|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  |  |  | | 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-07-22 | Козлов К.И. |
| Принял преподаватель | Филатов А.С. |

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

Москва 2023

# **Цель работы**

- Получение навыков по определению двумерного массива для структуры данных задачи

- Получение навыков 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) динамическим массивом.
   1. Размеры массива должны определяться пользователем.
   2. Двумерный массив определить как двойной указатель и выполнить его создание операцией new.
   3. Разработать функции по реализации алгоритмов заполнения матрицы: с клавиатуры, датчиком случайных чисел. Разработать функции по реализации алгоритма вывода массива на экран построчно.
   4. Выполнить декомпозицию задачи варианта, разработать алгоритм решения. Реализовать функцию, выполняющую задачу и отладить ее.
   5. Разработать программу, демонстрирующую работу всех функций.
2. 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.

Требования к упражнениям:

Вариант №9. Условие задания:

|  |  |
| --- | --- |
| Упражнение 1 | Дана квадратная матрица. Найти минимальное значение среди элементов, расположенных под главной диагональю. |
| Упражнение 2 | 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. |

# **Решение**

В данной практической работе была проведена работа с разными видами массивов: динамический двумерный массив и контейнером <vector>.

Двумерный массив - это одномерный массив, элементами которого являются одномерные массивы.

Наглядно двумерный массив удобно представлять в виде таблицы, в которой n строк и m столбцов, а под ячейкой таблицы, стоящей в i-й строке и j-м столбце понимают некоторый элемент массива a[i][j].

Для всех элементов массива общим является имя, но у каждого элемента массива уникальным является индекс.

Динамический массив – это структура данных, которая создается во время выполнения программы. Возможность изменения размера отличает динамический массив от статического, размер которого задаётся на момент компиляции программы. Динамические массивы дают возможность более гибкой работы с данными, так как позволяют не прогнозировать хранимые объёмы данных, а регулировать размер массива в соответствии с реально необходимыми объёмами.

Динамический массив в языке C++ определяется через указатель.

Указатель - переменная, в которой хранится адрес памяти объекта. Указатели широко используются в C++ для трех основных целей:

А поскольку наш массив является массивом массивов, то в динамической его реализации это массив указателей на одномерные массивы.

В стандартной библиотеке C++ вектором (std::vector) называется динамический массив, обеспечивающий быстрое добавление новых элементов в конец и меняющий свой размер при необходимости. Вектор гарантирует отсутствие утечек памяти- для итерации элементов в массивах или других структурах данных.

Контейнер <vector> - современная реализация динамических массивов.

Векторы - это реализация динамического массива в C++, то есть такого массива, чья длина меняется в ходе программы.

Конкретный тип данных определяется во время объявления вектора. Если размер вектора не определен, вектор называется пустым вектором. Размер вектора можно изменить, используя различные методы или инициализируя вектор.

Двумерный вектор содержит несколько строк, каждая из которых является другим вектором.

Для решения первого упражнения было написано несколько функций: EnterMatrixDynamic(), EnterMatrixStatic; MinNumBelowMainDiagDynamic(), MinNumBelowMainDiagStatic. Данные функции реализуют ввод матрицы для статического двумерного массива и статического двумерного массива.

Функция EnterMatrixDynamic() имеет два аргумента: массив указателей на целочисленные массивы a и целочисленную переменную n. Функция сначала выделяет помять под все массивы внутри основного двумерного, а затем по индексам заполняет его значениями, введёнными пользователем.

|  |
| --- |
| void EnterMatrixDynamic(int\*\* a, int n)  {  for (int i = 0; i < n; i++)  {  a[i] = (int\*)malloc(sizeof(int) \* n);  }  for (int i = 0; i < n; i++)  {  for (int j = 0; j < n; j++)  {  cin >> a[i][j];  }  }  } |

Функция EnterMatrixStatic() имеет два аргумента: статический двумерный вектор a и целочисленную переменную n. Эта функция также реализует пользовательский ввод значений в массив.

|  |
| --- |
| void EnterMatrixStatic(int a[50][50], int n)  {  for (int i = 0; i < n; i++)  {  for (int j = 0; j < n; j++)  {  cin >> a[i][j];  }  }  } |

Функция MinNumBelowMainDiagDynamic() имеет такие же аргументы, как и функция EnterMatrixDynamic(), однако отлична по функционалу. Данная функция возвращает значение наименьшего элемента под главной диагональю получившейся матрицы (двумерного динамического массива).

|  |
| --- |
| int MinNumBelowMainDiagDynamic(int\*\* a, int n)  {  int Min = a[0][0];  for (int i = 1; i < n; i++)  {  for (int j = 0; j < i; j++)  {  if (Min > a[i][j])  {  Min = a[i][j];  }  }  }  return (Min);  } |

