Министерство науки и высшего образования Российской Федерации
Мытищинский филиал
Федерального государственного бюджетного образовательного учреждение 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

№ 6

по ДИСЦИПЛИНЕ

«ИНФОРМАТИКА»

Студент ____К3-13Б____ _И.И. Остапенко_____
(Группа) (И.О.Фамилия)

Преподаватель _В.В. Афанасьева____
(И.О.Фамилия)

2022 г.

Задание № 6 (лабораторная работа №6)

Задача: дан массив вещественных чисел. Написать функцию, которая упорядочивает этот массив по возрастанию (с помощью алгоритма «пузырек» с оптимизацией: если на очередном «проходе» не было ни одного обмена, то нужно заканчивать работу алгоритма).

Необходимо:

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

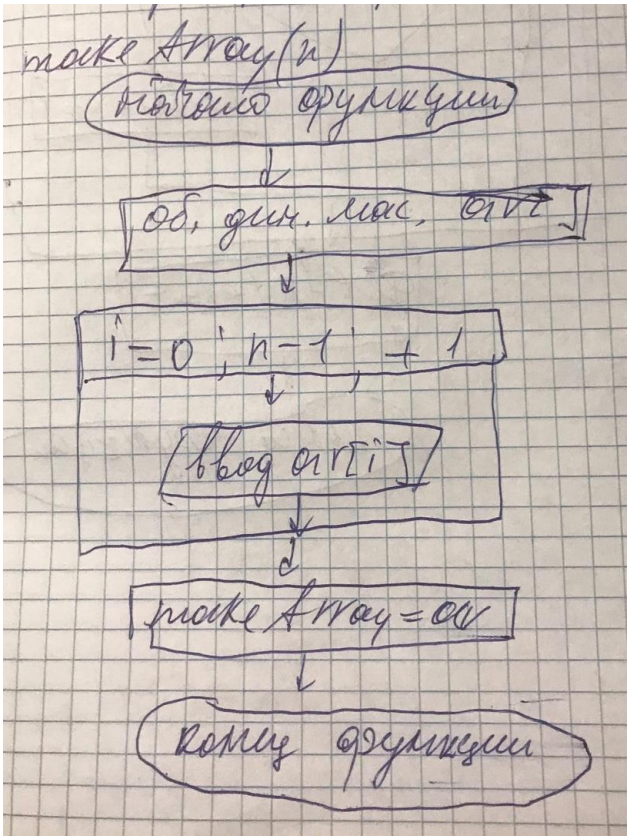
Указание: массив вводить с клавиатуры, предварительно ввести с клавиатуры количество элементов массива (делать проверку, чтобы количество элементов массива было больше или равно 1), желательно память под массив выделять динамически и не забывать её освобождать!

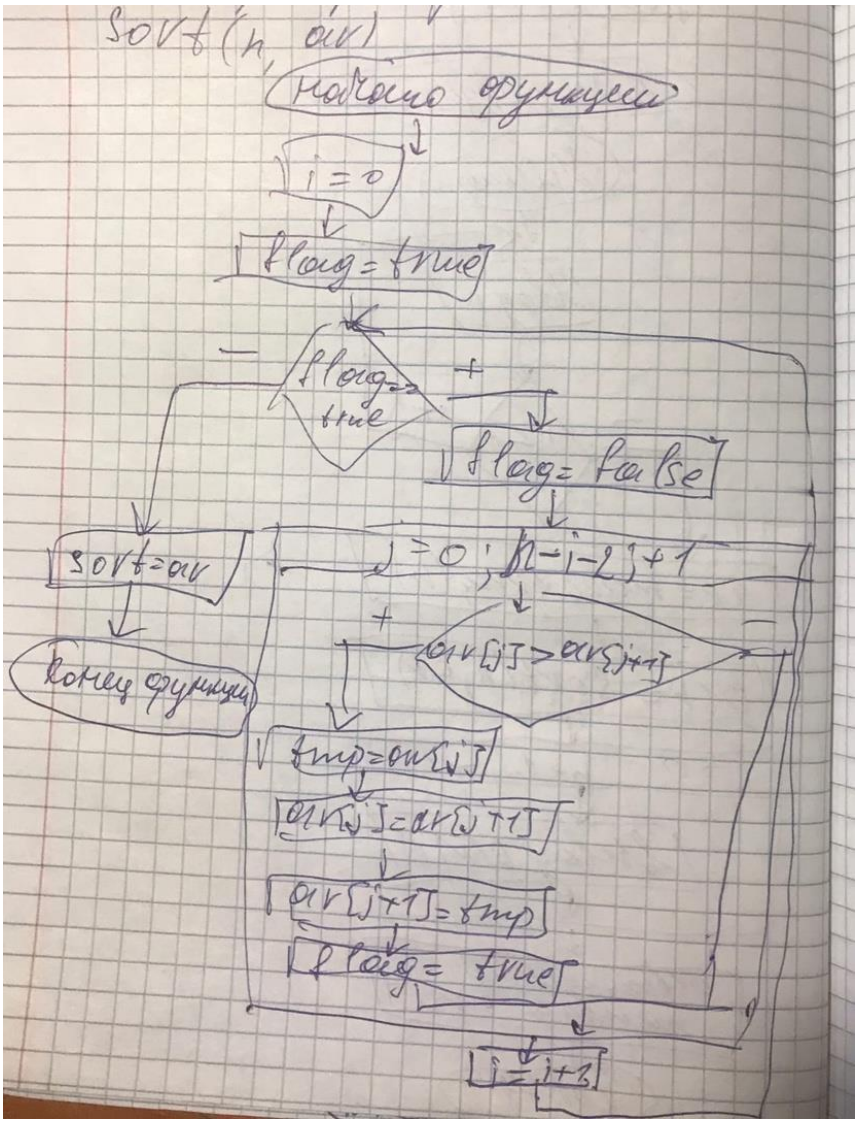
Сравнить быстроту работы вашей функции со стандартной функцией qsort.

