МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ПЕРМСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

ЛАБОРАТОРНАЯ РАБОТА №5  
на тему «Функции и массивы»

Выполнила: студентка группы РИС-23-2Б Жилина Анастасия Александровна

Проверила: доцент кафедры ИТАС О.А.Полякова

2024**Разработка алгоритма из варианта 18.**

1. **Постановка задачи**

Задан двумерный массив m x m. Последовательно рассматриваются квадратные подмассивы, правый верхний элемент которых лежит на побочной диагонали. В каждом таком подмассиве находится максимальный элемент. Путем перестановок строк и столбцов (целиком) элемент надо переместить в правый верхний угол подмассива. Проверить получилась ли на побочной диагонали убывающая последовательность элементов.

1. **Анализ задачи:**

Задачу можно разбить на 4 этапа:

*1 этап:* ввод самого массива;

2 и 3 этапы выполняются в связке (в цикле):

*2 этап:* нахождение в каждом квадратном подмассиве, правый верхний элемент которого лежит на побочной диагонали, максимального элемента;

*3 этап:* перестановка найденного максимального элемента в правый верхний угол;

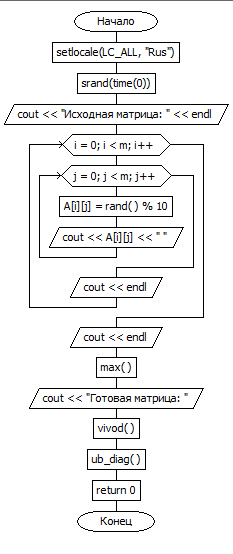
*4 этап:* после преобразований матрицы проверить побочную диагональ на убывающую последовательность.

Во втором этапе работаем по такой схеме: рассматриваем все элементы матрицы и ищем максимальный. После нахождения максимального, необходимо так же запомнить его индекс, а затем наступает третий этап: перестановка строк и столбцов так, чтобы максимальный элемент оказался в правом верхнем углу. Для этого мы рассматриваем варианты положения максимального элемента – он может стоять как на одном ряду или на одной строке с правым верхним элементом, так и никак не зависеть от него. В первых двух случаях необходимо поменять ряды местами или строки, в последнем – использовать оба метода – сначала поменять местами ряд, в котором находится максимальный элемент с рядом, в котором находится верхний правый элемент, а затем поменять строки. После всех манипуляций вновь наступает второй этап. Так происходит, пока не рассмотрится самый наименьший квадратный подмассив 2 x 2.

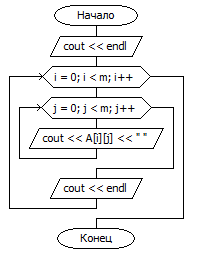
На 4 этапе необходимо лишь сравнить между собой элементы побочной диагонали, начиная с правого верхнего А[0][m-1], заканчивая левым нижним А[m-1][0]. Будет использоваться дополнительная переменная максимум для сравнения и флажок, который показывает, нарушилось ли условие убывающей последовательности или нет.

1. **Блок-схема**

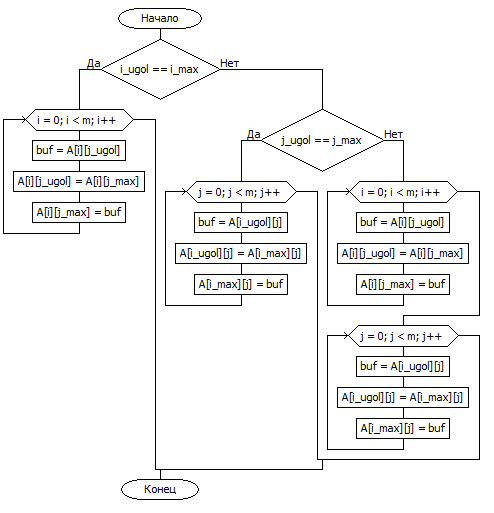
int main():

****

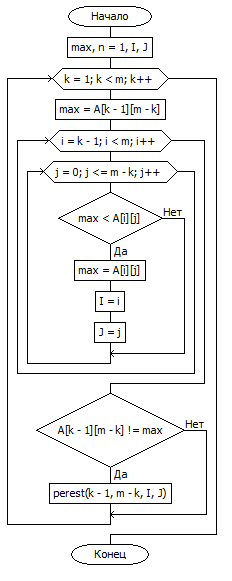
void vivod():



