

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 № 4**

**по дисциплине**

**«Структуры и алгоритмы обработки данных»**

Тема. Многомерные массивы

|  |  |  |
| --- | --- | --- |
| Выполнил студент группы ИКБО-65-23 |  | Петров А. |
| Принял старший преподаватель |  | Скворцова Л.А. |

Москва 2023

# **Оглавление**

[Оглавление 2](#_Toc164295387)

[1. УСЛОВИЕ ЗАДАЧИ. 3](#_Toc164295388)

[2. ЗАДАНИЕ 1. 5](#_Toc164295389)

[2.1. Разработка абстрактного типа данных задачи, используя шаблон 5](#_Toc164295390)

[2.2. Код алгоритма на языке С++. 7](#_Toc164295391)

[2.3. Тестирование алгоритма. 9](#_Toc164295392)

[3. ЗАДАНИЕ 2. 10](#_Toc164295393)

[3.1. Разработка дополнительной операции для создания многомерного массива заданного размера 10](#_Toc164295394)

[3.2. Код алгоритма на языке С++ 11](#_Toc164295395)

[3.3. Тестирование алгоритма. 14](#_Toc164295396)

[4. ЗАДАНИЕ 3. 15](#_Toc164295397)

[4.1. Изучение задачи 15](#_Toc164295398)

[4.2 Определение структуры для хранения данных задачи 16](#_Toc164295399)

[4.3. Разработка абстрактного типа данных задачи, используя шаблон 16](#_Toc164295400)

[4.4. Код алгоритма на языке С++ 18](#_Toc164295401)

[4.5 Тестирование алгоритма. 21](#_Toc164295402)

[4.6 Объем памяти для программы 22](#_Toc164295403)

[5. Выводы 23](#_Toc164295404)

[6. Информационные источники 24](#_Toc164295405)

# УСЛОВИЕ ЗАДАЧИ.

**Задание 1**

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

**Требования к выполнению задания 1**

1. Разработать АТД для задачи варианта.

Включить к описание АТД:

1) описание представления данных;

2) операции общие для всех вариантов:

− заполнения массива: с клавиатуры, датчиком случайных чисел;

− вывода массива на экран построчно.

3) Операции для решения задачи варианта;

2. Выполнить реализацию АТД. Для представления структуры данных задачи

использовать статический многомерный массив. Реализацию структуры

выполнить на основе типа данных struct (аналогично первым практическим работам).

3. Разработать тесты и выполнить тестирование программы.

4. Разработать и реализовать основную задачу варианта как отдельную функцию, возможно с параметрами.

Реализовать управление вычислительным процессом через диалоговый интерфейс – текстовое меню.

**Задание 2**

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

**Требования к выполнению задания 2**

1. Определить АТД задачи, используя АТД задания 1. Добавить в АТД дополнительную операцию создания многомерного массива заданного размера. Размеры массива передаются через параметры.

2. Выполнить реализацию АТД.

Требования к реализации АТД:

1) двумерный массив определить как двойной указатель;

2) для управления динамической памятью использовать аппарат языка С++ (операции new, delete).

3. Ввод размеров массива для хранения данных выполнить в основной программе (функция main).

4. Разработать программу задачи варианта. Используйте функции, разработанные в задании 1, возможно некоторые надо скорректировать в связи с изменением структуры представления.

**Вариант задачи к заданиям 1 и 2 - № 22**

Дана квадратная матрица. Найти определитель данной матрицы методом Гаусса.

**Задание 3**

Разработать программу решения задачи варианта (табл. 9) по управлению многомерными данными и реализовать с применением шаблона <vector> библиотеки STL.

**Требования к выполнению задания 3**

1. Изучить задачу. Определить предметную область. Описать математическую модель решения задачи.

*Примечание. Математическая модель = это описание решения задачи на языке математики, т.е. совокупность формул, которые определяют процесс решения*

*задачи.*

2. Определить структуру для хранения данных задачи. Изобразить структуру для понимания представления данных в памяти.

