Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования «Тульский государственный университет»

КАФЕДРА ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ

ДИНАМИЧЕСКИЕ ПЕРЕМЕННЫЕ В ЯЗЫКЕ ПРОГРАММИРОВАНИЯ С

отчет о
лабораторной работе № 12

по дисциплине
ПРОГРАММИРОВАНИЕ

ВАРИАНТ 10

Выполнил: ст. гр. 220451 Курбаков М.Ю.
Проверил: асс. каф. ИБ Курбаков М.Ю.

Тула, 2022 г.

ЦЕЛЬ И ЗАДАЧА РАБОТЫ

Цель: научиться использовать динамические переменные.

Задача: в данной работе требуется написать программу с использованием динамических переменных, в которой данные сначала из файла считываются в память, потом обрабатываются и записываются в файл.

ЗАДАНИЕ НА РАБОТУ

Переставляя строки и столбцы матрицы, добиться, чтобы в левом верхнем углу оказался наибольший элемент матрицы (один из них).

СХЕМА АЛГОРИТМА

Схема алгоритма перемещения наибольшего элемента матрицы в левый верхний угол, путем перестановки сток и столбцов, представлена на рисунке 1.

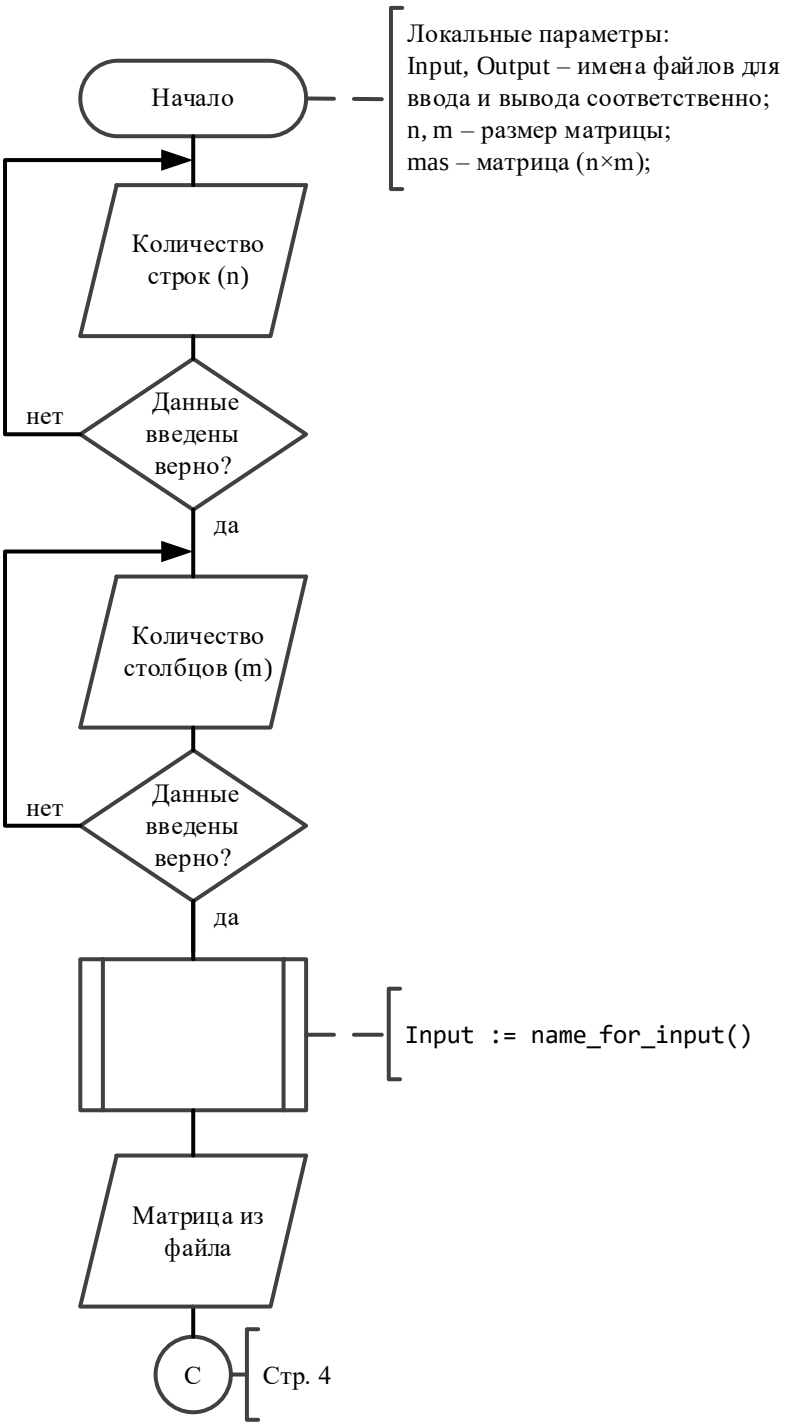

Рисунок 1 – Схема алгоритма перемещения наибольшего элемента матрицы в левый верхний угол

