МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

### федеральное государственное автономное образовательное учреждение высшего образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

(ГУАП)

КАФЕДРА ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ И СЕТЕЙ

Преподаватель

канд. техн. наук, доцент Л.Н. Бариков

Отчёт

по лабораторной работе №8

по дисциплине ОСНОВЫ ПРОГРАММИРОВАНИЯ

на тему: «Статические двумерные массивы»

Работу выполнил

студент гр. 4141 В.С. Сыворотнев

Санкт-Петербург

2022

***Цель лабораторной работы:*** *изучение структурной организации массивов и способов доступа к их элементам; совершенствование навыков процедурного программирования на языке C/C++ при решении задач обработки двумерных статических массивов (матриц).*

***Задание на программирование:*** *используя технологию процедурного программирования, разработать программу обработки двумерных статических массивов (матриц) в соответствии с индивидуальным заданием.*

**Вариант № 31.**

*В заданной квадратной матрице размера (2n+1)×(2n+1) найти элементы с наибольшим по абсолютной величине и с наименьшим по абсолютной величине значениями. Если хотя бы один из них находится в области правее главной и побочной диагоналей найти сумму значений элементов, лежащих ниже главной диагонали. В противном случае поменять найденные значения местами.*

***Математическая модель решения***

Решение задачи начинается с ввода исходных данных. В формулировке задачи сказано, что исходная матрица квадратная с **нечётным** числом (2*n + 1*) строк и столбцов. Следовательно, прежде всего, необходимо ввести значение (*n*) половины размера квадратной матрицы.

После этого необходимо задать значения всех элементов матрицы. Доступ к элементам двумерного массива (матрицы) осуществляется по двум индексам: номеру строки и номеру столбца, на пересечении которых находится данный элемент массива (нумерация строк и столбцов начинается с 0). Поэтому перебираем все строки (с 0-ой до 2*n*). Внутри каждой строки перебираем все столбцы (с 0-ого до 2*n*) и задаём значения элементов, лежащих на пересечении строки и столбца с текущими номерами.

Пусть в качестве примера исходная матрица имеет вид (*n*=4):

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **0** | **1** |  |  | **n** |  |  |  | **2n** |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| **1** | **8** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **1** |
|  | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **1** | **9** |
|  | **6** | **5** | **4** | **3** | **2** | **1** | **1** | **9** | **8** |
| **n** | **5** | **4** | **3** | **2** | **10** | **1** | **9** | **8** | **9** |
|  | **4** | **3** | **2** | **1** | **1** | **9** | **8** | **7** | **6** |
|  | **3** | **2** | **1** | **1** | **9** | **8** | **7** | **6** | **5** |
|  | **2** | **1** | **1** | **9** | **8** | **7** | **6** | **5** | **4** |
| **2n** | **1** | **1** | **9** | **8** | **7** | **6** | **5** | **4** | **3** |

Теперь можно приступать к решению задачи. Вначале находим наибольшее и наименьшее по абсолютной величине значения, затем проверяем, принадлежит ли хотя бы один их них области праве главной и побочной диагонали (оранжевая область), условие принадлежности: i<j и i>2\*n-j. Если принадлежит, то находим сумму нижней диагонали, перебирая номера строк с 0 до 2*n*, осуществляем суммирование значений элементов, принадлежащих области ниже главной диагонали (синяя область), условие принадлежности i>j.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **0** | **1** |  |  | **n** |  |  |  | **2n** |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| **1** | **8** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **1** |
|  | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **1** | **9** |
|  | **6** | **5** | **4** | **3** | **2** | **1** | **1** | **9** | **8** |
| **n** | **5** | **4** | **3** | **2** | **1** | **10** | **9** | **8** | **9** |
|  | **4** | **3** | **2** | **1** | **1** | **9** | **8** | **7** | **6** |
|  | **3** | **2** | **1** | **1** | **9** | **8** | **7** | **6** | **5** |
|  | **2** | **1** | **1** | **9** | **8** | **7** | **6** | **5** | **4** |
| **2n** | **1** | **1** | **9** | **8** | **7** | **6** | **5** | **4** | **3** |

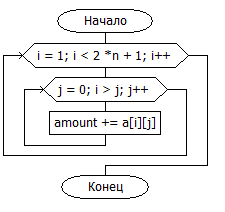
В противном случае меняем наибольшее и наименьшее по абсолютной величине значения местами.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **0** | **1** |  |  | **n** |  |  |  | **2n** |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| **1** | **8** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **1** |
|  | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **1** | **9** |
|  | **6** | **5** | **4** | **3** | **2** | **1** | **1** | **9** | **8** |
| **n** | **5** | **4** | **3** | **2** | **10** | **1** | **9** | **8** | **9** |
|  | **4** | **3** | **2** | **1** | **1** | **9** | **8** | **7** | **6** |
|  | **3** | **2** | **1** | **1** | **9** | **8** | **7** | **6** | **5** |
|  | **2** | **1** | **1** | **9** | **8** | **7** | **6** | **5** | **4** |
| **2n** | **1** | **1** | **9** | **8** | **7** | **6** | **5** | **4** | **3** |