3. Определить АТД задачи.

Описать структуру представления данных задачи. Включить операции:

заполнение структуры хранения исходными данными, вывода структуры

данных, операции, представляющие процесс решения задачи.

4. Реализовать АТД.

Структуру хранения данных реализовать на основе шаблона <vector>, размеры определить при вводе с клавиатуры.

Реализовать все операции АТД.

Разработать тесты для тестирования программы.

5. Разработать программу, управляющую вычислительным процессом посредством текстового меню.

6. Определить объем памяти, требующийся программе для хранения данных.

7. Выполнить тестирование программы.

**Вариант задачи к заданию 3 - № 22**

На плоскости задано множество точек с целочисленными координатами. Необходимо найти количество отрезков, обладающих следующими свойствами: 1) оба конца отрезка принадлежат заданному множеству;

2) ни один конец отрезка не лежит на осях координат;

3) отрезок пересекается ровно с одной осью координат.

Напишите эффективную по времени и по используемой памяти программу для решения этой задачи.

# ЗАДАНИЕ 1.

## 2.1. Разработка абстрактного типа данных задачи, используя шаблон

|  |
| --- |
| АТД MyStruct  {  Данные:  size – количество элементов множества.  А – список значений элементов множества. |

Операции:

|  |
| --- |
| **Операция 1**  //Операция Создание массива в структуре.  //Предусловие: А – список значений элементов множества, size - количество элементов множества.  size < 20 и size > 0  //Постусловие: результат при успешном создание – 0, при безуспешном код завершения – 1.  int createArray (A, size);.- заголовок |

|  |
| --- |
| **Операция 2**  //Операция: заполнение структуры данных значениями.  //Предусловие: А – список значений элементов множества,  temp = 0 – для заполнения массива вручную  temp = 1 – для заполнения массива с использование датчика случайных чисел  size < 20 и size > 0  //Постусловие: результат при успешном выводе – 0, при безуспешном код завершения – 1.  int fill(A, temp);.- заголовок |

|  |
| --- |
| **Операция 3**  //Операция: вывод массива на экран построчно.  //Предусловие: А – список значений элементов множества, A.size > 0.  //Постусловие: результат при успешном выводе – массив, при безуспешном – ничего.  size < 20 и size > 0  void print(A);.- заголовок |

|  |
| --- |
| **Операция 4**  //Операция: Дана квадратная матрица. Найти определитель данной матрицы методом Гаусса.  //Предусловие: А – список значений элементов множества, pos – выбор первого элемента в матрице pos < 19.  //Постусловие: результат при успешном завершение алгоритма – значение определителя, при безуспешном код завершения – 1.  float determinant (A, pos);.- заголовок |

