

**Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
"Уфимский государственный авиационный технический университет"**

**Кафедра** Высокопроизводительных вычислительных технологий и систем

**Дисциплина:** Программирование

**Отчет по лабораторной работе № 3**

**Тема:** Указатели и структурные типы данных

Группа ПМ-153	Фамилия И.О.	Подпись	Дата	Оценка
Студент	Шамаев И.Р.			
Принял	Гайнетдинова А.А.			

**Уфа 2019**

**Цель:** изучить принципы работы с указателями и структурными типами данных (массивы и структуры) в СИ. Приобрести практические навыки в реализации алгоритмов сортировки одномерных массивов.

### **Теоретический материал**

Были использованы указатели и одномерные массивы. Использовалась структура сортировки одномерных данных.

### **Ответы на контрольные вопросы**

*Дайте определение указателя.*

Указатель это переменная, хранящая адрес некой области памяти. Часто говорят, что указатель указывает на область памяти, или указатель ссылается на область памяти.

*Каким образом можно получить адрес переменной в явном виде?*

Чтобы получить адрес переменной, нужно перед ее именем написать амперсанд. Сначала с помощью оператора &, примененного к переменной x, будет получен адрес x. Затем адрес x будет сохранен в указателе

*Дайте определение массива.*

Массив - это пронумерованная совокупность однотипных данных. Каждое значение в массиве называется элементом массива, а номер элемента массива - индексом.

*Какой результат будет получен при разыменовании имени массива?*

Его начальный элемент (с индексом 0).

*Что происходит при добавлении целочисленного значения n к указателю, адресуящему некоторый элемент в массиве?*

Добавление целочисленного значения n к указателю, адресуящему некоторый элемент массива, приводит к тому, что указатель получает значение адреса того элемента, который отстоит от текущего на n элементов

*В чем отличие статического и динамического выделения памяти для массива?*

Под статический массив память выделяется в стеке, под динамический – в куче.

Размер статического массива должен быть известен до компиляции. Размер динамического массива можно определить во время выполнения программы.

Размер статического массива нельзя изменить во время выполнения программы. Динамический массив может изменять свой размер во время выполнения программы, как в сторону увеличения, так и в сторону уменьшения.

Память, выделенная под статический массив освобождается автоматически после выхода программы из блока, в котором он определён. Память, выделенную под динамический массив следует освобождать принудительно, чтобы избежать её утечек.