Функция MinNumBelowMainDiagStatic() имеет такие же аргументы, как и функция EnterMatrixStatic(), однако отлична по функционалу. Данная функция возвращает значение наименьшего элемента под главной диагональю получившейся матрицы (двумерного статического массива).

|  |
| --- |
| int MinNumBelowMainDiagStatic(int a[50][50], int n)  {  int Min = a[0][0];  for (int i = 1; i < n; i++)  {  for (int j = 0; j < i; j++)  {  if (Min > a[i][j])  {  Min = a[i][j];  }  }  }  return (Min);  } |

Для решения второго упражнения были написаны следующие функции: входит в библиотеку <algorithm>), Dots(), ShowTriangle(). Также была использована фунция sort(), которая отсортировала массивы внутри двумернрго по первому элементу. Исходя из задачи, все треугольники можно составить, если найти самую левую точку и далее брать тройки точек, где первая зафиксирована на самой левой, а последующие являются ближайшими к ней.

Функця Dots() имеет следующие аргументы: двумерный целочисленный veсtor a и целочисленную переменную n. Функция реализует пользовательский ввод точек.

|  |
| --- |
| vector <vector <int>> Dots(vector < vector < int>> a, int n)  {  for (int i = 0; i < n; i++)  {  for (int j = 0; j < 2; j++)  {  cin >> a[i][j];  }  }  return (a);  } |

Функция ShowTriangle() имеет такие же аргументы, как и Dots(). Эта функция выводит все получившиеся треугольники.

|  |
| --- |
| void ShowTriangles(vector <vector < int>> a, int n)  {  for (int i = 0; i < n - 2; i += 3)  {  cout << "Triangle = " << a[i][0] << ';' << a[i][1] << ' ' << a[i + 1][0] << ';' << a[i + 1][1] << ' ' << a[i + 2][0] << ';' << a[i + 2][1] << endl;  }  cout << endl;  } |

Функция comp(), имеющая аргументы в виде двух векторов, возвращает логическое значение выражения. Она выступает компаратором для функции sort() так, чтоб sort отсортировала все векторы по возрастанию нулевого значения, т.е. точки по ординате.

|  |
| --- |
| bool comp(vector <int> a, vector <int> b)  {  return a[0] < b[0];  } |

# **Тестирование**

Покажем сообщения пользователю при запуске программы.



Рисунок 1. Интерфейс программы

Протестируем программу. Для этого выберем работу с динамическим массивом, нажав 1, введем размер квадратной матрицы и введем ее значения. Программа корректно находит меньший элемент строго под диагональю

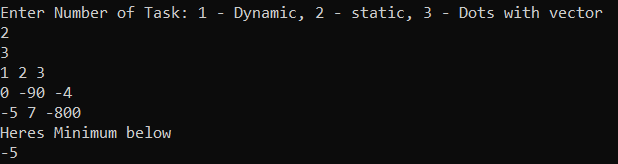


Рисунок 2. Нахождение минимального элемента под главной диагональю

Как видно, программа корректно находит минимальный элемент под главной диагональю.

Протестируем работу с двумерным вектором.

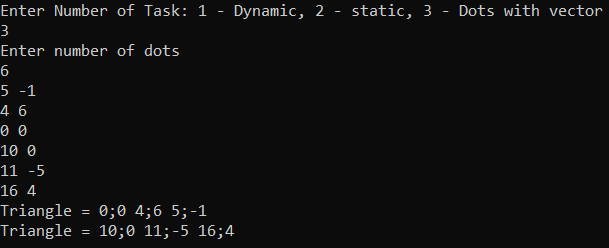


Рисунок 3. Выделение треугольников из множества точек

Ниже приведена графическая иллюстрация того, какие треугольник по заданным точкам вывела программа.

