



Министерство науки и высшего образования Российской Федерации
Мытищинский филиал
Федерального государственного бюджетного образовательного учреждения
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МФ МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ космический

КАФЕДРА К-3

ОТЧЕТ
К ЛАБОРАТОРНОЙ РАБОТЕ

№ 5

по ДИСЦИПЛИНЕ
«ИНФОРМАТИКА»

Студент К3-13Б
(Группа)

И.И. Остапенко
(И.О.Фамилия)

Преподаватель

В.В. Афанасьева
(И.О.Фамилия)

2022 г.

Задание № 5 (лабораторная работа №5)

Задача: дан массив вещественных чисел. Написать функцию, которая упорядочивает этот массив по возрастанию (с помощью алгоритма простого выбора).

Необходимо:

- 1) Определить входные и выходные данные.
- 2) Разработать блок-схему алгоритма.
- 3) Реализовать алгоритм на языке C.
- 4) Составить тесты для проверки работоспособности алгоритма.
- 5) Проверить работоспособность программы на разработанных тестах.
- 6) Оформить отчет по лабораторной работе (см. требования к отчету).

Указание: массив вводить с клавиатуры, предварительно ввести с клавиатуры количество элементов массива (делать проверку, чтобы количество элементов массива было больше или равно 1), желательно память под массив выделять динамически и не забывать её освобождать!

Сравнить быстроту работы вашей функции со стандартной функцией qsort.

ТЗ:

Входные данные: n – целое число, длина массива; $ar[]$ – массив вещественного типа;

Выходные данные: отсортированный массив $ar[]$;

i – текущий номер элемента массива, целое число;

Функция `makeArray` возвращает массив вещественных чисел:

n – целое число, длина массива;

$ar[]$ – массив вещественных чисел;

i – текущий номер элемента массива;

Функция `sort` возвращает массив вещественных чисел:

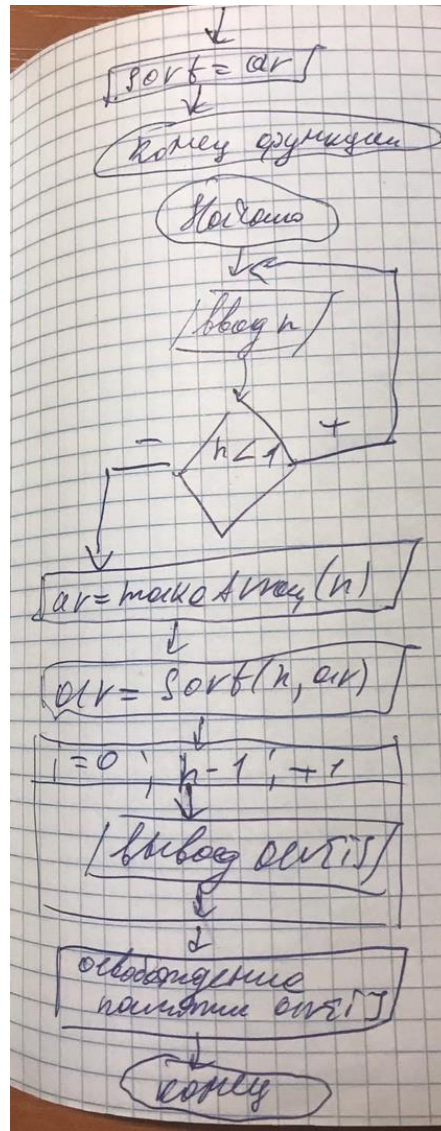
n – целое число, длина массива;

$ar[]$ – массив вещественных чисел;

i, j – счётчики, целое число;

min – вещественное число, служит для поиска минимума;

$imin$ – целое число, номер минимума;



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

```
double* makeArray(int n);
double* sort(int n, double* ar);
```

```
double* makeRandomArray(int length) {
    double* ar = (double*)malloc(sizeof(double) * length);
    double min = -100;
    double max = 100;
    for (int i = 0; i < length; i++) {
        ar[i] = (double)(rand()) / RAND_MAX * (max - min) + min;
    }
    return ar;
}
```

```
double compare(const double* first, const double* second) {
    return *first - *second;
}
```

```
double getTimeSort(void) {
    int length = 100000;
    double* ar = makeRandomArray(length);
    clock_t start, finish;
    start = clock();
    ar = sort(length, ar);
    finish = clock();
    double time = (finish - start) / CLOCKS_PER_SEC;
    free(ar);
    return time;
}
```

```
double getTimeQSort(void) {
    int length = 100000;
    double* ar = makeRandomArray(length);
    clock_t start, finish;
    start = clock();
    qsort(ar, length, sizeof(double*), (double(*) (const void *, const void *))compare);
    finish = clock();
    double time = (finish - start) / CLOCKS_PER_SEC;
    free(ar);
    return time;
}
```

```
double* sort(int n, double* ar) {
    for (int i = 0; i < n - 1; i++) {
        double min = ar[i];
        int imin = i;
        for (int j = i + 1; j < n; j++) {
            if (ar[j] < min) {
                min = ar[j];
                imin = j;
            }
        }
        ar[imin] = ar[i];
        ar[i] = min;
    }
    return ar;
}
```

```
double* makeArray(int n) {
    double* ar = (double*)malloc(sizeof(double) * n);
    for (int i = 0; i < n; i++) {
        printf("ar[%d]: ", i);
        scanf("%lf", &ar[i]);
    }
    return ar;
}
```

```
int main()
{
    int n;
    do {
        printf("Enter n: ");
        scanf("%d", &n);
    } while(n < 1);
    double* ar = makeArray(n);
    ar = sort(n, ar);
    for (int i = 0; i < n; i++)
        printf("%.3f ", ar[i]);
    free(ar);

    double time1 = getTimeSort();
    double time2 = getTimeQSort();
    if (time1 > time2) {
        printf("\nThe qsort function is faster: %f", time1 - time2);
    } else {
        printf("\nThe qsort function is slower: %f", time2 - time1);
    }

    return 0;
}
```

```
Enter n: 0
Enter n:
```

```
Enter n: 1
ar[0]: 4
4.000
```

```
Enter n: 4
ar[0]: -10
ar[1]: -102.345
ar[2]: 10.34
ar[3]: 4.56
-102.345 -10.000 4.560 10.340
```