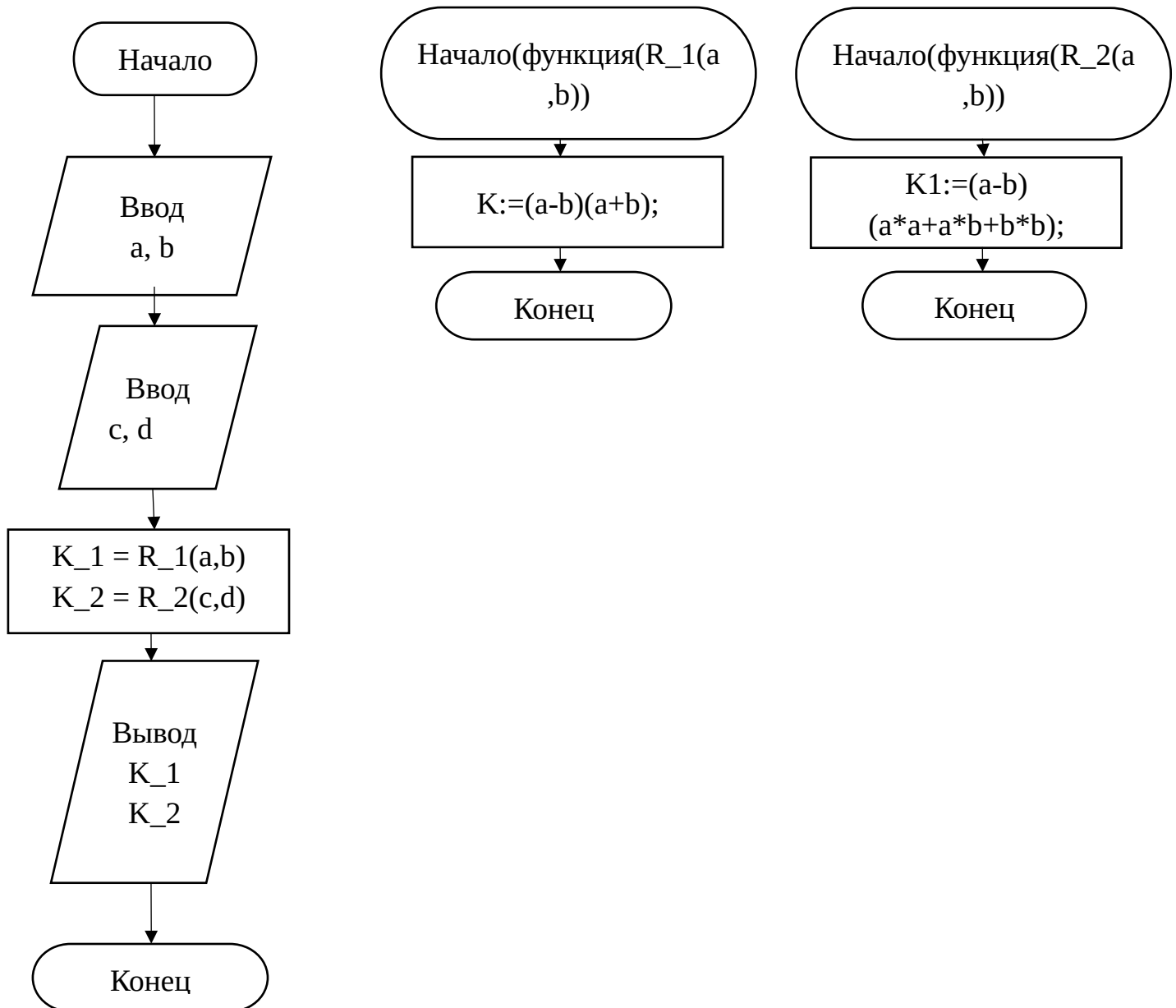
*В чем отличие функций динамического выделения памяти calloc() и malloc()?*

malloc просто выделяет память, оставляя ее содержимое неопределенным, а calloc ее гарантированно обнуляет.

## Индивидуальное задание №1

Задание: Написать программу, вычисляющую разность квадратов и разность кубов двух чисел. Расчет должен производиться в функции, которая получает данные по указателю и возвращает результат по указателю

### Блок-схема



### Описание программы:

На вход подаются два числа для разности квадратов и для разности кубов.

Функция R\_1 вычисляет разность квадратов

Функция R\_2 вычисляет разность кубов

### Исходный код программы

```
#include <math.h>
```

```
#include <stdio.h>
```

```
using namespace std;
```

```
int* R_1(int* a, int* b)//функция, для расчета разности квадратов
```

```
{
```

```
static int k;
```

```
k = (*a - *b)*(*a + *b);//формула расчета разности квадратов
```

```
return &k;
```

```
}
```

```
int* R_2(int* a, int* b)//функция, для расчета разности кубов
```

```
{
```

```
static int k1;
```

```
k1 = (*a - *b)*((*a) * (*a) + (*a) * (*b) + (*b) * (*b)); //формула расчета
```

разности кубов

```
return &k1;
```

```
}
```

```
int main()
```

```
{
```

```
int a, b, c, d;
```

```
printf("Vvedi chisla dlya raznosti kvadratov");
```

```
puts("\n");
```

```
scanf("%i %i", &a, &b);//вводим числа
```

```
printf("Vvedi chisla dlya raznosti kubov");
```

```
puts("\n");
```

```
scanf("%i %i", &c, &d); //вводим числа
```

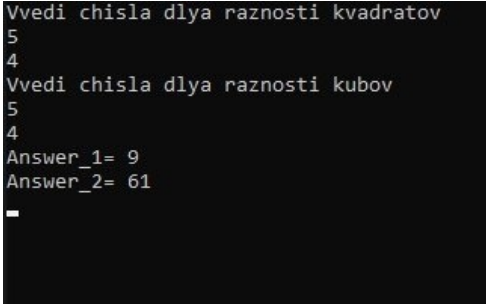
```
int* k_1 = R_1(&a, &b); //указываем на функцию
```

```
int* k_2 = R_2(&c, &d); //указываем на функцию
```

```
printf("Answer_1= %i", *k_1); //выводим результат разности кубов
puts("\n");
printf("Answer_2= %i", *k_2); //выводим результат разности кубов

return 0;
}
```