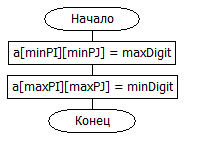
|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **0** | **1** |  |  | **n** |  |  |  | **2n** |
| **0** | **10** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| **1** | **8** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **1** |
|  | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **1** | **9** |
|  | **6** | **5** | **4** | **3** | **2** | **1** | **1** | **9** | **8** |
| **n** | **5** | **4** | **3** | **2** | **1** | **1** | **9** | **8** | **9** |
|  | **4** | **3** | **2** | **1** | **1** | **9** | **8** | **7** | **6** |
|  | **3** | **2** | **1** | **1** | **9** | **8** | **7** | **6** | **5** |
|  | **2** | **1** | **1** | **9** | **8** | **7** | **6** | **5** | **4** |
| **2n** | **1** | **1** | **9** | **8** | **7** | **6** | **5** | **4** | **3** |

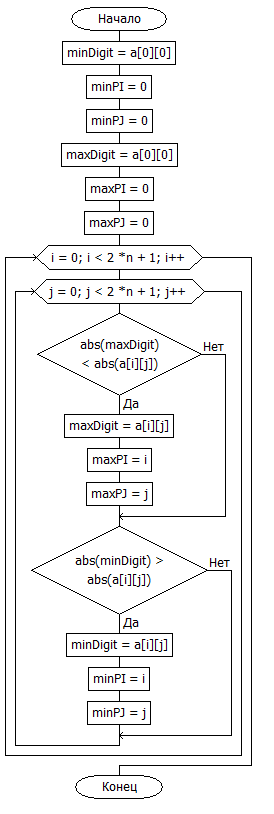
Задача решена.

***Схема алгоритма суммирования элементов матрицы, находящихся ниже главной диагонали (функция sumMatr)***

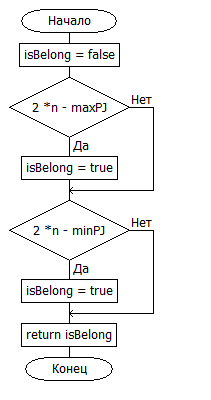


***Схема алгоритма перестановки минимального и максимального значений***

*** (функция swapValues)***

***Схема алгоритма поиска минимального и максимального значения по абсолютной величине (функция findAbs)***

***Схема алгоритма поиска минимального и максимального значения по абсолютной величине (функция isBelong)***

******

***Текст программы***

*/\**

*В заданной квадратной матрице размера (2n+1)×(2n+1) найти элементы с наибольшим по абсолютной величине*

*и с наименьшим по абсолютной величине значениями. Если хотя бы один из них находится в области правее*

*главной и побочной диагоналей найти сумму значений элементов, лежащих ниже главной диагонали.*

*В противном случае поменять найденные значения местами.*

*\*/*

#include<iostream>

#include<clocale>

#include <cmath>

using namespace std;

const int RAZ = 10;            // максимальный размер квадратной матрицы

typedef int telem;            // определение типа значений элементов массива

typedef telem tmatr[RAZ][RAZ];    // определение типа массива

bool input\_matr(tmatr a, int&);

void output\_matr(tmatr a, int);

void findAbs(tmatr a, int n, telem& maxDigit, telem& minDigit, int& minPI, int& minPJ, int& maxPI, int& maxPJ);

void swapValues(tmatr a, int minDigit, int maxDigit, int& minPI, int& minPJ, int& maxPI, int& maxPJ);

void sumMatr(tmatr a, int n, telem& amount);

bool isBelong(int minPI, int minPJ, int maxPI, int maxPJ, int n);

int main() {

  tmatr a;                           // двумерный массив

  int n;                            // половина размера квадратной матрицы

  int minPI = 0;

  int minPJ = 0;

  int maxPI = 0;

  int maxPJ = 0;

  telem amount = 0;                // сумма элементов ниже главной диагонали

  telem maxDigit;       // максимальное число по модулю

  telem minDigit;      // минимальное число по модулю

  setlocale(LC\_ALL, "Russian");

  if (!input\_matr(a, n)) {

    cout << "Введено неверное значение n < 5";

    return 1;

  }

  cout << "Исходная матрица:" << endl;

  output\_matr(a, n);

  findAbs(a, n, maxDigit, minDigit, minPI, minPJ, maxPI, maxPJ);

  if (isBelong(minPI, minPI, maxPI, maxPJ, n)) {

    sumMatr(a, n, amount);

    cout << "Сумма элементов, находящихся ниже главной диагонали: " << amount;

  } else {

    swapValues(a, minDigit, maxDigit, minPI, minPJ, maxPI, maxPJ);

    cout << "Матрица после преобразования:" << endl;

    output\_matr(a, n);

  }

  return 0;

}

//Ввод исходных данных

bool input\_matr(tmatr a, int& n) {

  cout << "Введите n  < 5, (матрица размера (2\*n+1)x(2\*n+1)):" << endl;

  cin >> n;

  if (n >= 5) {

    return false;

