|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  |  |  | | МИНОБРНАУКИ РОССИИ | | | | Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | |   Институт Информационных технологий | |
|  | |
| Кафедра Математического обеспечения и стандартизации информационных технологий | |
|  | |
|  | |

|  |  |
| --- | --- |
| **ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ № 2** | |
| **по дисциплине** | |
| **«**Структуры и алгоритмы обработки данных**»**  **Тема: «Двумерный массив»** | |
|  | |
| Выполнил студент группы ИКБО-08-21 | Тараканова Е.О. |
| Принял преподаватель | Ермаков С.Р. |

|  |  |  |
| --- | --- | --- |
| Лабораторная работа выполнена | «\_\_»\_\_\_\_\_\_\_202\_\_ г. | *(подпись студента)* |
|  |  |  |
| «Зачтено» | «\_\_»\_\_\_\_\_\_\_202\_\_ г. | *(подпись руководителя)* |

Москва 2022

# **Цель работы**

Получение навыков по определению двумерного массива для структуры данных задачи, а также по разработке алгоритмов операций на двумерном массиве в соответствии с задачей.

# **Постановка задачи**

1. Разработать программу по обработке данных, представленных в задаче матрицей и реализованной в программе двумерным (многомерным) статическим массивом.
2. Максимально допустимые размеры статического массива установить через глобальные константы Rows, Cols.
3. Если тип элемента массива не определен в варианте, определить массив для хранения значений целого типа.
4. Разработать функции по реализации алгоритмов заполнения матрицы: с клавиатуры, датчиком случайных чисел. Разработать функции по реализации алгоритма вывода массива на экран построчно.
5. Выполнить декомпозицию задачи варианта, разработать алгоритм решения. Реализовать функцию, выполняющую задачу и отладить ее.
6. Разработать программу, демонстрирующую работу всех функций.
7. Разработать программу по обработке данных, представленных в задаче матрицей и реализованной в программе двумерным (многомерным) динамическим массивом.
8. Размеры массива должны определяться пользователем.
9. Двумерный массив определить как двойной указатель и выполнить его создание операцией new.
10. Разработать функции по реализации алгоритмов заполнения матрицы: с клавиатуры, датчиком случайных чисел. Разработать функции по реализации алгоритма вывода массива на экран построчно.
11. Выполнить декомпозицию задачи варианта, разработать алгоритм решения. Реализовать функцию, выполняющую задачу и отладить ее.
12. Разработать программу, демонстрирующую работу всех функций.
13. Разработать программу по задаче варианта с реализацией данных задачи с применением контейнера vector библиотеки STL.
    1. Реализовать структуру хранения данных на основе шаблона <vector>, размеры определить при вводе с клавиатуры.
    2. Разработать функции: заполнение структуры хранения исходных данных, вывода структуры хранения.
    3. Выполнить декомпозицию задачи варианта, разработать алгоритм решения. Реализовать функцию, выполняющую задачу и отладить ее.
    4. Разработать программу, демонстрирующую работу всех функций.
14. Составить отчет, отобразив в нем описание выполнения всех этапов разработки, тестирования и код всей программы со скриншотами результатов тестирования.

Таблица 1. Упражнения варианта №20.

|  |  |
| --- | --- |
| Упражнение 1 | Дана квадратная матрица. Найти минимальное значение среди элементов, расположенных над главной диагональю. |
| Упражнение 2 | Выбрать три различных точки из множества точек на плоскости так, чтобы была минимальной разность между количеством точек, лежащих внутри и вне треугольника с вершинами в выбранных точках. |

# **Решение**

Двумерный массив предназначен для представления структур данных элементы которых подчиняются бинарным (2-арным)отношениям. Каждый элемент структуры данных располагается в массиве в ячейке в соответствии с двумя параметрами (их еще называют координатами)- номер строки и номер столбца.  Например, для хранения матрицы размером 3×4 очень удобно использовать двумерный массив.

**3.1. Статический двумерный массив**

Определение float y[3][3]; можно разделить на составные части: float y[3] – можно прочитать так: у – указатель на массив из трех элементов, а вторая скобка [3] вновь указывает на массив из трех элементов – т.е. снова на массив (а массив и указатель это одно и то же).

В язык Си имя двумерного массива это указатель на массив указателей на строки.

Имя двумерного массива – это указатель на массив указателей. Двумерный массив можно рассматривать как одномерный массив, элементы которого - одномерные массивы.