#### Пример выполнения программы

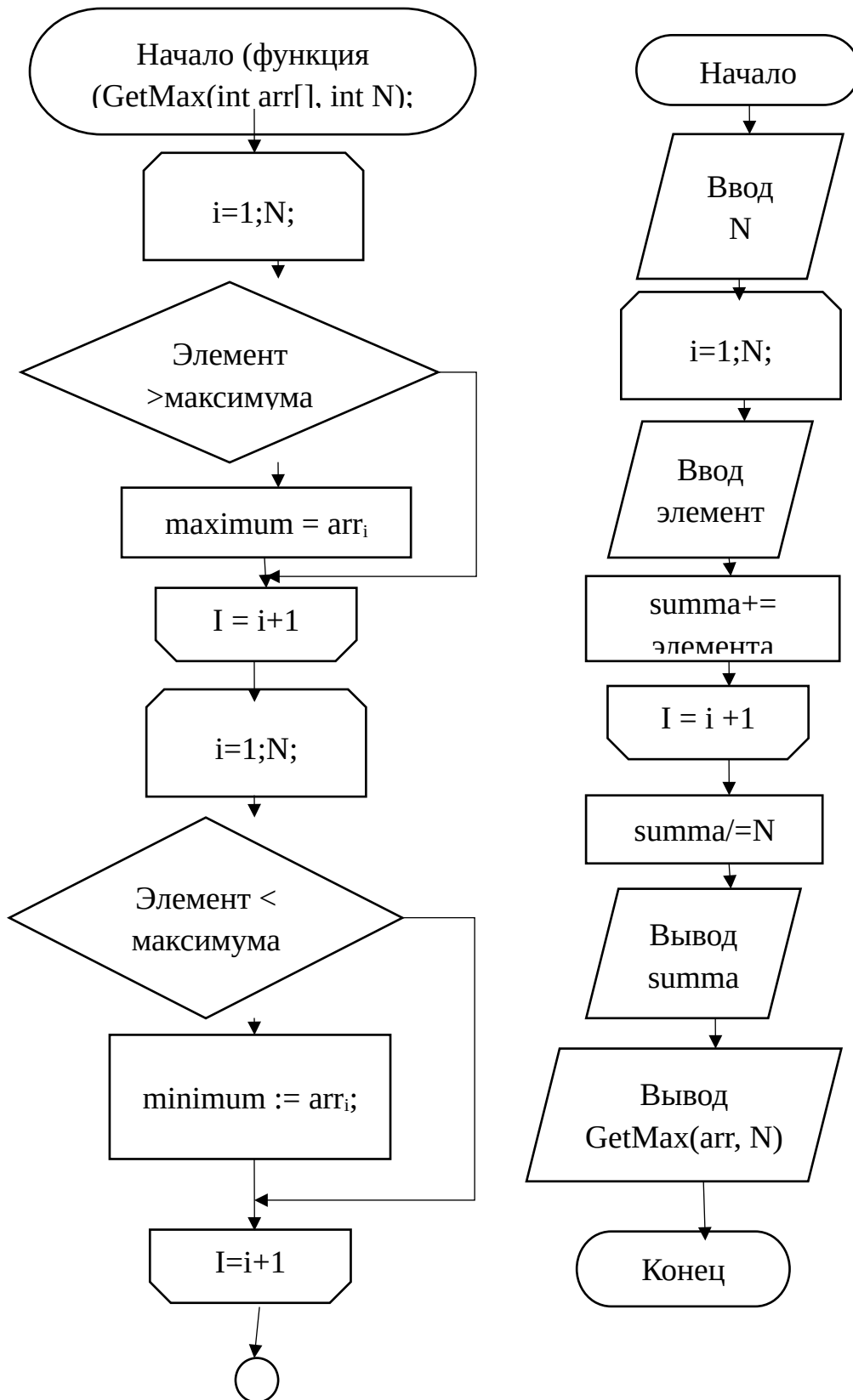


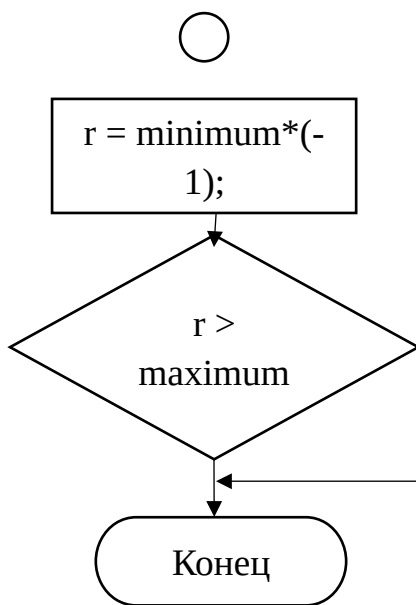
```
Vvedi chisla dlya raznosti kvadratov
5
4
Vvedi chisla dlya raznosti kubov
5
4
Answer_1= 9
Answer_2= 61
_
```

## Индивидуальное задание №2

Задание: Написать программу, определяющую модуль максимального отклонения элементов массива от среднего значения всех элементов одномерного массива. При вводе/выводе элементов использовать индексы, а обработку элементов массива осуществлять с помощью указателей. Расчет должен производиться в функции, в которую массив передается по указателю

Блок-схема





### Описание программы:

Вводим количество элементов массива и при помощи массива вводим значения в массив и находим среднее значение.

При помощи функции находим максимальное отклонение от среднего значения

### Исходный код программы

```

#include <math.h>
#include <stdio.h>
#include <malloc.h>
using namespace std;

```

```

int GetMax(int *arr, int N);
int r;

```

```

int main()
{
    int summa=0, N;
    int *arr;
    arr = (int*)malloc(N * sizeof(int));
    scanf("%i", &N); // вводим количество элементов массива

```

```

    printf("Enter numbers:");
    for (int i=0; i<N; i++){
        scanf(" %i", &arr[i]); //вводим элементы
        summa+=*(arr+i); // находим сумму всех элементов
    }

```



```
summa/=N; //находим среднее значение
```

```
printf("Average %i", summa); //выводим среднее значение
```

```
printf("\nMax element %i", GetMax(arr, N)); // выводим максимальное  
отклонение  
free(arr);  
return 0;  
}
```

```
int GetMax(int *arr, int N) // функция нахождения максимального отклонения  
{
```

```
int maximum = *arr;
```

```
for (int i=0; i<=N; i++) //цикл нахождения максимального элемента  
if (*(arr+i) > maximum) //находим максимальный элемент  
maximum = *(arr +i-1));
```

```
int minimum = arr[0];  
for (int i=0; i<=N; i++){ //цикл нахождения минимального элемента  
if (*(arr+i) < minimum) //находим минимальный элемент  
minimum = arr[i-1];  
}  
r = minimum;  
if( r > maximum){ //находим максимальное отклонение  
return minimum;  
}  
else  
return maximum;  
}
```