## 2.2. Код алгоритма на языке С++.

|  |
| --- |
| #include <iostream>  #include <cstdlib>  #include <stdlib.h>  #include <ctime>  #include <iomanip>  using namespace std;  typedef unsigned int typesize;  typedef int typeitem;  struct MyStruct  {  typesize size;  int A[2][20];  };  //создание массива в структуре  int createArray(MyStruct& s, int size);  //заполнение структуры данных значениями  int fill(MyStruct& s, int temp);  //вывод массива на экран построчно  void print(MyStruct s);  //Дана квадратная матрица. Найти определитель данной матрицы методом Гаусса.  float determinant (MyStruct& s, typesize pos);  // текстовое меню для управления программой  void menu(MyStruct& s);  int main()  {  setlocale(LC\_ALL, "Rus");  MyStruct A;  menu(A);  }  void menu(MyStruct& A)  {  cout << "Доступные функции программы" << endl  << "1. создание массива в структуре - createArray" << endl  << "2. заполнение структуры данных значениями - fill" << endl  << "3. вывод массива на экран построчно - print" << endl  << "4. Найти определитель данной матрицы методом Гаусса - determinant " << endl  << "0. Выход" << endl << endl;  int choice, pos, size, temp;  while (true)  {  cout << "Выберете функцию программы = ";  cin >> choice;  switch (choice)  {  case 1:  cout << "Количество элементов в строке = ";  cin >> size;  createArray(A, size);  break;  case 2:  cout << "0 – для заполнения массива вручную\n1 – для заполнения массива с использование датчика случайных чисел\n = ";  cin >> temp;  fill(A, temp);  break;  case 3:  print(A);  break;  case 4:  cout << "выбор первого элемента в матрице = ";  cin >> pos;  cout << "Определитель равен = " << determinant(A, pos) << endl;  break;  case 0:  cout << "Выход" << endl << endl;  return;  default:  cout << "Не правильный выбор" << endl;  break;  }  }  }  int createArray(MyStruct& s, int size)  {  if (size > 0 && size < 21)  {  s.size = size;  return 0;  }  cout << "Вы ввели не правильное значение size\n";  return 1;  }  int fill(MyStruct& s, int temp)  {  int temp1;  if (s.size < 0)  {  cout << "В массиве нет ячеек";  return 1;  }  if (temp == 0)  {  for (typesize i = 0; i < 2; i++)  {  for (typesize j = 0; j < s.size; j++)  {  cout << "Введите значение для строки" << i << "и столбца" << j << " = ";  cin >> temp1;  s.A[i][j] = temp1;  }  }  return 0;  }  else if (temp == 1)  {  for (typesize i = 0; i < 2; i++)  {  for (typesize j = 0; j < s.size; j++)  {  s.A[i][j] = rand();  }  }  return 0;  }  else  {  cout << "Вы ввели не правильное значение нужно 0 или 1" << endl;  return 1;  }  }  void print(MyStruct s)  {  if (s.size < 0 && s.size > 20)  return;  cout << "Множество чисел \n";  for (typesize i = 0; i < 2; i++)  {  for (typesize j = 0; j < s.size; j++)  {  cout << setw(5) << s.A[i][j] << " ";  }  cout << endl;  }  }  float determinant (MyStruct& s, typesize pos)  {  pos -= 1;  if (pos > 19)  return 1;  int matrix[2][2];  for (typesize i = 0; i < 2; i++)  {  for (typesize j = 0; j < 2; j++)  {  matrix[i][j] = s.A[i][j + pos];  }  }  // Приведение матрицы к треугольному виду  if (matrix[0][0] == 0)  {  swap(matrix[0][0], matrix[1][0]);  swap(matrix[0][1], matrix[1][1]);  return -1\*(matrix[0][0] \* matrix[1][1]);  }  else if(matrix[1][0] == 0)  return matrix[0][0] \* matrix[1][1];  double delitel = (double)matrix[1][0] / (double)matrix[0][0];  matrix[1][0] -= delitel \* matrix[0][1];  return matrix[0][0] \* matrix[1][1];  } |

## 2.3. Тестирование алгоритма.

Составленные тесты (Таблица 1), которые обрабатывают базовый случай для размерности массива равной 20 (n = 20).

Таблица 1 - Таблица тестирования

|  |  |  |
| --- | --- | --- |
| заполнение структуры данных значениями matrix2X2(A, pos) | | |
| Номер теста | Входные данные | Эталон результата |
| 1 | createArray (A, 20)  fill(A, 1)  print(A)  determinant (A, 1) | Определитель равен = 598764 |