**3.2. Динамический двумерный массив**

int \*\*x - такая запись называется косвенным указателем, т.к. х является указателем на ячейку, которая так же является указателем. Двойной указатель используется для определения динамического двумерного массива.

**3.3. Контейнер вектор и двумерный массив**

Если считать, что одна строка — это vector<int>, то двумерный массив — это вектор элементов типа vector<int>, то есть его нужно объявлять как vector<vector<int> >. При этом по стандарту языка C++ до 2011 года, в конце определения между двумя символами “<” должен стоять пробел, начиная с C++11 этот пробел необязателен.

Когда вектор объявляется внутри другого вектора, этот вектор называется 2-мерным вектором, который работает как 2-мерный массив. Двумерный вектор содержит несколько строк, каждая из которых является другим вектором.

**3.4. Реализация упражнения 1 на статическом двумерном массиве**

|  |
| --- |
| void gen\_stat\_matrix(int m[ROWS][COLS], int min = -25, int max = 25) //генератор статич матрицы  {  std::random\_device r;  std::default\_random\_engine e1(r());  std::uniform\_int\_distribution<int> uniform\_dist(min, max);  for (int i = 0; i < ROWS; i++)  {  for (int j = 0; j < COLS; j++)  m[i][j] = uniform\_dist(e1);  }  }  void print\_matrix(const int m[ROWS][COLS]) //функция вывода матрицы на экран статит  {  for (int i = 0; i < ROWS; i++)  {  for (int j = 0; j < COLS; j++) {  std::cout << std::setw(5) << m[i][j];  if (j != ROWS)  std::cout << " ";  }  if (i != COLS)  std::cout << std::endl;  }  }  int thelowestofhighest\_stat(int m[ROWS][COLS]) //определение минимального элекмента над главной диагональю  {  int min = m[0][1];  int max = m[1][0];  int imin = 0, jmin = 0, imax = 0, jmax = 0;  for (int i = 0; i < ROWS; i++)  for (int j = 0; j < ROWS; j++)  {  if (i < j) //условие главной диагонали  {  if (min > (m[i][j]))  {  min = m[i][j];  imin = i + 1;  jmin = j + 1;  }  }  }  return min;  }  // реализация для статического массива. (для динамического массива реализация аналогична)  void fill\_stat\_and\_dyn\_matrix(int m[ROWS][COLS]) //ввод матрицы с клавиатуры  {  for (int i = 0; i < ROWS; i++)  {  for (int j = 0; j < COLS; j++)  std::cin >> m[i][j];  }  } |

Сначала вызывается функция gen\_stat\_matrix, в которую передаются статический массив и диапазон значений допустимых для его элементов. В функции статическая матрица заполняется случайными значениями.

Функция print\_matrix предназначена для выведения статической матрицы на экран. В функцию передаются непосредственно матрица, количество строк и количество столбцов.

Функция fill\_stat\_and\_dyn\_matrix предназначена для заполнения статического массива с клавиатуры.

В функции thelowestofhighest\_stat ищется минимальный среди элементов расположенных над главной диагональю матрицы. В функцию передаются непосредственно матрица.

**3.5. Реализация упражнения 1 на двумерном динамическом массиве.**

|  |
| --- |
| int\*\* gen\_dyn\_matrix(int rows, int cols, int min = -25, int max = 25) //генератор динам матрицы  {  std::random\_device r;  std::default\_random\_engine e1(r());  std::uniform\_int\_distribution<int> uniform\_dist(min, max);  int\*\* m = (int\*\*)malloc(sizeof(int) \* rows);  for (int i = 0; i < rows; i++)  {  m[i] = (int\*)malloc(sizeof(int) \* cols);  for (int j = 0; j < cols; j++)  m[i][j] = uniform\_dist(e1);  }  return m;  }  void print\_dyn\_matrix(int\*\* m, int rows, int cols) // вывод динам матрицы на экран  {  for (int i = 0; i < rows; i++)  {  for (int j = 0; j < cols; j++) {  std::cout << std::setw(5) << m[i][j];  if (j != rows)  std::cout << " ";  }  if (i != cols)  std::cout << std::endl;  }  }  int thelowestofhighest\_dyn(int\*\* m)  {  int min = m[0][1];  int max = m[1][0];  int imin = 0, jmin = 0, imax = 0, jmax = 0;  for (int i = 0; i < ROWS; i++)  for (int j = 0; j < ROWS; j++)  {  if (i < j) //условие главной диагонали  {  if (min > (m[i][j]))  {  min = m[i][j];  imin = i + 1;  jmin = j + 1;  }  }  }  return min;  } |

Сначала вызывается функция gen\_dyn\_matrix, в которую передаются динамический массив и диапазон значений допустимых для его элементов. В функции матрица заполняется случайными значениями.