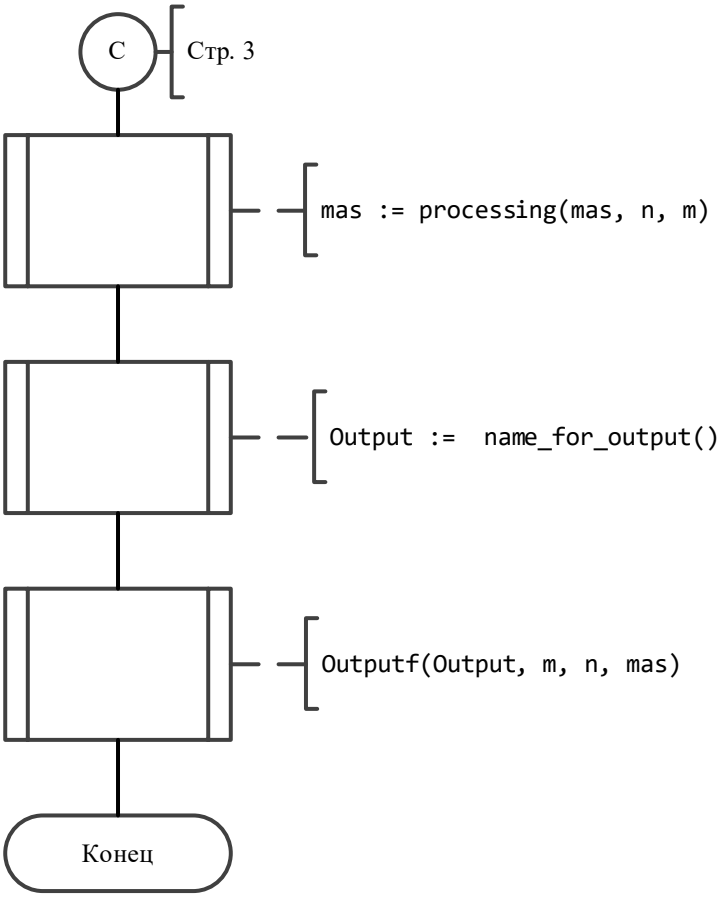

Рисунок 1 – Схема алгоритма перемещения наибольшего элемента матрицы в левый верхний угол (продолжение)

Схема алгоритма ввода названия файла для записи результата представлена на рисунке 2.