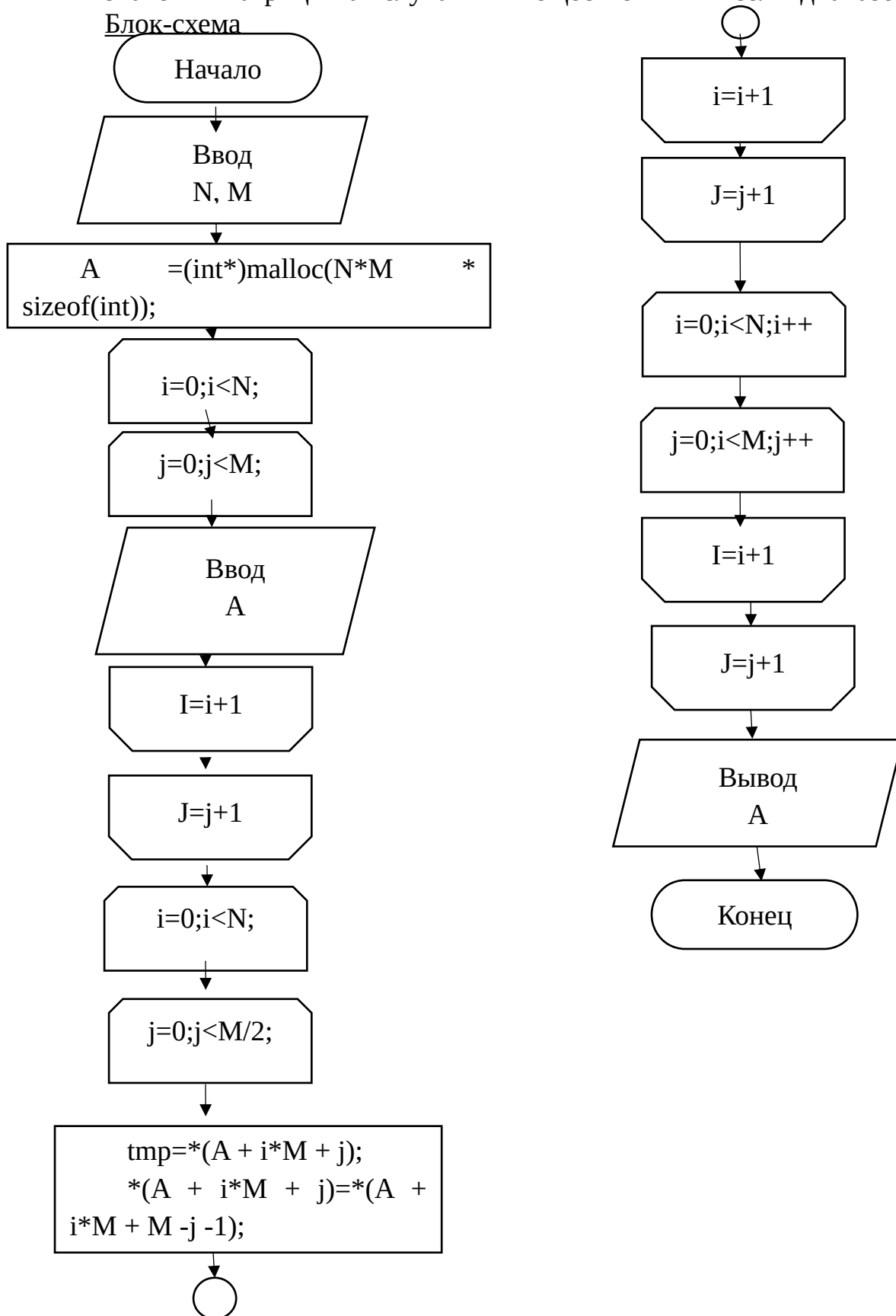
Пример выполнения программы

```
5
Enter numbers:1
2
3
4
10
Average 4
Max element 10
Process returned 0 (0x0)   execution time : 24.210 s
Press any key to continue.
```

### Индивидуальное задание №3

Задание: Написать программу, выполняющую отражение неквадратной матрицы относительно центральной вертикальной оси. Матрицы должны храниться в памяти в виде двумерного динамического массива, размерности исходной матрицы вводятся пользователем с клавиатуры. Предусмотреть генерацию значений матрицы как случайных вещественных чисел в диапазоне от -10 до 10.

Блок-схема



### Описание программы:

Сперва вводится размер матрицы. Вводятся значения в матрицу и выводится полученная матрица. Далее зеркально отражаем матрицу относительно вертикальной линии посредством попеременного обмена значений ячеек массива