Функция print\_matrix предназначена для выведения динамической матрицы на экран. В функцию передаются непосредственно матрица, количество строк и количество столбцов.

В функции thelowestofhighest\_dyn ищется минимальный среди элементов расположенных над главной диагональю динамической матрицы. В функцию передаются непосредственно матрица.

**3.6. Реализация упражнения 2 на двумерном векторе.**

|  |
| --- |
| const int n = 16;  point p[n] = { {0,0},{0,1},{0,2},{0,3},{1,0},{1,1},{1,2},{1,3},{2,0},{2,1},{2,2},{2,3},{3,0},{3,1},{3,2},{3,3} };  int a, b, c;  int difference = n;  for (int i = 0; i < n - 2; i++) {  for (int j = i + 1; j < n - 1; j++) {  for (int k = j + 1; k < n; k++) {  int tmp = 0;  for (int x = 0; x < n; x++) {  if ((x - i) \* (x - j) \* (x - k)) {  if (inTriangle(p[x], p[i], p[j], p[k])) {  tmp++;  }  else {  tmp--;  }  }  else {  continue;  }  }  tmp = abs(tmp);  if (tmp < difference) {  difference = tmp;  a = i;  b = j;  c = k;  }  if (tmp == 0) {  goto Find;  }  }  }  }  Find:  std::cout << a << ' ' << b << ' ' << c; |

Для решения задачи функции не использовались, программа реализуется в основной функции main()

Принцип реализации в том, что каждую точку нужно проверить на предмет вхождения в периметр треугольника с выбранными вершинами. Для этого нужно вычислить площадь треугольника, сумму площадей треугольников образованных выбранной точкой и двумя вершинами основного треугольника (все варианты). Если площадь основного треугольника меньше суммы, точка лежит за пределами. Нам нужно подобрать такие вершины основного треугольника, чтобы количество точек внутри и снаружи треугольника было примерно одинаковым.

**3.7. Интерфейс**

Изначально пользователю предоставляется выбор задачи, для задачи №1 есть две реализации: на двумерном статическом и двумерном динамическом массивах. Пользователь может заполнить матрицу как вручную, так и случайными числами. В зависимости от выбора режима заполнения матрицы, пользователю либо придётся сначала ввести элементы матрицы, либо же выведется сформированная матрица. В конце выводится ответ на выбранную задачу.