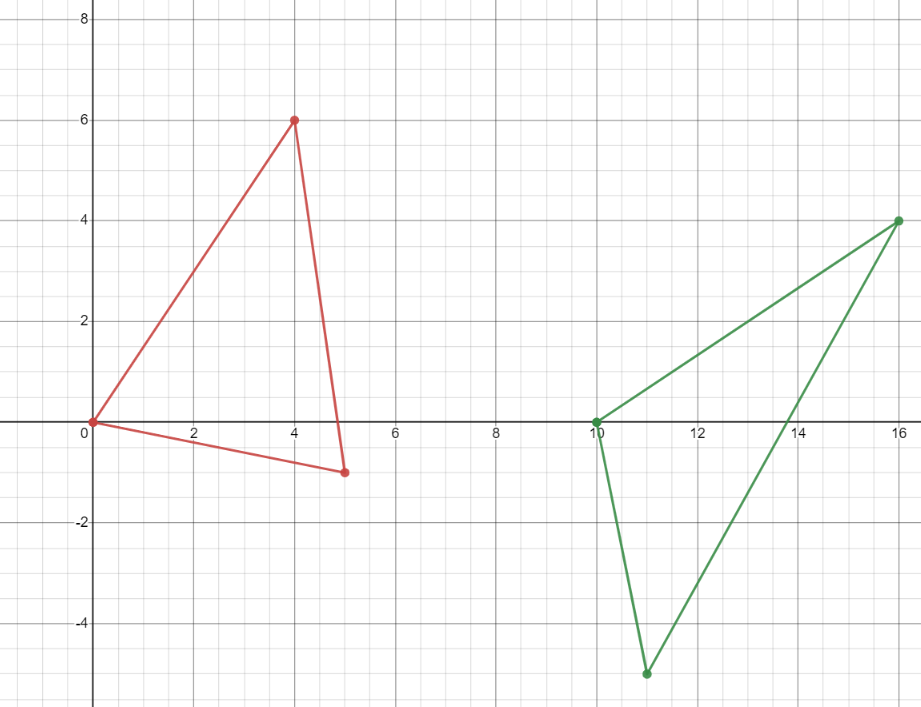


Рисунок 5. Графическая иллюстрация

# **5. Выводы**

В результате выполнения работы я:

1. Освоил алгоритмы работы с двумерными статическими, динамическими массивами, с векторами.
2. Научился проводить операции над векторами и разными видами массивов.
3. Реализовал работу с массивами и векторами на языке программирования C++.
4. Приобрел навыки по работе с массивами и векторами.

# **6. Исходный код программы**

|  |
| --- |
| #include <iostream>  #include <vector>  #include <cmath>  #include <algorithm>  using namespace std;  void EnterMatrixDynamic(int\*\* a, int n)  {  for (int i = 0; i < n; i++)  {  a[i] = (int\*)malloc(sizeof(int) \* n);  }  for (int i = 0; i < n; i++)  {  for (int j = 0; j < n; j++)  {  cin >> a[i][j];  }  }  }  void EnterMatrixStatic(int a[50][50], int n)  {  for (int i = 0; i < n; i++)  {  for (int j = 0; j < n; j++)  {  cin >> a[i][j];  }  }  }  vector <vector <int>> Dots(vector < vector < int>> a, int n)  {  for (int i = 0; i < n; i++)  {  for (int j = 0; j < 2; j++)  {  cin >> a[i][j];  }  }  return (a);  }  void Show(vector <vector < int>> a, int n)  {  for (int i = 0; i < n; i++)  {  for (int j = 0; j < 2; j++)  {  cout << a[i][j] << ' ';  }  cout << endl;  }  }  void ShowTriangles(vector <vector < int>> a, int n)  {  for (int i = 0; i < n - 2; i += 3)  {  cout << "Triangle = " << a[i][0] << ';' << a[i][1] << ' ' << a[i + 1][0] << ';' << a[i + 1][1] << ' ' << a[i + 2][0] << ';' << a[i + 2][1] << endl;  }  cout << endl;  }  int MinNumBelowMainDiagDynamic(int\*\* a, int n)  {  int Min = a[0][0];  for (int i = 1; i < n; i++)  {  for (int j = 0; j < i; j++)  {  if (Min > a[i][j])  {  Min = a[i][j];  }  }  }  return (Min);  }  int MinNumBelowMainDiagStatic(int a[50][50], int n)  {  int Min = a[0][0];  for (int i = 1; i < n; i++)  {  for (int j = 0; j < i; j++)  {  if (Min > a[i][j])  {  Min = a[i][j];  }  }  }  return (Min);  }  bool comp(vector <int> a, vector <int> b)  {  return a[0] < b[0];  }  int main()  {  int Choise;  cout << "Start of program" << endl;  do  {  cout << "Enter Number of Task: 1 - Dynamic, 2 - static, 3 - Dots with vector" << endl;  cin >> Choise;  if (Choise == 1)  {  int n;  int\*\* a;  do  {  cout << "Enter n for Matrix" << endl;  cin >> n;  } while (n <= 0);  a = (int\*\*)malloc(sizeof(int\*) \* n);  EnterMatrixDynamic(a, n);  cout << "Heres Minimum below" << endl;  cout << MinNumBelowMainDiagDynamic(a, n);  cout << endl;  }  else if (Choise == 2)  {  int n;  cin >> n;  int a[50][50];  EnterMatrixStatic(a, n);  cout << "Heres Minimum below" << endl;  cout << MinNumBelowMainDiagStatic(a, n);  cout << endl;  }  else if (Choise == 3)  {  cout << "Enter number of dots" << endl;  int n;  cin >> n;  vector < vector <int> > a(n, vector <int>(2));  a = Dots(a, n);  sort(begin(a), end(a), comp);  ShowTriangles(a, n);  }  } while (Choise == 1 || Choise == 2 || Choise == 3);  } |