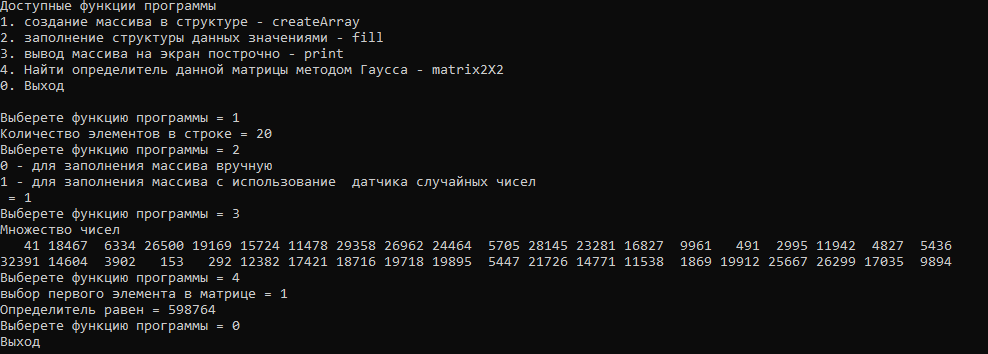


Рисунок 1 - Скриншот тестирования

# 3. ЗАДАНИЕ 2.

## 3.1. Разработка дополнительной операции для создания многомерного массива заданного размера

|  |
| --- |
| АТД MyStruct  {  Данные:  rows – количество элементов в строке.  columns - количество элементов в столбце  А – список значений элементов множества.  } |

|  |
| --- |
| **Переопределение операция 1**  //Операция Создание массива в структуре.  //Предусловие: А – список значений элементов множества, rows - количество строк в массиве, columns - количество столбцов в массиве.  //Постусловие: результат при успешном создание – 0, при безуспешном код завершения – 1.  int createArray (A, rows, columns);.- заголовок |
| **Переопределение Операция 4**  //Операция: Дана квадратная матрица. Найти определитель данной матрицы методом Гаусса.  //Предусловие: А – список значений элементов множества.   * Если в параметрах дана прямоугольная матрица то операция сначала ищет меньший столбец либо строку, и по его размеру делает квадратную матрицу.   //Постусловие: результат при успешном завершение алгоритма – значение определителя, при безуспешном код завершения – 1.  float determinant (A);.- заголовок |