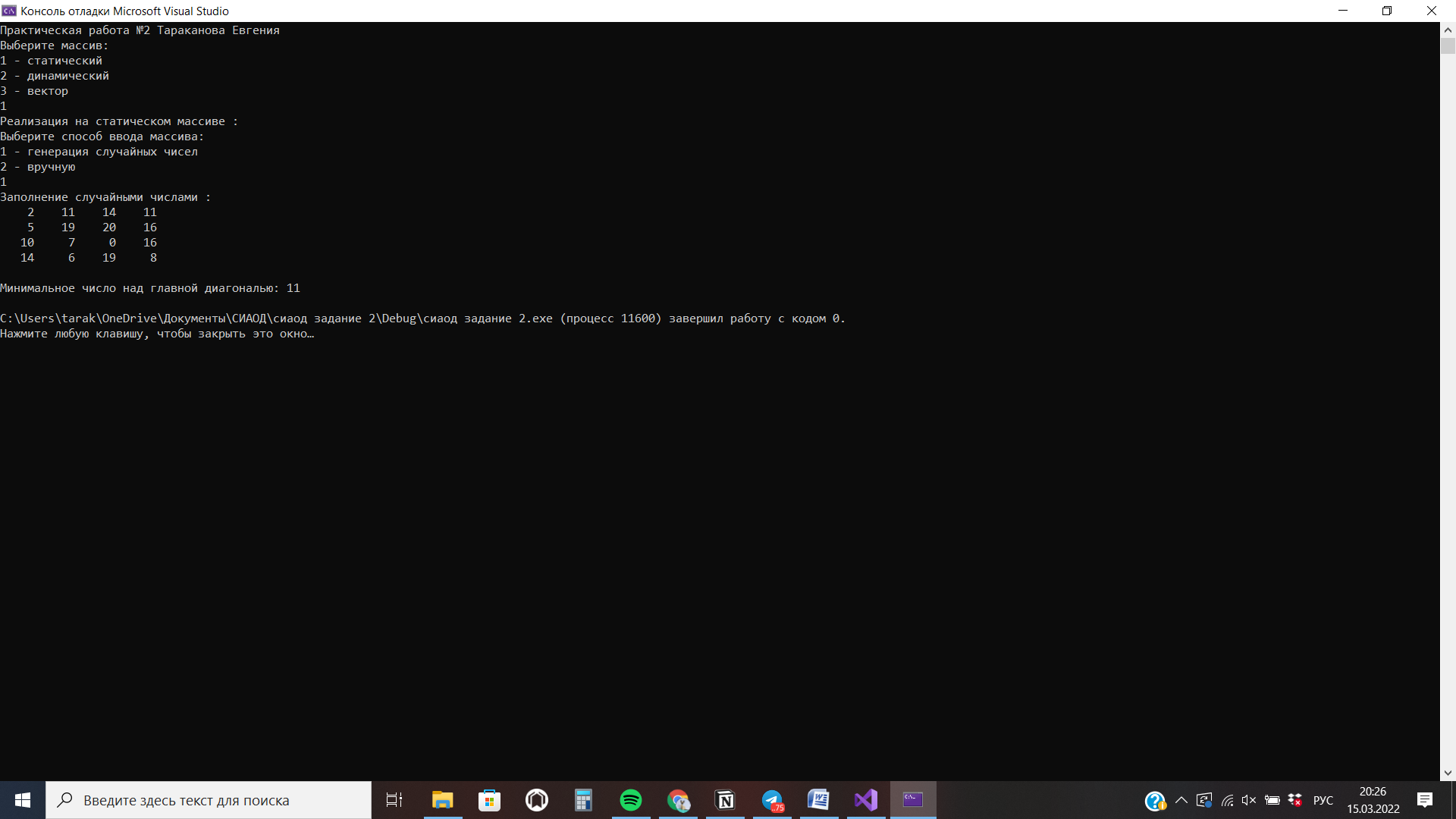


Рисунок 1. Интерфейс

# **Тестирование**

1. Тест №1. Упражнение 1 на статическом массиве

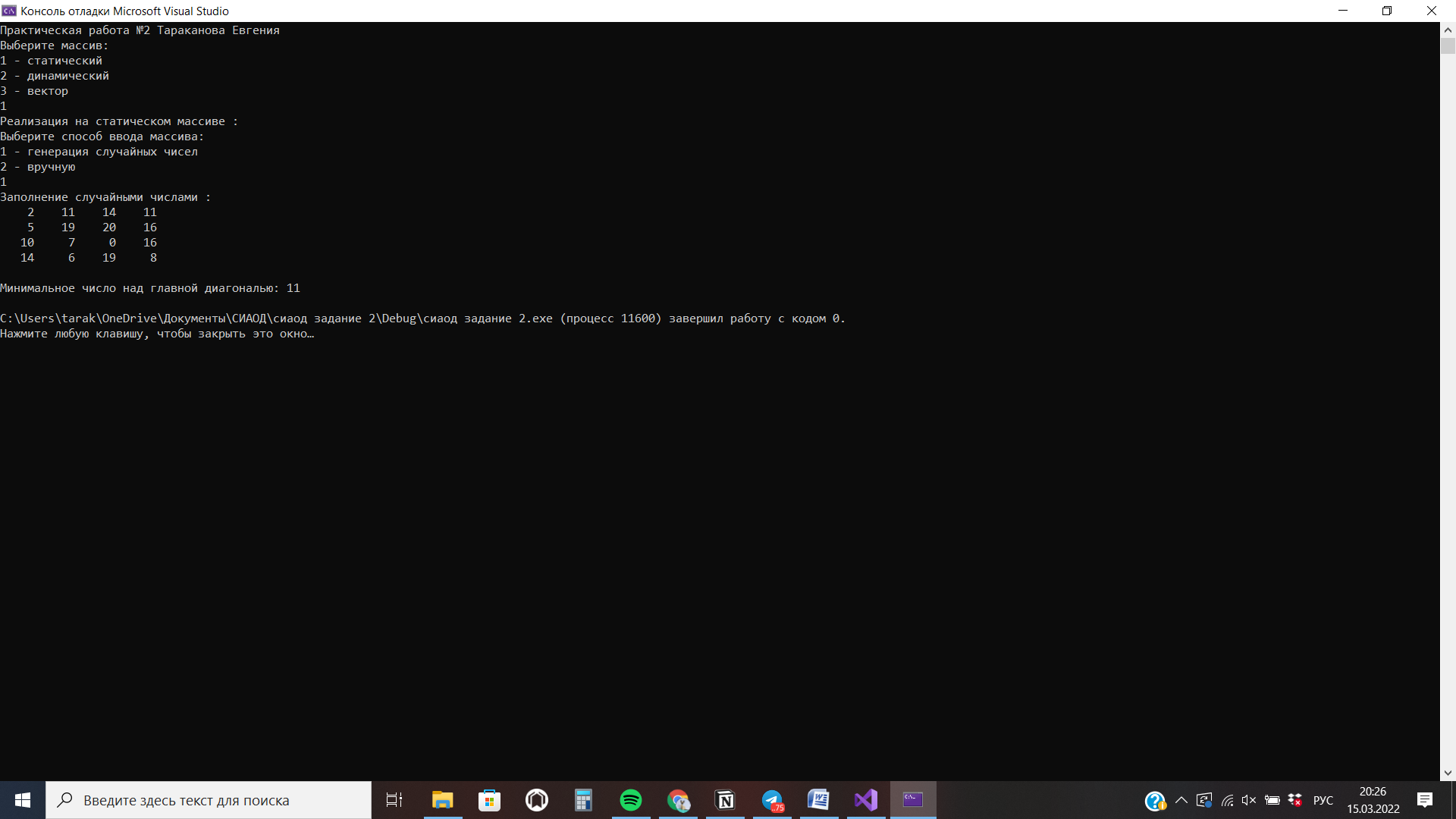


Рисунок 2. Тест № 1

1. Тест №2. Упражнение 1 на динамическом массиве. Программа работает корректно.