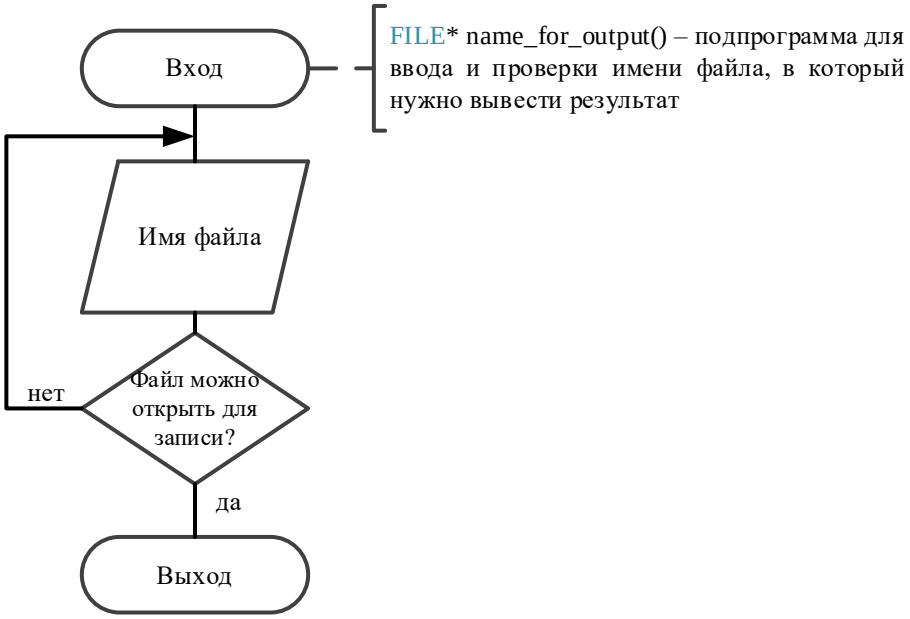

Рисунок 2 – Схема алгоритма ввода названия файла

Схема алгоритма вывода результата в файл и на экран представлена на рисунке 3.

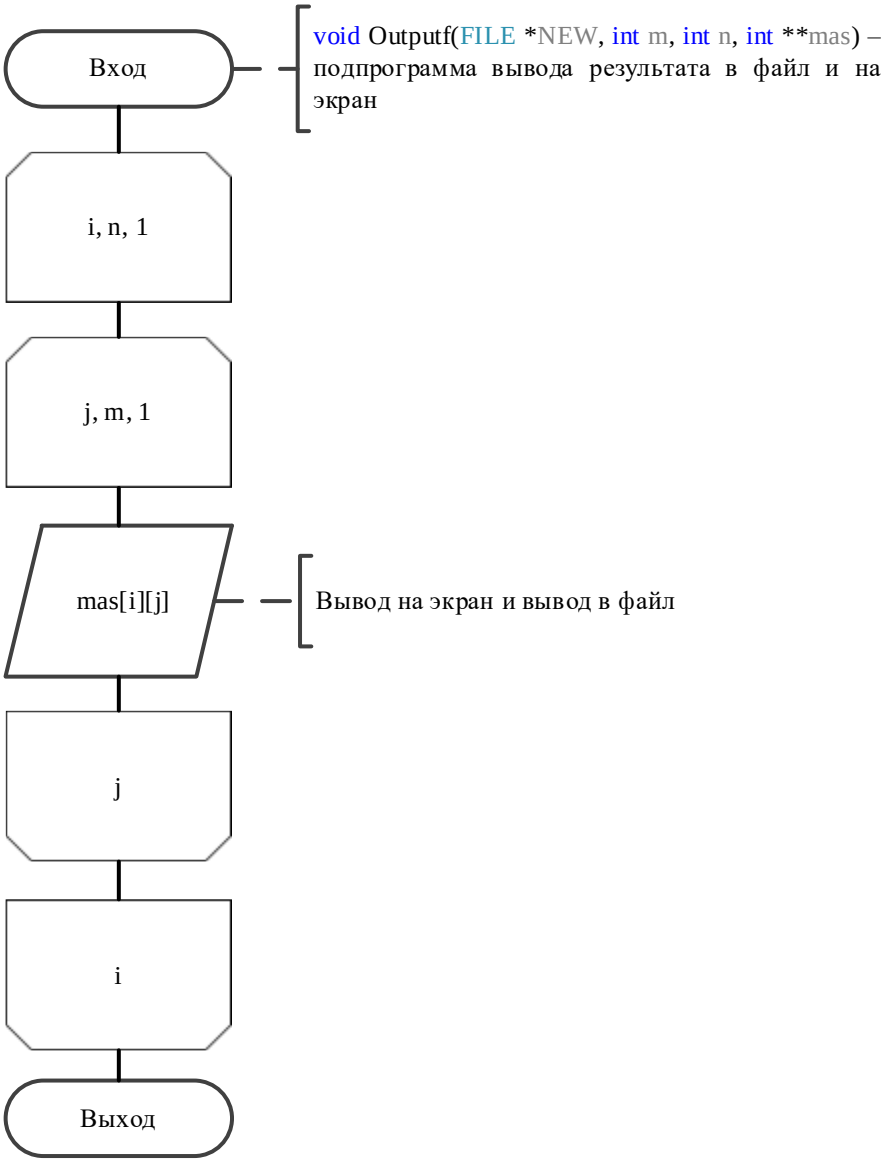

Рисунок 3 – Схема алгоритма вывода результата в файл и на экран

ТЕКСТ ПРОГРАММЫ

Текст программы, которая перемещает наибольший элемента матрицы в левый верхний угол, путем перестановки сток и столбцов, на языке программирования С представлен в листинге 1.

