**Проектирование архитектуры программных систем**

**Отчёт по лабораторной работа № 1**

**Тема: Реализация алгоритма сортировки пузырьком**

**Выполнила:**

**Сологуб Алина**

**Группа 21 ПИ-1**

**Алгоритм**

Сортировка простыми обменами или сортировка пузырьком (bubble sort) — это простой алгоритм сортировки. Он основан на методе повторяющихся проходов по массиву. За каждый проход элементы последовательно сравниваются попарно и, если порядок в паре неверный, то выполняется перестановка элементов. Проходы по массиву повторяются N-1 раз или до тех пор, пока на очередном проходе не окажется, что обмены больше не нужны, что означает — массив отсортирован. При каждом проходе алгоритма по внутреннему циклу очередной наибольший элемент массива ставится на своё место в конце массива рядом с предыдущим наибольшим элементом, а наименьший элемент перемещается на одну позицию к началу массива

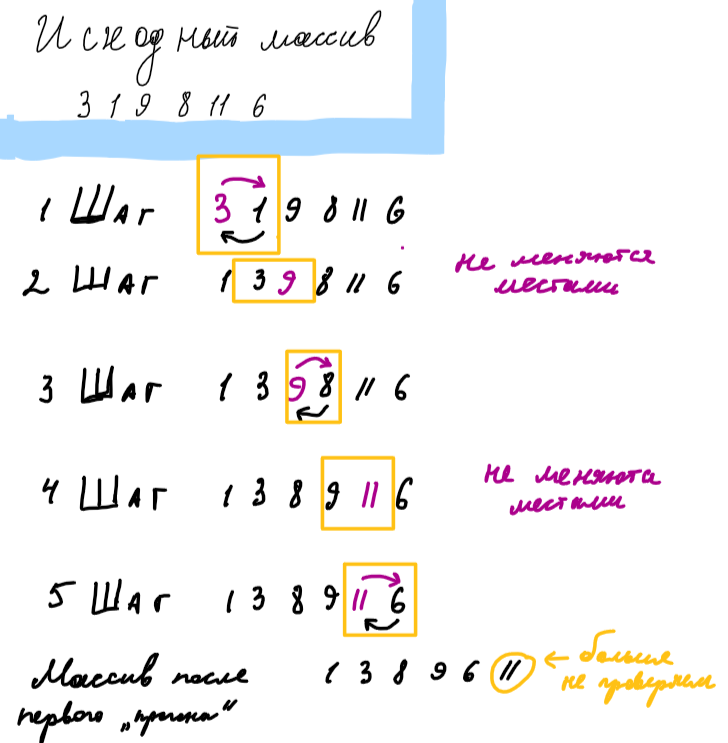
## **Сложность**

Сложность данного алгоритма O(n^2). Особенность данного алгоритма заключается в следующем: после первого завершения внутреннего цикла максимальный элемент массива всегда находится на N-1 ой позиции. При втором проходе, следующий по значению максимальный элемент находится на N месте. И так далее. Таким образом, на каждом следующем проходе число обрабатываемых элементов уменьшается на 1 и нет необходимости «обходить» весь массив от начала до конца каждый раз.

Сложность алгоритма сортировки пузырьком оценивается следующим образом:

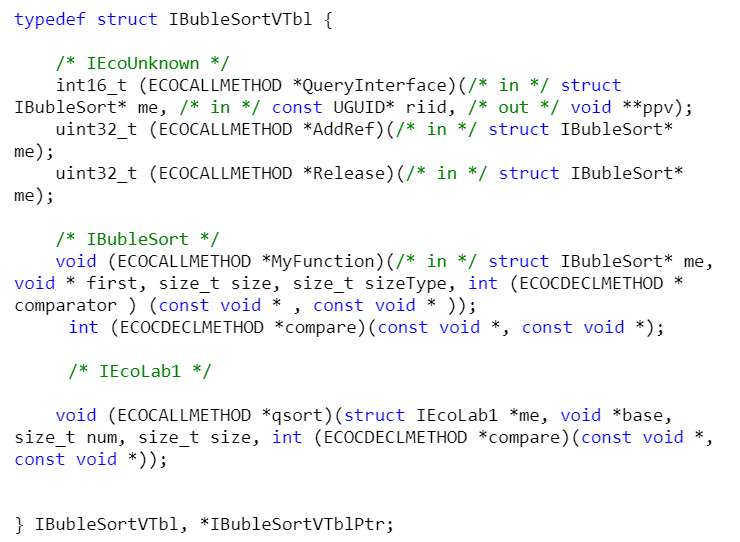
* В худшем случае, когда массив полностью не отсортирован и требуется выполнить максимальное количество итераций, время выполнения составляет O(n^2), где n - количество элементов в массиве.
* В лучшем случае, когда массив уже отсортирован, время выполнения составляет O(n), так как в таком случае нужно всего один проход по массиву для проверки отсутствия обменов.