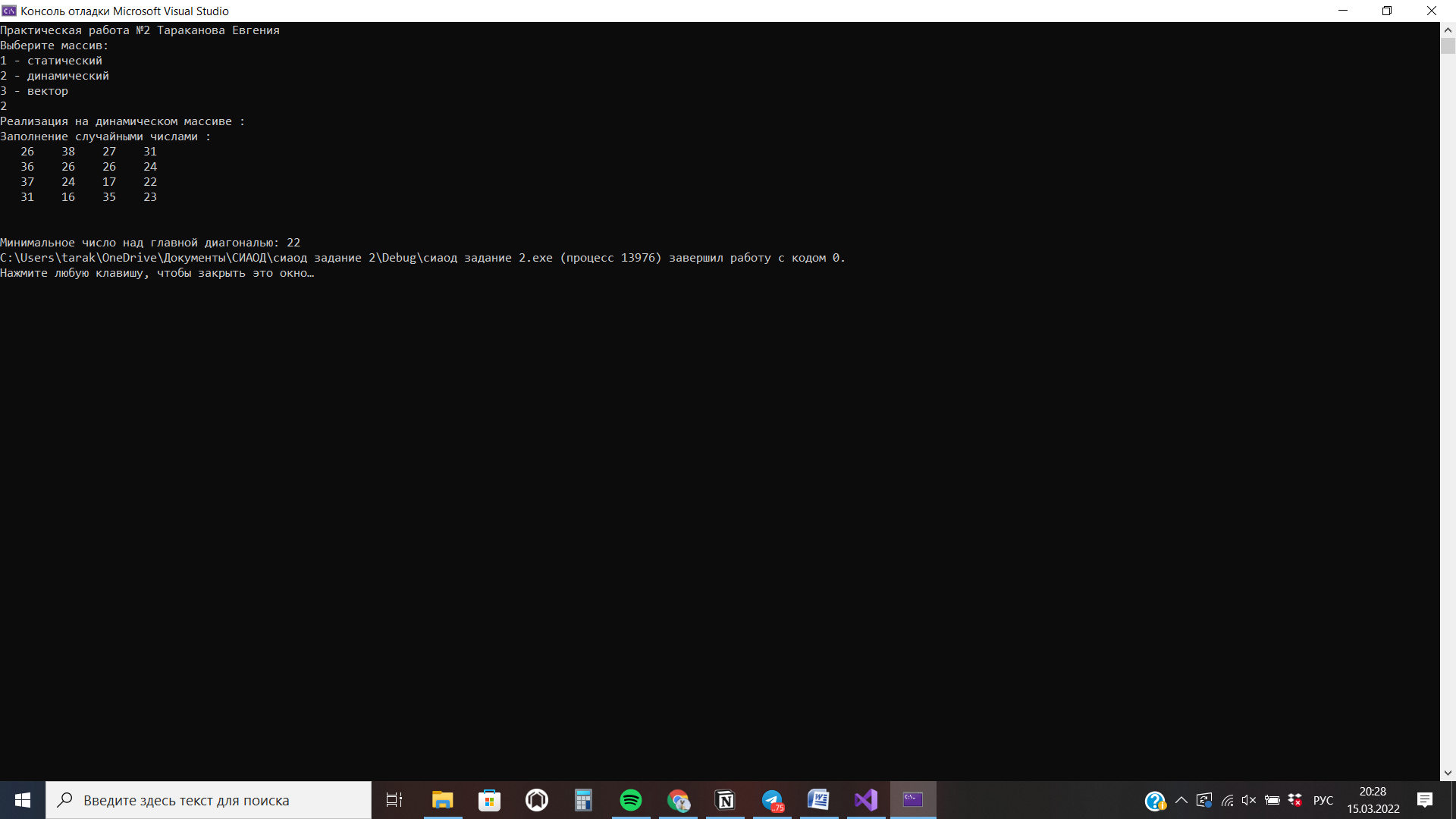


Рисунок 3. Тест № 2

1. Тест №3. Упражнение 2 на векторе. Программа работает корректно.