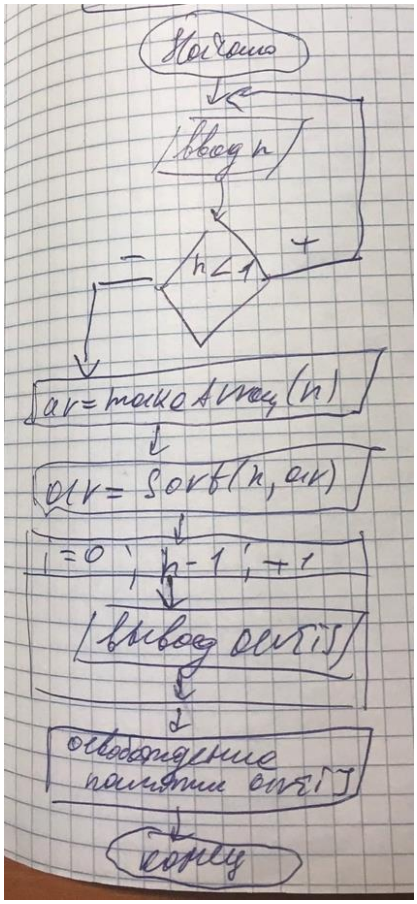
ТЗ:
Входные данные: n – целое число, длина массива; ar[] – массив вещественного типа;
Выходные данные: отсортированный массив ar[]; i – текущий номер элемента массива, целое число;

Функция makeArray возвращает массив вещественных чисел: n – целое число, длина массива; ar[] – массив вещественных чисел; i – текущий номер элемента массива, целое число;

Функция sort возвращает массив вещественных чисел:
n – целое число, длина массива;
ar[] – массив вещественных чисел;
i, j – счётчики, целое число;
flag – булевский тип, служит для прекращения цикла;
tmp – вещественное число, переменная для обмена значениями двух переменных;


make Array(n)
Начало функции
об, дин. мас, arr
i=0; n-1; +1
ввод arr[i]
make Array = arr
Конец функции


sort(n, arr)
Начало функции
i=0
flag=true
flag==true
+
flag=false
sort=arr
j=0; n-i-2+1
arr[j]>arr[j+1]
Конец функции
tmp=arr[j]
arr[j]=arr[j+1]
arr[j+1]=tmp
flag=true
j=j+1




```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <stdbool.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) {
    int i = 0;
    bool flag = true;
    while (flag) {
        flag = false;
        for (int j = 0; j < n-i-1; j++) {
            if (ar[j] > ar[j+1]) {
                double tmp = ar[j];
                ar[j] = ar[j+1];
                ar[j+1] = tmp;
                flag = true;
            }
        }
        i++;
    }
    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: -1
Enter n:
```

```
Enter n: 1
ar[0]: -45.45
-45.450
```

```
Enter n: 5
ar[0]: -3
ar[1]: 23.32
ar[2]: -10.23
ar[3]: 0
ar[4]: 299
-10.230 -3.000 0.000 23.320 299.000
```

```
Enter n: 6
ar[0]: -10
ar[1]: 23
ar[2]: 12.343
ar[3]: -123.34
ar[4]: 2
ar[5]: 2
-123.340 -10.000 2.000 2.000 12.343 23.000
```