Таким образом, **средняя временная сложность** сортировки пузырьком составляет **O(n^2)**.



*Пример работы сортировки при первом прогоне*

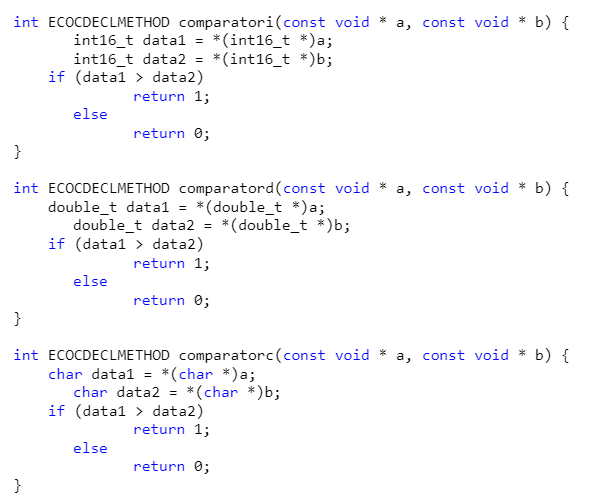
**Реализация на языке Си**

****

*Реализация структуры интерфейса*

## 

*Реализация алгоритма сортировки пузырьком*

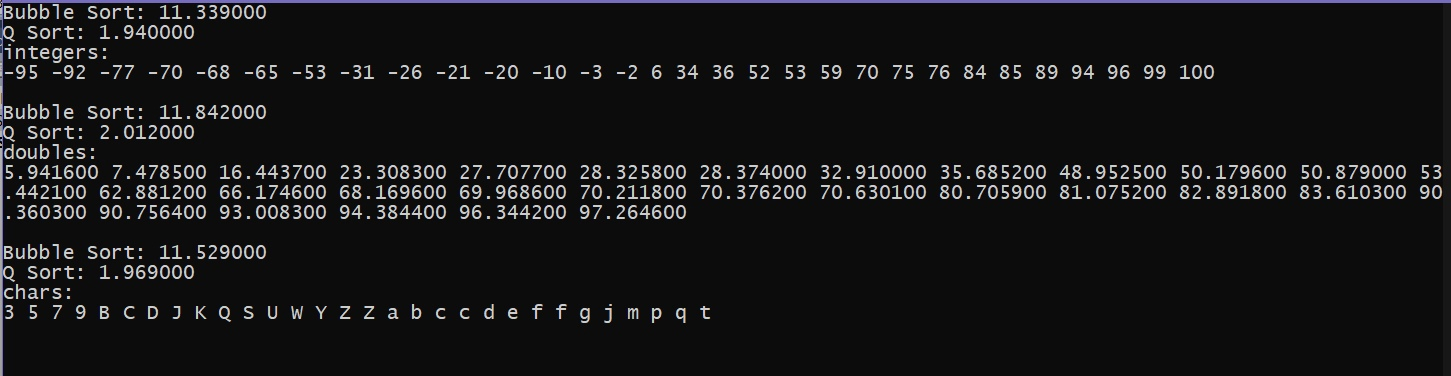
******

*Реализация компараторов*

## **Тестирование**

Для проверки работоспособности написан юнит тест, который запускает сортировку массивов трёх типов, таких как: int, double, char \*. Объявлены 3 статических массива размером 30 элементов каждым типом данных.