## 3.2. Код алгоритма на языке С++

|  |
| --- |
| #include <iostream>  #include <cstdlib>  #include <stdlib.h>  #include <ctime>  #include <iomanip>  using namespace std;  typedef unsigned int typesize;  typedef int typeitem;  struct MyStruct  {  typesize rows{};  typesize columns{};  int\*\* A = nullptr;  };  //создание массива в структуре  int createArray(MyStruct& s, typesize rows, typesize columns);  //заполнение структуры данных значениями  int fill(MyStruct& s, int temp);  //вывод массива на экран построчно  void print(MyStruct s);  //Дана квадратная матрица. Найти определитель данной матрицы методом Гаусса.  float determinant(MyStruct& s);  // текстовое меню для управления программой  void menu(MyStruct& s);  int main()  {  setlocale(LC\_ALL, "Rus");  MyStruct A;  menu(A);  }  void menu(MyStruct& A)  {  cout << "Доступные функции программы" << endl  << "1. создание массива в структуре - createArray" << endl  << "2. заполнение структуры данных значениями - fill" << endl  << "3. вывод массива на экран построчно - print" << endl  << "4. Найти определитель данной матрицы методом Гаусса - determinant" << endl  << "0. Выход" << endl << endl;  int choice, pos, rows, columns, temp;  while (true)  {  cout << "Выберете функцию программы = ";  cin >> choice;  switch (choice)  {  case 1:  cout << "Количество элементов в строке и в столбце = ";  cin >> rows >> columns;  createArray(A, rows, columns);  break;  case 2:  cout << "0 – для заполнения массива вручную\n1 – для заполнения массива с использование датчика случайных чисел\n = ";  cin >> temp;  fill(A, temp);  break;  case 3:  print(A);  break;  case 4:  cout << "Определитель равен = " << determinant(A) << endl;  break;  case 0:  cout << "Выход" << endl << endl;  return;  default:  cout << "Не правильный выбор" << endl;  break;  }  }  }  int createArray(MyStruct& s, typesize rows, typesize columns)  {  if (s.rows >= 0 && s.columns >= 0)  {  s.rows = rows;  s.columns = columns;  s.A = new int\* [rows];  for (int i = 0; i < rows; i++)  s.A[i] = new int[columns];  return 0;  }  cout << "Вы ввели не правильное значение size\n";  return 1;  }  int fill(MyStruct& s, int temp)  {  int temp1;  if (s.rows < 0 || s.columns < 0)  {  cout << "В массиве нет ячеек";  return 1;  }  if (temp == 0)  {  for (typesize i = 0; i < s.rows; i++)  {  for (typesize j = 0; j < s.columns; j++)  {  cout << "Введите значение для строки " << i << " и столбца " << j << " = " << endl;  cin >> temp1;  s.A[i][j] = temp1;  }  }  return 0;  }  else if (temp == 1)  {  for (typesize i = 0; i < s.rows; i++)  {  for (typesize j = 0; j < s.columns; j++)  {  s.A[i][j] = rand();  }  }  return 0;  }  else  {  cout << "Вы ввели не правильное значение нужно 0 или 1" << endl;  return 1;  }  }  void print(MyStruct s)  {  if (s.rows < 0 || s.columns < 0)  return;  cout << "Множество чисел \n";  for (typesize i = 0; i < s.rows; i++)  {  for (typesize j = 0; j < s.columns; j++)  {  cout << setw(5) << s.A[i][j] << " ";  }  cout << endl;  }  }  float determinant(MyStruct& s)  {  int columns{}, rows{};  if (s.columns > s.rows)  {  columns = s.columns - (s.columns - s.rows);  rows = s.rows;  }  else if (s.rows > s.columns)  {  rows = s.rows - (s.rows - s.columns);  columns = s.columns;  }  else  {  rows = s.rows;  columns = s.columns;  }  int\*\* matrix{ new int\* [rows] {} };  for (typesize i = 0; i < rows; i++)  {  matrix[i] = new int[columns] {};  }  for (typesize i = 0; i < rows; i++)  {  for (typesize j = 0; j < columns; j++)  {  matrix[i][j] = s.A[i][j];  }  }  double det = 1;  int size = rows;  for (int i = 0; i < size; ++i)  {  double mx = fabs(matrix[i][i]);  int idx = i;  for (int j = i + 1; j < size; ++j)  if (mx < fabs(matrix[i][j])) mx = fabs(matrix[i][idx = j]);  if (idx != i)  {  for (int j = i; j < size; ++j)  {  double t = matrix[j][i];  matrix[j][i] = matrix[j][idx];  matrix[j][idx] = t;  }  det = -det;  }  for (int k = i + 1; k < size; ++k)  {  double t = matrix[k][i] / matrix[i][i];  for (int j = i; j < size; ++j)  matrix[k][j] -= matrix[i][j] \* t;  }  }  for (int i = 0; i < size; ++i) det \*= matrix[i][i];  return det;  } |

## 3.3. Тестирование алгоритма.

Составленные тесты (Таблица 2), которые обрабатывают базовый случай для размерности массива равной 2 на 2.

Таблица 2 - Таблица тестирования

|  |  |  |
| --- | --- | --- |
| заполнение структуры данных значениями determinant (A) | | |
| Номер теста | Входные данные | Эталон результата |
| 1 | createArray (A, 2, 2)  fill(A, 0)  << 1 << 2 << 3 << 4  print(A)  determinant(A)  print(A) | Определитель равен = -2 |

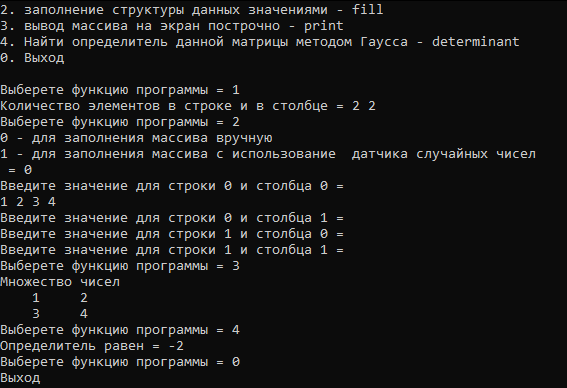


Рисунок 2 - Скриншот тестирования

# 4. ЗАДАНИЕ 3.

## 4.1. Изучение задачи

Предметная область этой задачи связана с анализом геометрических фигур на плоскости, а также с применением алгоритмов для подсчета количества отрезков, удовлетворяющих определенным условиям.