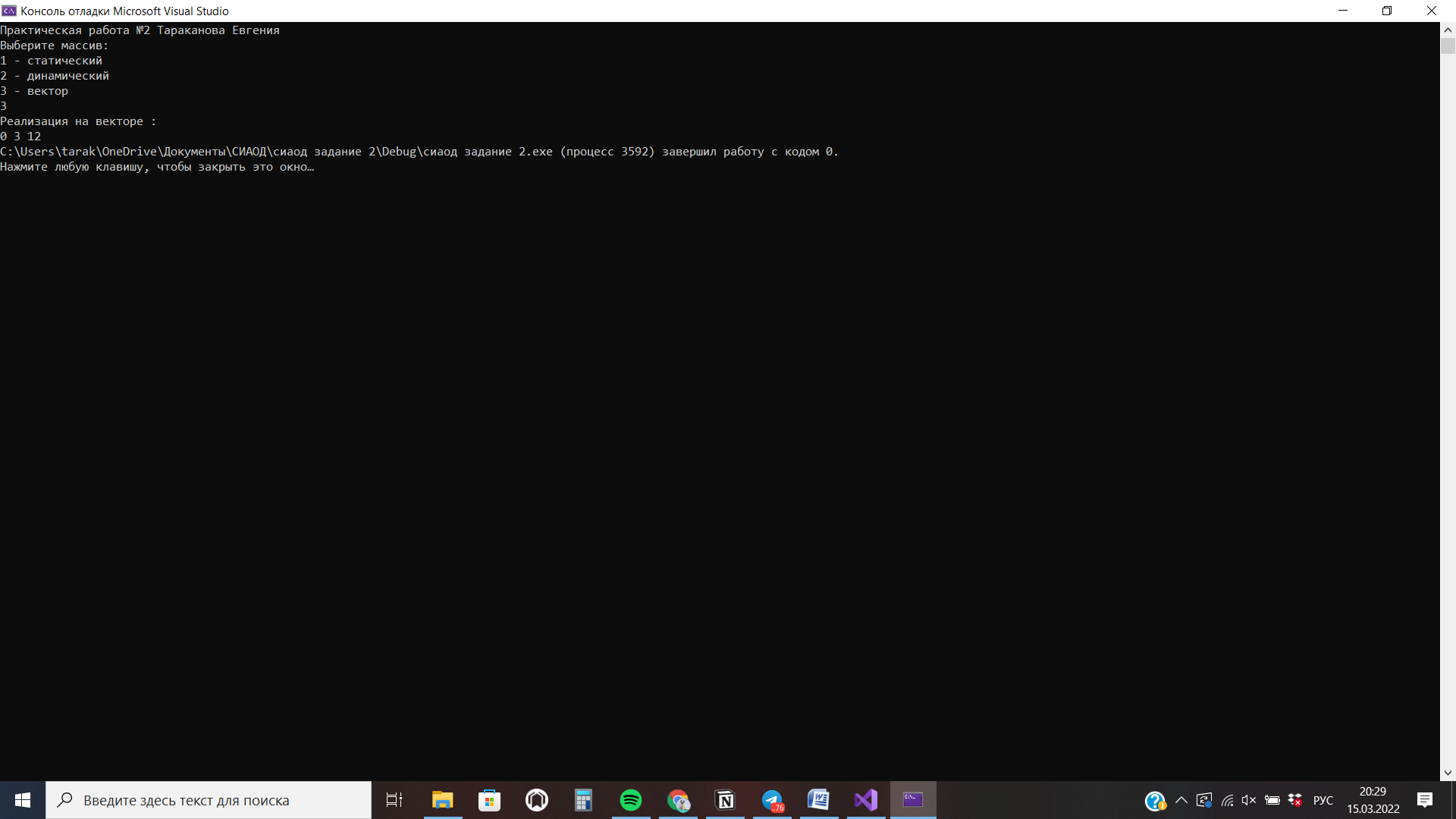


Рисунок 4. Тест № 3

# **Вывод**

В результате выполнения работы я получила навыки по определению двумерного массива для структуры данных задачи, а также по разработке алгоритмов операций на двумерном массиве в соответствии с задачей.

