**Лабораторная работа №4**

**Динамические массивы**

**Вариант 9**

***Условие***

При поиске элементов матрицы, строк или столбцов с экстремальными свойствами следует предполагать их единственность.

Переставить в каждом столбце матрицы все отрицательные эле­менты в конец столбца. Вывести часть полученной матрицы, со­стоящую из первых строк, не имеющих отрицательных элементов.

***Состав данных***

|  |  |  |  |
| --- | --- | --- | --- |
| **Имя** | **Смысл** | **Тип** | **Структура** |
| Исходные данные | | | |
| **x** | заданная матрица | вещественный | двумерный динамический массив |
| **n** | количество строк матрицы x | целый | простая переменная |
| **m** | количество столбцов матрицы x | целый | простая переменная |
| **create\_flag** | выбор ввода матрицы x | символьный | простая переменная |
| Промежуточные данные | | | |
| **i** | счетчик цикла | целый | простая переменная |
| **j** | счетчик цикла | целый | простая переменная |
| **k** | счетчик цикла | целый | простая переменная |
| **flag** | выполнение 2 условия в столбцах | целый | простая переменная |

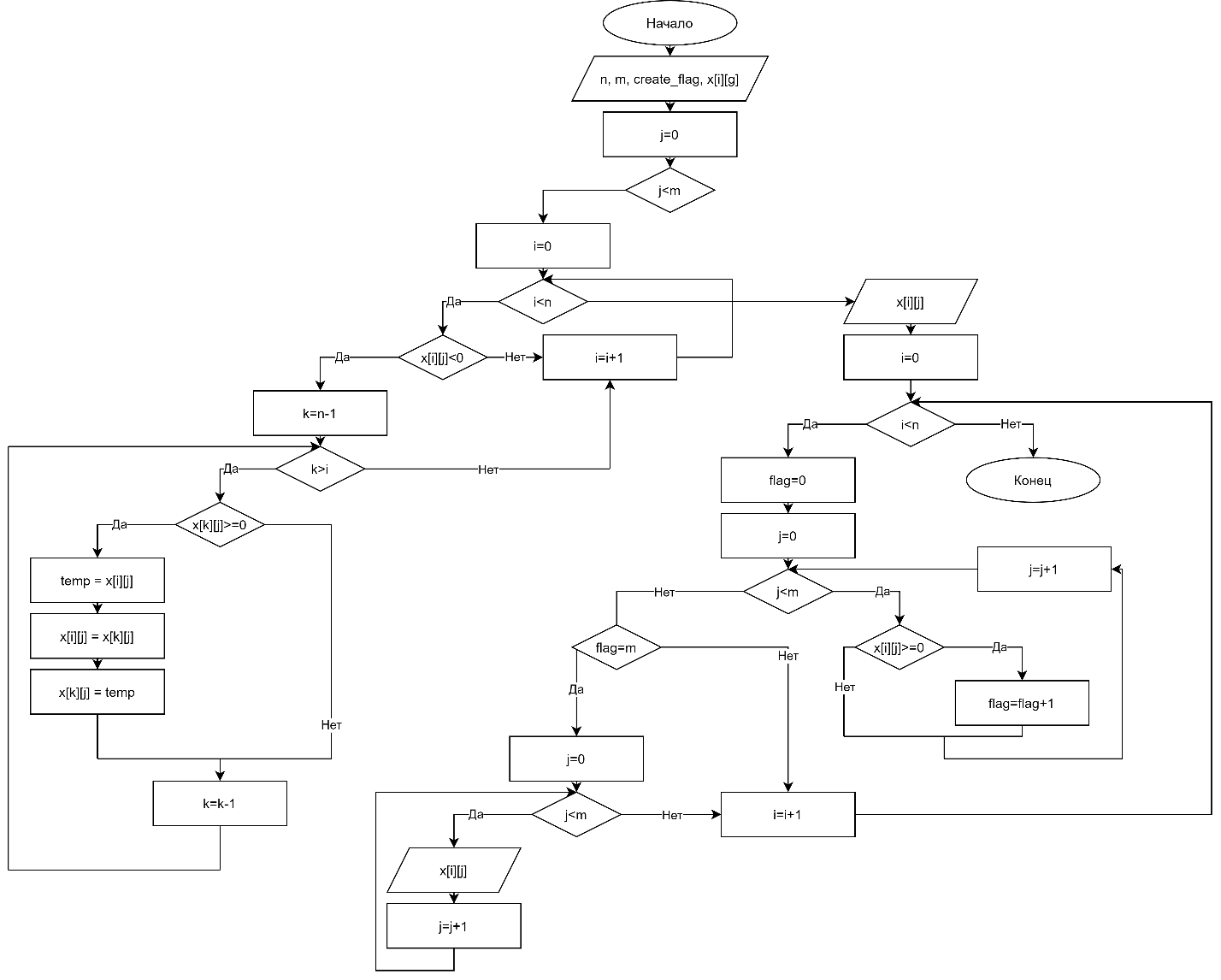
***Форма ввода***

|  |
| --- |
| Введите число строк матрицы:  <n>  Введите число столбцов матрицы:  <m>  Создать матрицу автоматически? (Y-да/N-нет):  <create\_flag>  Введите элементы массива А:  Строка №<i+1>  <x[i][j]> |

***Форма вывода***

|  |
| --- |
| Полученный массив:  <x[0][0]> … <x[0][m]>  <x[1][0]> … <x[1][m]>  …  <x[n][0]> … <x[n][m]>  Вывод после сортировки:  <x[0][0]> … <x[0][m]>  <x[1][0]> … <x[1][m]>  …  <x[n][0]> … <x[n][m]>  Вывод массива после 2 условия:  <x[i][1]> … <x[i][m]>  …  <x[k][1]> … <x[k][m]> |

***Блок-схема программы***



***Код программы***

#include "stdafx.h"

#include <iostream>

#include <ctime>

/\*

Переставить в каждом столбце матрицы все отрицательные элементы в конец столбца.

Вывести часть полученной матрицы, состоящую из первых строк, не имеющих отрицательных элементов.

\*/

using namespace std;

int main()

{

setlocale(LC\_ALL, "Russian");

srand(static\_cast<unsigned int>(time(0))); // Привязка к времени системных часов для получения "рандомных" чисел

int n, m;