Математическая модель решения этой задачи может быть сформулирована следующим образом:

1. Пусть P - это множество точек на плоскости с целочисленными координатами (х и у)
2. Пусть S - это множество всех возможных отрезков, образованных парами точек из множества P, таких что:
   * Оба конца отрезка принадлежат множеству P
   * Ни один конец отрезка не лежит на осях координат
   * Отрезок пересекается ровно с одной из осей координат.
3. Необходимо найти количество отрезков из множества S
4. Для этого будем перебирать все пары точек из множества P и проверять их на соответствие условиям задачи.
5. Если точки (x1, y1) и (x2,y2) образуют отрезок, то: Отрезок пересекается ровно с одной осью координат, если точки находятся по разные стороны от оси x = 0 (одна точка имеет положительную координату по x, а другая отрицательную) либо обе с одной стороны но координаты y по разные.
6. Подсчитываем количество отрезков, удовлетворяющих условиям задачи

## 4.2 Определение структуры для хранения данных задачи

Для хранения данных задачи можно использовать простую структуру данных, такую как двумерный вектор (вектор векторов) в C++, где каждый внутренний вектор представляет собой координаты одной точки на плоскости. Каждая точка будет представлена парой целочисленных координат(x и y)

vector -> [ [x1, y1],

[x2, y2],

[x3, y3],

... ]

## 4.3. Разработка абстрактного типа данных задачи, используя шаблон

|  |
| --- |
| АТД MyStruct  {  Данные:  rows – количество элементов в строке.  columns = 2 – количество координат (X,Y)  А – список значений элементов множества. |

Операции:

|  |
| --- |
| **Операция 1**  **Переопределение операция 1**  //Операция Создание массива в структуре.  //Предусловие: А – список значений элементов множества, rows - количество строк в массиве которые содержат 2 элемента X и Y.  //Постусловие: результат при успешном создание – 0, при безуспешном код завершения – 1.  int createArray (A, rows);.- заголовок |

|  |
| --- |
| **Операция 2**  //Операция: заполнение структуры данных значениями.  //Предусловие: А – список значений элементов множества,  temp = 0 – для заполнения массива вручную  temp = 1 – для заполнения массива с использование датчика случайных чисел  //Постусловие: результат при успешном выводе – 0, при безуспешном код завершения – 1.  int fill(A, temp);.- заголовок |

|  |
| --- |
| **Операция 3**  //Операция: вывод массива на экран построчно.  //Предусловие: А – список значений элементов множества,  //Постусловие: результат при успешном выводе – массив, при безуспешном – ничего.  void print(A);.- заголовок |

|  |
| --- |
| **Операция 4**  //Операция: На плоскости задано множество точек с целочисленными координатами.  Необходимо найти количество отрезков, обладающих следующими  свойствами:  1) оба конца отрезка принадлежат заданному множеству;  2) ни один конец отрезка не лежит на осях координат;  3) отрезок пересекается ровно с одной осью координат.  //Предусловие: А – список значений элементов множества.  //Постусловие: результат при успешном завершение алгоритма – количество отрезков.  int specOtrez(MyStruct& s);.- заголовок |