# **Исходный код программы**

|  |
| --- |
| // Дана квадратная матрица. Найти минимальное значение среди элементов, расположенных над главной диагональю.  //Выбрать три различных точки из множества точек на плоскости так, чтобы была минимальной разность между количеством точек,  //...лежащих внутри и вне треугольника с вершинами в выбранных точках.  #define ROWS 4  #define COLS 4  #include <iostream>  #include <iomanip>  #include <random>  #include <vector>  #include <set>  #include <algorithm>  #include <cstdlib>  #include <ctime>  #include <cmath>  using namespace std;  void gen\_stat\_matrix(int m[ROWS][COLS], int min = -25, int max = 25) //генератор статич матрицы  {  std::random\_device r;  std::default\_random\_engine e1(r());  std::uniform\_int\_distribution<int> uniform\_dist(min, max);  for (int i = 0; i < ROWS; i++)  {  for (int j = 0; j < COLS; j++)  m[i][j] = uniform\_dist(e1);  }  }  int\*\* gen\_dyn\_matrix(int rows, int cols, int min = -25, int max = 25) //генератор динам матрицы  {  std::random\_device r;  std::default\_random\_engine e1(r());  std::uniform\_int\_distribution<int> uniform\_dist(min, max);  int\*\* m = (int\*\*)malloc(sizeof(int) \* rows);  for (int i = 0; i < rows; i++)  {  m[i] = (int\*)malloc(sizeof(int) \* cols);  for (int j = 0; j < cols; j++)  m[i][j] = uniform\_dist(e1);  }  return m;  }  void print\_matrix(const int m[ROWS][COLS]) //функция вывода матрицы на экран статит  {  for (int i = 0; i < ROWS; i++)  {  for (int j = 0; j < COLS; j++) {  std::cout << std::setw(5) << m[i][j];  if (j != ROWS)  std::cout << " ";  }  if (i != COLS)  std::cout << std::endl;  }  }  void print\_dyn\_matrix(int\*\* m, int rows, int cols) // вывод динам матрицы на экран  {  for (int i = 0; i < rows; i++)  {  for (int j = 0; j < cols; j++) {  std::cout << std::setw(5) << m[i][j];  if (j != rows)  std::cout << " ";  }  if (i != cols)  std::cout << std::endl;  }  }  int thelowestofhighest\_stat(int m[ROWS][COLS])  {  int min = m[0][1];  int max = m[1][0];  int imin = 0, jmin = 0, imax = 0, jmax = 0;  for (int i = 0; i < ROWS; i++)  for (int j = 0; j < ROWS; j++)  {  if (i < j) //условие главной диагонали  {  if (min > (m[i][j]))  {  min = m[i][j];  imin = i + 1;  jmin = j + 1;  }  }  }  return min;  }  int thelowestofhighest\_dyn(int\*\* m)  {  int min = m[0][1];  int max = m[1][0];  int imin = 0, jmin = 0, imax = 0, jmax = 0;  for (int i = 0; i < ROWS; i++)  for (int j = 0; j < ROWS; j++)  {  if (i < j) //условие главной диагонали  {  if (min > (m[i][j]))  {  min = m[i][j];  imin = i + 1;  jmin = j + 1;  }  }  }  return min;  }  // реализация для статического массива. (для динамического массива реализация аналогична)  void fill\_stat\_and\_dyn\_matrix(int m[ROWS][COLS]) //ввод матрицы с клавиатуры  {  for (int i = 0; i < ROWS; i++)  {  for (int j = 0; j < COLS; j++)  std::cin >> m[i][j];  }  }  void fill\_vector\_matrix(std::vector<std::vector<char> >& vec, int rows = 20, int cols = -1) //заполнение двумерного вектора точками  {  if (cols == -1)  cols = rows;  for (int r = 0; r < rows; r++)  {  std::vector<char> row;  for (int c = 0; c < cols; c++)  row.push\_back('.');  vec.push\_back(row);  }  }  struct point  {  double x;  double y;  };  double getArea(const point& a, const point& b, const point& c) //функция вычисляет плозадь треугольника вершин