cout << "Введите число строк матрицы:" << endl;

cin >> n;

cout << "Введите число столбцов матрицы:" << endl;

cin >> m;

try

{

if ((n <= 1) || (m <= 1))

throw "Величина строк и столбцов должна быть больше единицы";

double\*\* x;

x = new double\* [n];

for (int i = 0; i < n; i += 1) x[i] = new double[m];

cout << "Создать матрицу автоматически? (Y-да/N-нет):" << endl;

char create\_flag;

cin >> create\_flag;

try

{

if ((create\_flag != 'Y') && (create\_flag != 'N'))

throw "Необходимо выбрать Y-да/N-нет!";

// Автоматическое создание матрицы

if (create\_flag == 'Y') {

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

for (int j = 0; j < m; j += 1) {

x[i][j] = (500 - rand() % 1000) / 10;

}

}

}

// Ручное создание матрицы

if (create\_flag == 'N') {

cout << "Введите элементы массива А:" << endl;

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

cout << "Строка №" << i + 1 << ":" << endl;

for (int j = 0; j < m; j += 1) {

cin >> x[i][j];

}

}

}

// Вывод массива

cout << "Полученный массив: " << endl;

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

for (int j = 0; j < m; j += 1) {

cout << x[i][j] << "\t";

}

cout << endl;

}

// Сортировка по условию

int temp;

for (int j = 0; j < m; j += 1) {

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

if (x[i][j] < 0) {

for (int k = n - 1; k > i; k -= 1) {

if (x[k][j] >= 0) {

temp = x[i][j];

x[i][j] = x[k][j];

x[k][j] = temp;

}

}

}

}

}

cout << "Вывод после сортировки:" << endl;

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

for (int j = 0; j < m; j += 1) {

cout << x[i][j] << "\t";

}

cout << endl;

}

cout << "Вывод массива после 2 условия:" << endl;

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

int flag = 0;

for (int j = 0; j < m; j += 1) if (x[i][j] >= 0) flag++;

if (flag == m) {

for (int j = 0; j < m; j += 1) {

cout << x[i][j] << "\t";

}

cout << endl;

}

}

// Удаление массива из памяти

for (int i = 0; i < n; i += 1) delete[] x[i];

delete[] x;

}

catch (const char\* exception)

{

cerr << "Ошибка: " << exception << '\n';

}

}

catch (const char\* exception)

{

cerr << "Ошибка: " << exception << '\n';

}

system("pause");

return 0;

}