### Исходный код программы

```
#include <stdio.h>
#include <stdlib.h>
#include <malloc.h>

int main()
{
    double *A;
    int N,M,i,j;
    scanf("%d",&N); //вводим длину строки матрицы
    scanf("%d",&M); //вводим длину столбца матрицы
    //(A + i*M + j) = rand()% (M - N +1) +N;

    A =(double*)malloc(N*M * sizeof(double));

    for (i = 0; i < N; i++ ) { //ввод значений в матрицу
        for (j = 0; j < M; j++ ) {
            scanf("%d", (A + i*M + j));
        }
        puts("\n");
    }

    for (i = 0; i < N; i++ ) { //вывод получившейся матрицы
        for (j = 0; j < M; j++ ) {
            printf("%d ",*(A + i*M + j));
        }
        puts("\n");
    }

    puts("\n \n \n ");
```

```

{
int tmp;
for(int i=0;i<N;i++)//обмен значений в матрицы, зеркально вертикальной
линии
{
for(int j=0;j<M/2;j++)//обмен значений в матрицы, зеркально вертикальной
линии
{
tmp=*(A + i*M + j); //обмен значений в массиве
*(A + i*M + j)=*(A + i*M + M -j -1); //обмен значений в массиве
*(A + i*M + M -j -1)=tmp; //обмен значений в массиве
}
}
for (i = 0; i < N; i++ ) { //вывод открашенной матрицы
for (j = 0; j < M; j++ ) {
printf("%d ", *(A + i*M + j));
}
puts("\n");
}
}
free(A); //очищение памяти
return 0;
}

```

#### Пример выполнения программы

The left screenshot shows the input matrix, which is a 3x4 grid of numbers:

```

3
4
3
4
5
7
3
5
7
9
3
6
89
9

```

The right screenshot shows the output matrix, which is a 3x4 grid of numbers, representing the input matrix mirrored vertically:

```

3 4 5 7
3 5 7 9
3 6 89 9
7 5 4 3
9 7 5 3
9 89 6 3

```

Below the output matrix, the terminal displays the following text:

```

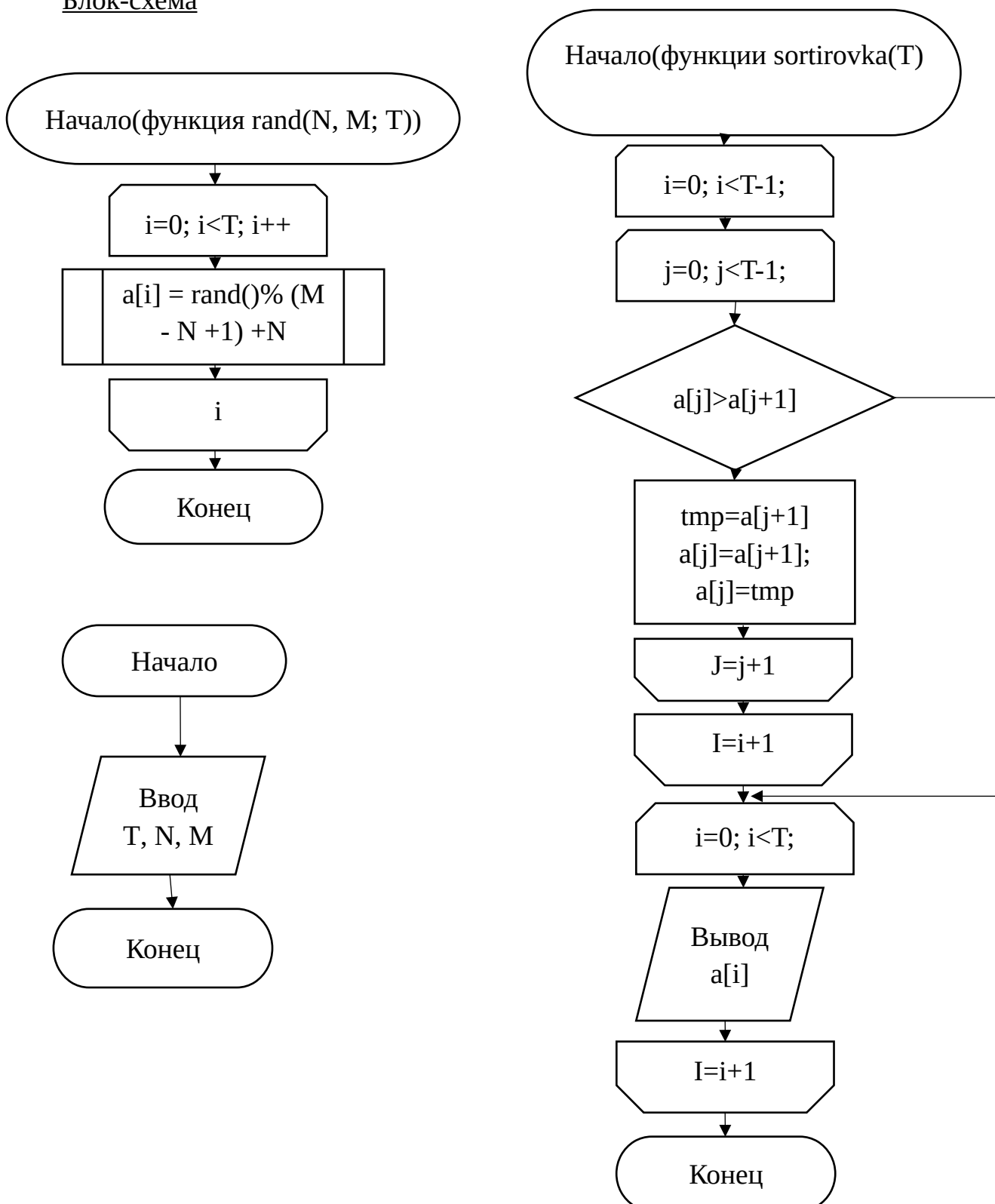
Process returned 0 (0x0)   execution time : 14.745 s
Press any key to continue.