## 4.4. Код алгоритма на языке С++

|  |
| --- |
| #include <vector>  #include <iostream>  #include <cstdlib>  #include <iostream>  #include <stdlib.h>  #include <ctime>  #include <iomanip>  using namespace std;  typedef unsigned int typesize;  struct MyStruct  {  typesize rows{};  typesize columns{2};  vector<vector<int>> A;  };  //создание массива в структуре  int createArray(MyStruct& s, int rows);  //заполнение структуры данных значениями  int fill(MyStruct& s, int temp);  //вывод массива на экран построчно  void print(MyStruct s);  /\*На плоскости задано множество точек с целочисленными координатами.  Необходимо найти количество отрезков, обладающих следующими  свойствами :  1) оба конца отрезка принадлежат заданному множеству;  2) ни один конец отрезка не лежит на осях координат;  3) отрезок пересекается ровно с одной осью координат.\*/  int specOtrez(MyStruct& s);  // текстовое меню для управления программой  void menu(MyStruct& s);  int main()  {  setlocale(LC\_ALL, "Rus");  MyStruct A;  menu(A);  }  void menu(MyStruct& A)  {  cout << "Доступные функции программы" << endl  << "1. создание массива в структуре - createArray" << endl  << "2. заполнение структуры данных значениями - fill" << endl  << "3. вывод массива на экран построчно - print" << endl  << "4. Поиск специальных отрезков - specOtrez" << endl  << "0. Выход" << endl << endl;  int choice, pos, size, temp;  while (true)  {  cout << "Выберете функцию программы = ";  cin >> choice;  switch (choice)  {  case 1:  cout << "Количество отрезков = ";  cin >> size;  createArray(A, size);  break;  case 2:  cout << "0 – для заполнения массива вручную\n1 – для заполнения массива с использование датчика случайных чисел\n = ";  cin >> temp;  fill(A, temp);  break;  case 3:  print(A);  break;  case 4:  cout << "Количество специальных отрезков: " << specOtrez(A) << endl;  break;  case 0:  cout << "Выход" << endl << endl;  return;  default:  cout << "Не правильный выбор" << endl;  break;  }  }  }  int createArray(MyStruct& s, int rows)  {  if (rows < 0)  {  cout << "Вы ввели не правильное значение n";  return 1;  }  s.rows = rows;  s.A.reserve(rows);  for (int i = 0; i < rows; ++i)  {  s.A.push\_back(vector<int>(s.columns));  }  return 0;  }  int fill(MyStruct& s, int temp)  {  if (s.rows < 0 || s.columns < 0)  {  cout << "В массиве нет ячеек";  return 1;  }  if (temp == 0)  {  for (typesize i = 0; i < s.rows; i++)  {  cout << "Введите значения для " << i+1 << " координат х и y точки = " << endl;  cin >> s.A[i][0] >> s.A[i][1];  }  return 0;  }  else if (temp == 1)  {  for (typesize i = 0; i < s.rows; i++)  {  for (typesize j = 0; j < s.columns; j++)  {  s.A[i][j] = rand();  }  }  return 0;  }  else  {  cout << "Вы ввели не правильное значение нужно 0 или 1" << endl;  return 1;  }  }  void print(MyStruct s)  {  cout << "Множество чисел \n";  for (int i = 0; i < s.rows; i++)  {  for (int j = 0; j < s.columns; j++)  {  cout << setw(5) << (j == 0 ? "x:" : "y:") << s.A[i][j] << " ";  }  cout << endl;  }  }  int specOtrez(MyStruct& s)  {  int count = 0;  int n = s.A.size();  // Перебираем все возможные пары точек  for (int i = 0; i < n; ++i)  {  for (int j = i + 1; j < n; ++j)  {  int x1 = s.A[i][0];  int y1 = s.A[i][1];  int x2 = s.A[j][0];  int y2 = s.A[j][1];  // Проверяем, что обе точки не лежат на осях координат  if (x1 != 0 && y1 != 0 && x2 != 0 && y2 != 0)  {  // Проверяем, пересекается ли отрезок ровно с одной осью координат  if ((x1 > 0 && x2 < 0 && y1 > 0 && y2 > 0) || (x1 > 0 && x2 > 0 && y1 < 0 && y2 > 0) || (x1 > 0 && x2 < 0 && y1 < 0 && y2 < 0) || (x1 < 0 && x2 < 0 && y1 > 0 && y2 < 0))  {  count++;  }  }  }  }  return count; |