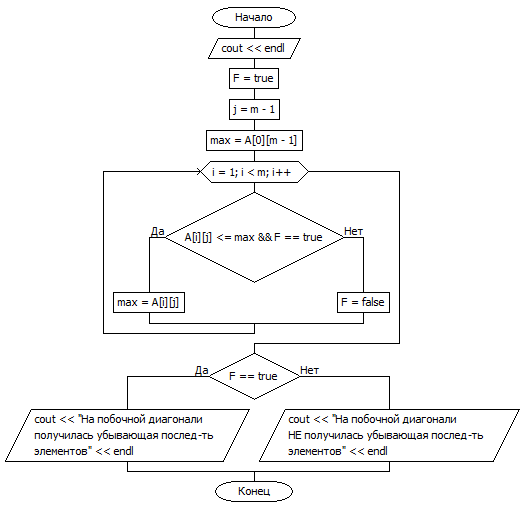
void perest(int i\_ugol, int j\_ugol, int i\_max, int j\_max)

****

void max():

****

void ub\_diag():



1. **Листинг программы**

#include <iostream>

using namespace std;

const int m = 4;

int A[m][m];

void vivod()

{

cout << endl;

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

{

for (int j = 0; j < m; j++)

{

cout << A[i][j] << " ";

}

cout << endl;

}

}

void perest(int i\_ugol, int j\_ugol, int i\_max, int j\_max)

{

int buf;

if (i\_ugol == i\_max) //Если индекс строки текущего макс элемента совпадает со строкой углового элемента

{

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

{

buf = A[i][j\_ugol];

A[i][j\_ugol] = A[i][j\_max];

A[i][j\_max] = buf;

}

}

else if (j\_ugol == j\_max) //Если индекс столбца текущего макс элемента совпадает со столбцом углового элемента

{

for (int j = 0; j < m; j++)

{

buf = A[i\_ugol][j];

A[i\_ugol][j] = A[i\_max][j];

A[i\_max][j] = buf;

}

}

else //Если не совпадает ни индекс столбца, ни индекс строки текущего макс элемента совпадает индексами углового элемента

{

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

{

buf = A[i][j\_ugol];

A[i][j\_ugol] = A[i][j\_max];

A[i][j\_max] = buf;

}

for (int j = 0; j < m; j++)

{

buf = A[i\_ugol][j];

A[i\_ugol][j] = A[i\_max][j];

A[i\_max][j] = buf;

}

}

}

void max()

{

int max, n=1, I, J;

int i, j;

for (int k = 1; k < m; k++)

{

max = A[k - 1][m - k];//максимальное возьмем как верхний правый угол

for (int i = k - 1; i < m; i++)

{

for (int j = 0; j <= m - k; j++)

{

if (max < A[i][j])

{

max = A[i][j];

I = i;

J = j;

}

}

}

if (A[k - 1][m - k] != max)

{

perest(k - 1, m - k, I, J);

}

}

}

void ub\_diag()

{

cout << endl;

bool F = true;

int j = m - 1;

int max = A[0][m - 1];

for (int i = 1; i < m; i++)

{

--j;

if (A[i][j] <= max && F == true)

{

max = A[i][j];

}

else F = false;

}

if (F == true)

{

cout << "На побочной диагонали получилась убывающая послед-ть элементов" << endl;

}

else cout << "На побочной диагонали НЕ получилась убывающая послед-ть элементов" << endl;

}

int main()

{

setlocale(LC\_ALL, "Rus");

srand(time(0));

cout << "Исходная матрица: " << endl;

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

{

for (int j = 0; j < m; j++)

{

A[i][j] = rand() % 10;

cout << A[i][j] << " ";

}

cout << endl;

}

cout << endl;

max();

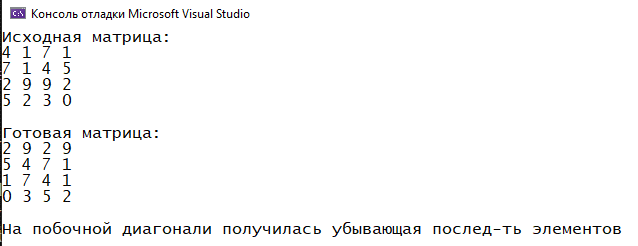
cout << "Готовая матрица: ";

vivod();

ub\_diag();

}

1. **Результаты работы**



Разместим готовые исполнимые файлы в созданный репозиторий на GitHub