```

## Индивидуальное задание №4

Задание: Написать программу для сортировки одномерного массива символов по возрастанию методом обмена. Размерность массива, а также границы диапазона изменения элементов  $N$  и  $M$  вводятся пользователем с клавиатуры, а сам массив заполняется случайными числами из указанного пользователем диапазона. Сортировку массива и генерацию массива оформить в виде отдельных функций.

### Блок-схема



### Описание программы:

Задаются две функции.

Rand() – функция ввода случайного числа

Sortirovka() – функция сортировки методом обмена

T – количество значений в массиве

### Исходный код программы

```
#include <stdio.h>
```

```
#include <iostream>
```

```
#include <stdlib.h>
```

```
int rand(int N, int M, int T){//функция ввода случайных чисел
```

```
int a[1000];
```

```
for(int i=0; i<T; i++){
```

```
    a[i] = rand()%(M - N +1) +N;
```

```
}
```

```
}
```

```
int sortirovka(int T){ функция сортировки массива
```

```
int a[1000], tmp;
```

```
    for(int i=0; i<T-1;i++){//посредством сравнения производим сортировку
```

обменом

```
        for(int j=0; j<T-1; j++){
```

```
            if(a[j]>a[j+1]){
```

```
                tmp=a[j+1];
```

```
                a[j]=a[j+1];
```

```
                a[j]=tmp;
```

```
            }
```

```
        }
```

```
    }
```

```
    for(int i=0; i<T; i++){Выводим получившийся массив
```

```
        printf("%i", a[i]);
```

```
    }
```

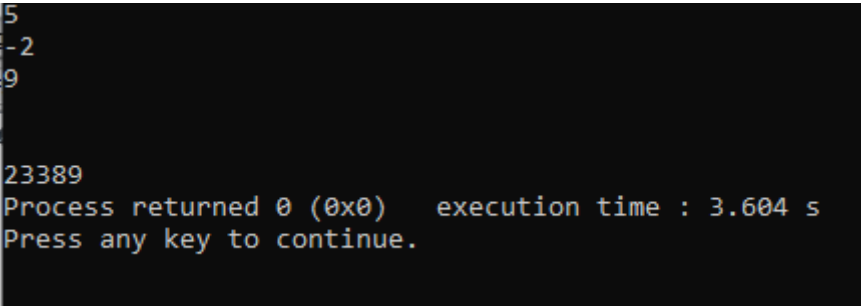
```
}
```

```
using namespace std;
```

```
int main()
{
    int a[1000], T, N, M;
    scanf("%i", &T); // Вводим количество чисел в массива
    scanf("%i %i", &N, &M); // Вводим диапазон
    rand(N, M, T); //указываем на функцию ввода рандомного числа
    sortirivka(T); //указываем на функцию сортировки массива

    return 0;
}
```

#### Пример выполнения программы



```
5
-2
9

23389
Process returned 0 (0x0)   execution time : 3.604 s
Press any key to continue.
```



**Вывод** Изучили принципы работы с указателями и структурными типами данных (массивы и структуры) в СИ. Приобрели практические навыки в реализации алгоритмов сортировки одномерных массивов.