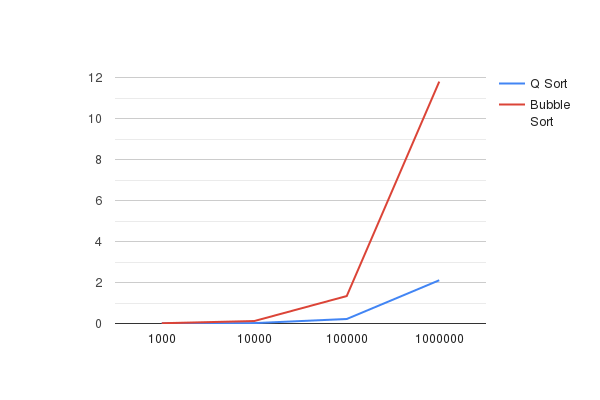
Для сравнения реализованного алгоритма была подключена при помощи #include <stdlib.h> библиотечная функция qsort. Также для замера времени работы данных функций использована библиотека <time.h>. Для сравнения время измерялось на массивах размерами 1000, 10000, 100000 и 1000000 элементов.



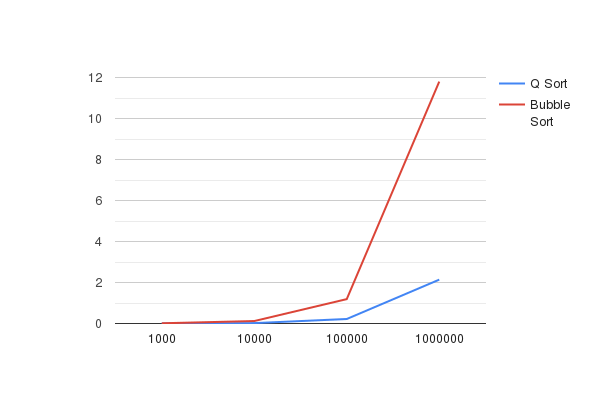
*Пример вывода данных на массивах размером 1000000 элементов*

|  | **int16\_t** | | | | **double\_t** | | | | **char\_t** | | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **1000** | **10000** | **100000** | **1000000** | **1000** | **10000** | **100000** | **1000000** | **1000** | **10000** | **100000** | **1000000** |
| **bubble sort** | 0.013 | 0.12 | 1.34 | 11.339 | 0.011 | 0.12 | 1.19 | 11.842 | 0.013 | 0.12 | 1.14 | 11.529 |
| **qsort** | 0.003 | 0.02 | 0.22 | 1.940 | 0.003 | 0.02 | 0.22 | 2.012 | 0.003 | 0.02 | 0.22 | 1.969 |
| **\*Единицы измерения в секундах, а также представлены в таблице в округленном виде** | | | | | |  |  |  |  |  |  |  |

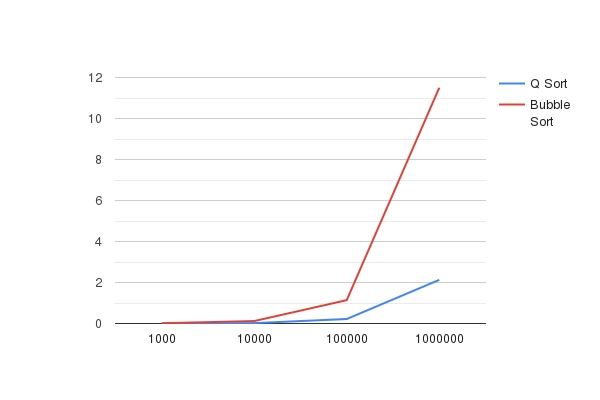
*Сравнительная таблица*



*График сравнения qsort и bubble sort с типом данных int*



*График сравнения qsort и bubble sort с типом данных double*



*График сравнения qsort и bubble sort с типом данных char \**

Оказалось, что дольше всего алгоритмы сортируют массивы с типом double, а быстрее всего массивы с типом int.

По полученным данным можно сделать вывод о том, что алгоритм сортировки пузырьком менее целесообразно использовать, чем qsort из библиотеки подключенной с помощью #include <stdlib.h> , которая обеспечивает выполнение сортировки за минимальное время, особенно это сказывается при больших массивах. На массиве размером 1000000 разница составила 9830 ms.