Листинг 1. Текст программы

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <math.h>
#include <conio.h>
#include <malloc.h>
```

Листинг 1. Текст программы (продолжение)

```
FILE* name_for_output();
float** getmem(float **arr, int n, int m);
float** freemem(float **arr, int n);
float** processing(float** mas, int n, int m);
void Outputf(FILE *NEW, int m, int n, float **mas);
void main()
{
  FILE *Output; float **mas = NULL; int n = 0, m = 0; int s = 0;
  setlocale(LC_ALL, "Russian");

  do
  {
     printf("Введите количество строк в матрице - n, где n > 2: ");
     scanf("%i", &n);
  } while (n < 2);

  do
  {
     printf("Введите количество столбцов в матрице- m, где m > 2: ");
     scanf("%i", &m);
  } while (m < 2);

  mas = getmem(mas, n, m);

  if (mas != NULL)
  {
     printf("Хотите заполнить матрицу вручную, введите 1. "
        " Иначе матрица заполнится случайными числами. >> ");
     scanf("%i", &s);
     if (s == 1)
     {
        for (int i = 0; i < n; i++)
           for (int j = 0; j < m; j++)
           {
              printf("Элемент матрицы[%i][%i]: ", i + 1, j + 1);
              scanf("%f", &mas[i][j]);
           }
     }
     else
     {
        int r, ver = 0;
        printf("\nВведите диапазон генератора случайных чисел: ");
        ver = scanf("%d", &r);
        for (int i = 0; i < n; i++)
           for (int j = 0; j <= m - 1; j++)
              mas[i][j] = rand() %r;
     }

     printf("Ваша матрица:\n");
     for (int i = 0; i < n; i++)
     {
        for (int j = 0; j < m; j++)
        {
           printf("%8.2f", mas[i][j]);
        }
        printf("\n");
     }
```

Листинг 1. Текст программы (продолжение)

```
        }
        printf("\n");
        fprintf(NEW, "\n");
    }
}
```

ИНСТРУКЦИЯ ПОЛЬЗОВАТЕЛЯ

Данная программа предназначена для перемещения наибольшего элемента матрицы в левый верхний угол, путем перестановки сток и столбцов. При запуске программы появляется запрос на ввод размеров матрицы (чисто строк и столбцов). Далее необходимо указать вариант заполнения матрицы:

- Вручную. В таком случае необходимо по очереди указать значения элементов матрицы в соответствии с подсказками на экране о номере вводимого элемента;
- Случающими числами. В таком случае необходимо максимальную указать границу генерации случайных чисел – r. После матрица будет заполнена целыми случайными числами в диапазоне $[0, r]$.

После корректного ввода необходимых данных программа выведет на экран сформированию матрицу значений. Далее необходимо указать имя файла для вывода результата перестановки. Если указанный файл можно использовать для записи, то программа выведет на экране результат перестановки и запишет его в указанный файл.

В случае, если максимальный элемент (один из максимальных) уже находится в левом верхнем углу матрицы, то никаких перестановок не требуется, и программа сообщит об этом, записав в файл матрицу без изменений.

ИНСТРУКЦИЯ ПРОГРАММИСТА

Данная программа предназначена для перемещения наибольшего элемента матрицы в левый верхний угол, путем перестановки сток и столбцов.

Структуры данных, используемых в программе, приведены в таблице 1.

Таблица 1 – Структуры данных в основной программе

Имя	Тип	Предназначение
Input, Output	FILE	Имена файлов с входными и выходными данными соответственно
n, m	int	Количество строк и столбцов в матрице соответственно
mas	float	Обрабатываемая матрица

Программа разбита на следующие подпрограмм:

1) float** getmem(float** arr, int n, int m) – подпрограмма для выделения памяти под массив. Структуры данных, используемых в подпрограмме приведена в таблице 2.

Таблица 2 – Структуры данных, используемые в подпрограмме getmem

Имя	Тип	Предназначение
		формальные параметры
n, m	int	Количество строк и столбцов в матрице соответственно
arr	float**	Обрабатываемая матрица

ДЕМОНСТРАЦИОННЫЙ ПРИМЕР

Результат работы программы для перемещения наибольшего элемента матрицы в левый верхний угол, путем перестановки сток и столбцов, изображен на рисунках 4,5.