  }

  cout << "Введите построчно через пробел элементы" << endl;

  cout << "Матрицы размера " << 2 \* n + 1 << "x" << 2 \* n + 1 << endl;

  cout << "После ввода строки нажимайте <Enter>" << endl;

  for (int i = 0; i < 2 \* n + 1; i++)

    for (int j = 0; j < 2 \* n + 1; j++)

      cin >> a[i][j];

  return true;

}

//Вывод матрицы

void output\_matr(tmatr a, int n) {

  for (int i = 0; i < 2 \* n + 1; i++) {

    for (int j = 0; j < 2 \* n + 1; j++) {

      cout.width(3);

      cout << a[i][j];

    }

    cout << endl;

  }

}

// Поиск сумм значений элементов ниже главной диагонали матрицы

void sumMatr(tmatr a, int n, telem& amount) {

  for (int i = 1; i < 2 \* n + 1; i++) {

    for (int j = 0; i > j; j++) {

      amount += a[i][j];

    }

  }

}

// Перестановка минимального и максимального значений

void swapValues(tmatr a, int minDigit, int maxDigit, int& minPI, int& minPJ, int& maxPI, int& maxPJ) {

  a[minPI][minPJ] = maxDigit;

  a[maxPI][maxPJ] = minDigit;

}

// Поиск минимального и максимального числа по модулю и определение их принадлежности области

void findAbs(tmatr a, int n, telem& maxDigit, telem& minDigit,int& minPI, int& minPJ, int& maxPI, int& maxPJ) {

  minDigit = a[0][0];

  minPI = 0;

  minPJ = 0;

  maxDigit = a[0][0];

  maxPI = 0;

  maxPJ = 0;

  for (int i = 0; i < 2 \* n + 1; i++) {

    for (int j = 0; j < 2 \* n + 1; j++) {

      if (abs(maxDigit) < abs(a[i][j])) {

        maxDigit = a[i][j];

        maxPI = i;

        maxPJ = j;

      }

      if (abs(minDigit) > abs(a[i][j])) {

        minDigit = a[i][j];

        minPI = i;

        minPJ = j;

      }

    }

  }

}

bool isBelong(int minPI, int minPJ, int maxPI, int maxPJ, int n) {

  bool isBelong = false;

  if (maxPI < maxPJ && maxPI > 2 \* n - maxPJ) {

    isBelong = true;

  }

  if (minPI < minPJ && minPI > 2 \* n - minPJ) {

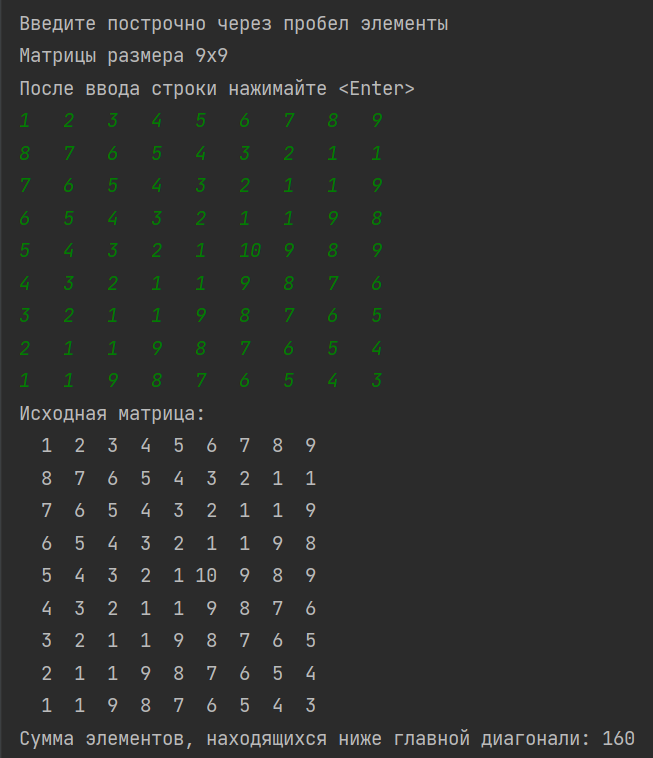
    isBelong = true;

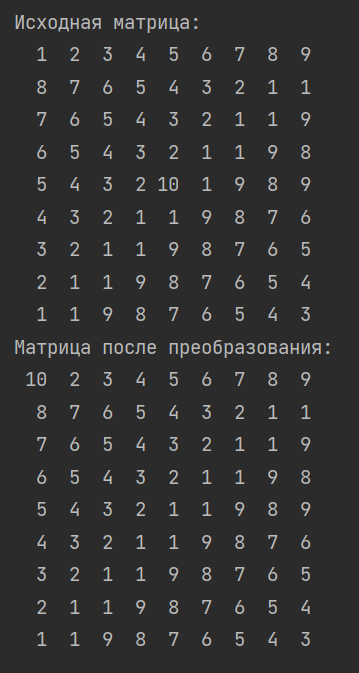
  }

  return isBelong;

}

***Скриншоты результатов выполнения программы***

******



***Вывод:***

*Я изучил структурную организацию массивов и способов доступа к их элементам; усовершенствовал навыки процедурного программирования на языке C/C++ при решении задач обработки двумерных статических массивов (матриц).*