Введите значение переменной n для задания 3n точек\n";

cin >> n;

N = 3 \* n;

if (n == 0)

{

cout << " Пустое множество точек\n";

}

else

{

input\_points(data,N);

}

break;

case 2:

if (n == 0)

{

cout << " Пустое множество точек\n";

}

else

{

output\_points(data, N);

}

break;

case 3:

if (n == 0)

{

cout << " Пустое множество точек\n";

}

else

{

if (check\_all\_points(data) != 1)

{

generate\_permutations(figures, data, 3);

create\_set(figures,new\_set,N);

}

}

break;

case 4:

data.clear();

return 0;

default:

cout << " Нет такого пункта\n";

}

}

return 0;

}

## 3.6 РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 1 – Результат теста 3.1

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 2 – Результат теста 3.2

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 3 – Результат теста №3.3

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 4 – Результат теста №3.4

# 4 ВЫВОДЫ

В ходе данной работы я приобрёл навыки работы со двумерными статическими и динамическими массивами данных, а также научился составлять собственные структуры данных для поставленной задачи и использовать математический аппарат для её решения.

# 5 ИСПОЛЬЗУЕМЫЕ ИСТОЧНИКИ

1. Учебно-методическое пособие СиАОД (часть 1)

2. Приложение к практическим работам – СДО (online-edu.mirea.ru)