которого заданы точнами в структуре point  {  return fabs((a.x - c.x) \* (b.y - c.y) - (b.x - c.x) \* (a.y - c.y)); //fabs вычисляет абсолютное значение  }  bool inTriangle(const point& p, const point& a, const point& b, const point& c)  {  return (getArea(a, b, c) >= (getArea(p, b, c) + getArea(a, p, c) + getArea(a, b, p))); //показывает больше ли плозадь основного треугольника, чем суммы побочных  } //если да, то точка Р лежит за пределами треугольника. Нам нужно чтобы кол-во внутри было примерно равно за пределами  int main()  {  setlocale(LC\_ALL, "");  int zello;  cout << "Практическая работа №2 Тараканова Евгения " << endl;  cout << "Выберите массив:" << endl << "1 - статический" << endl << "2 - динамический" << endl << "3 - вектор" << endl;  int mode;  do  {  cin >> mode;  if (mode < 1 || mode > 3) cout << "Такого режима нет. Введите число от 1 до 3: ";  } while (mode < 1 || mode > 3);  switch (mode)  {  case 1: //!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! C A S E 1  {//Статический массив  cout << "Реализация на статическом массиве : " << endl;  int m1[ROWS][COLS];  int result[ROWS][COLS];  cout << "Выберите способ ввода массива:" << endl << "1 - генерация случайных чисел" << endl << "2 - вручную" << endl;  int mode1;  do  {  cin >> mode1;  if (mode1 < 1 || mode1 > 2) cout << "Такого режима нет. Введите число 1 или 2: ";  } while (mode1 < 1 || mode1 > 2);  switch (mode1)  {  case 1:  {  cout << "Заполнение случайными числами : " << endl;  gen\_stat\_matrix(m1, 0, 20); //генерация матрицы м1 случайными значениями от 0 до 20  print\_matrix(m1); // вывод матрицы на экран  cout << endl;  zello = thelowestofhighest\_stat(m1); //нашли минимальное значение над главной диагональю  cout << "Минимальное число над главной диагональю: " << zello << endl;  break;  }  case 2:  {  cout << "Заполнение числами с клавиатуры: " << endl;  fill\_stat\_and\_dyn\_matrix(m1);  print\_matrix(m1); // вывод матрицы на экран  cout << endl;  int zello = thelowestofhighest\_stat(m1); //нашли минимальное значение над главной диагональю  cout << zello << endl;  break;  }  }  break;  }  case 2:  { //Динамический массив  int rows = 4;  int cols = 4;  cout << "Реализация на динамическом массиве : " << endl;  int mode1;  cout << "Заполнение случайными числами : " << endl;  int\*\* m1 = gen\_dyn\_matrix(rows, cols, 1, 40);  print\_dyn\_matrix(m1, rows, cols); // вывод матрицы на экран  cout << endl;  int zello = thelowestofhighest\_dyn(m1); //нашли минимальное значение над главной диагональю  cout << endl << "Минимальное число над главной диагональю: " << zello;  break;  }  case 3:  {//вектор  cout << "Реализация на векторе : " << endl;  const int n = 16;  point p[n] = { {0,0},{0,1},{0,2},{0,3},{1,0},{1,1},{1,2},{1,3},{2,0},{2,1},{2,2},{2,3},{3,0},{3,1},{3,2},{3,3} };  int a, b, c;  int difference = n;  for (int i = 0; i < n - 2; i++) {  for (int j = i + 1; j < n - 1; j++) {  for (int k = j + 1; k < n; k++) {  int tmp = 0;  for (int x = 0; x < n; x++) {  if ((x - i) \* (x - j) \* (x - k)) {  if (inTriangle(p[x], p[i], p[j], p[k])) {  tmp++;  }  else {  tmp--;  }  }  else {  continue;  }  }  tmp = abs(tmp);  if (tmp < difference) {  difference = tmp;  a = i;  b = j;  c = k;  }  if (tmp == 0) {  goto Find;  }  }  }  }  Find:  std::cout << a << ' ' << b << ' ' << c;    break;  }  }  } |