```
C:\WINDOWS\system32\cmd.exe
Введите количество строк в матрице - n, где n > 2: 4
Введите количество столбцов в матрице- m, где m > 2: 4
Хотите заполнить матрицу вручную, введите 1. Иначе матрица заполнится случайными числами. >> 1
Элемент матрицы[1][1]: -3,323
Элемент матрицы[1][2]: 8,72
Элемент матрицы[1][3]: 6,622
Элемент матрицы[1][4]: 9,22
Элемент матрицы[2][1]: 6,9
Элемент матрицы[2][2]: -10,8
Элемент матрицы[2][3]: 43,9
Элемент матрицы[2][4]: 9,99
Элемент матрицы[3][1]: -87,98
Элемент матрицы[3][2]: 102
Элемент матрицы[3][3]: 7,62
Элемент матрицы[3][4]: 5,45
Элемент матрицы[4][1]: 4,32
Элемент матрицы[4][2]: 9,22
Элемент матрицы[4][3]: 74,34
Элемент матрицы[4][4]: 78
Ваша матрица:
   -3,32    8,72    6,62    9,22
    6,90  -10,80   43,90    9,99
  -87,98  102,00    7,62    5,45
    4,32    9,22   74,34   78,00

Введите назание файла, В который необходимо записать результат: output.txt
  102,00  -87,98    7,62    5,45
  -10,80    6,90   43,90    9,99
    8,72   -3,32    6,62    9,22
    9,22    4,32   74,34   78,00
Программа завершена, результат записан в файл.
Для продолжения нажмите любую клавишу . . .
```

Рисунок 4 – Пример работы программы

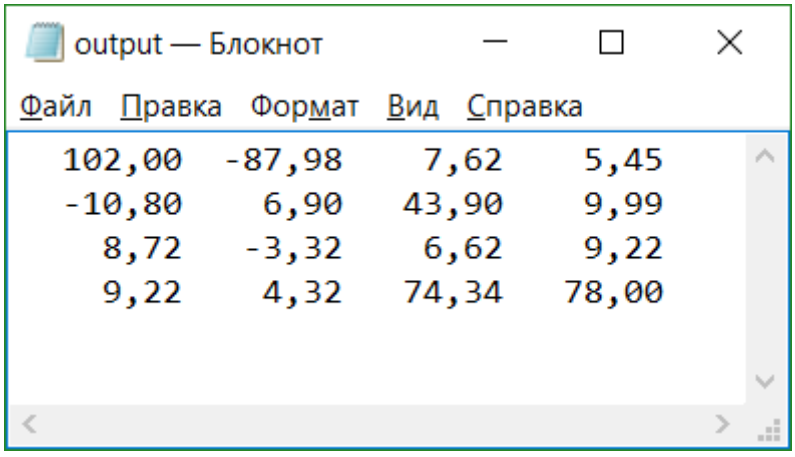

Рисунок 5 – Итоговый файл

Проверим результат работы программы аналитически. Исходя из введенных данных, представленных на рисунке 4, можно сделать вывод, что максимальный элемент равен 102, который находится в матрице в 3 строке, 2 столбце. Следовательно, по задания мы должны поменять первую строку с третий и после второй столбец с первым:

$$\begin{pmatrix} -87{,}98 & 102{,}00 & 7{,}62 & 5{,}45 \\ 6{,}90 & -10{,}80 & 43{,}90 & 9{,}99 \\ -3{,}32 & 8{,}72 & 6{,}62 & 9{,}22 \\ 4{,}32 & 9{,}22 & 74{,}34 & 78{,}00 \end{pmatrix} => \begin{pmatrix} 102{,}00 & -87{,}98 & 7{,}62 & 5{,}45 \\ -10{,}80 & 6{,}90 & 43{,}90 & 9{,}99 \\ 8{,}72 & -3{,}32 & 6{,}62 & 9{,}22 \\ 9{,}22 & 4{,}32 & 74{,}34 & 78{,}00 \end{pmatrix}$$

Если сравнить полученную матрицу и матрицу из файла, можно убедиться, что в данном случае программа отработала корректно.

ВЫВОДЫ

Создание массива с фиксированным размером подразумевает, что под него выделяется определенная память, соответствующая заданному размеру. Однако это не всегда удобно, а в некоторых случаях бывает необходимо, чтобы количество элементов и соответственно размер выделяемой памяти для массива определялись динамически в зависимости от некоторых условий. В этом случае для создания массива мы можем использовать динамическое выделение памяти.