## 4.5 Тестирование алгоритма.

Таблица 3 - Таблица тестирования

|  |  |  |
| --- | --- | --- |
| заполнение структуры данных значениями determinant (A) | | |
| Номер теста | Входные данные | Эталон результата |
| 1 | createArray (A, 4)  fill(A, 1)  print(A)  specOtrez (A) | Количество специальных отрезков: 0 |

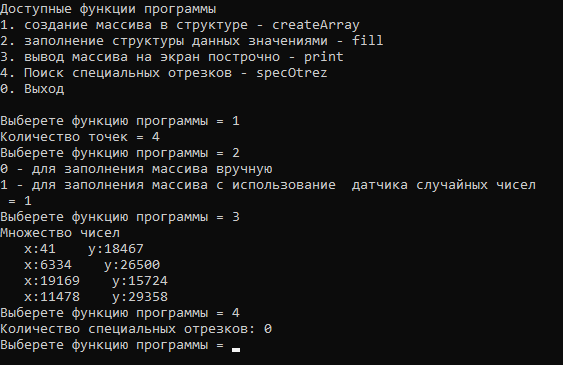


Рисунок 3 - Скриншот тестирования

Составленные тесты (Таблица 4)

Таблица 4 - Таблица тестирования

|  |  |  |
| --- | --- | --- |
| заполнение структуры данных значениями determinant (A) | | |
| Номер теста | Входные данные | Эталон результата |
| 1 | createArray (A, 2)  fill(A, 0)  << 1 << 2 << -1 << 2  print(A)  specOtrez (A) | Количество специальных отрезков: 1 |

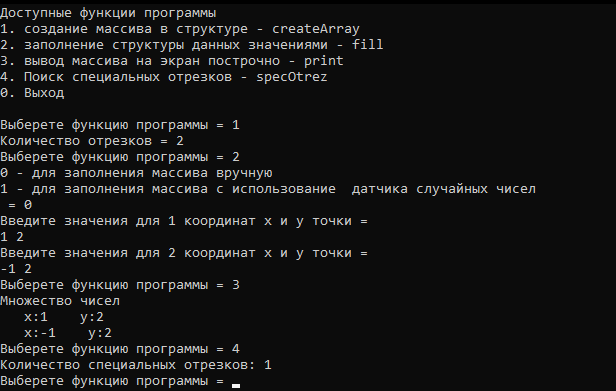


Рисунок 4 - Скриншот тестирования

## 4.6 Объем памяти для программы

Для хранения данных в программе мы используем двумерный вектор, где каждая точка представлена парой целочисленных координат (x и y). Объем памяти, необходимый для хранения данных, зависит от количества точек и размера целых чисел в системе.

Общий объем памяти, требуемый программой, можно выразить следующим образом:

*Объем памяти = Размер вектора + 2\*(Размер целого числа + Размер целого числа)*

# 5. Выводы

Исследование позволило достичь поставленных целей, связанных с работой с многомерными статическими и динамическими массивами в программировании. Полученные навыки включают определение и использование таких массивов, а также понимание их представления в оперативной памяти. Были разработаны структуры данных для эффективного хранения информации, а также алгоритмы для выполнения операций на многомерных массивах в соответствии с поставленными задачами. В результате данного исследования участники приобрели необходимые знания и навыки для успешной работы с многомерными массивами в программировании.

# 6. Информационные источники

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

2. Структуры и алгоритмы обработки данных – методические указания / Скворцова Л.А., Филатов А.С., Гусев К.В., Трушин С.М. - Москва, МИРЭА - Российский технологический университет, 2023.