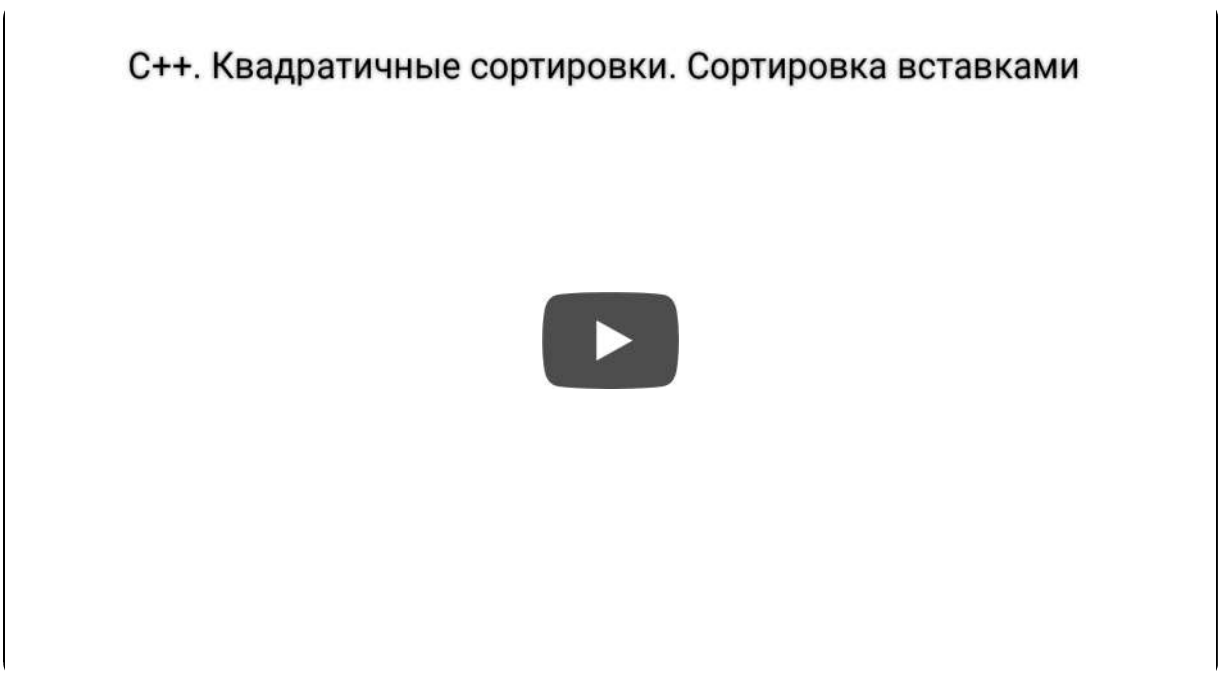
Сортировка вставками



Рассмотрим ещё один алгоритм. Будем сортировать список, добавляя в рассмотрение его элементы по одному, в том порядке, в каком они изначально находятся. Изначально имеем отсортированный список из одного элемента. Затем берём следующий элемент и вставляем его на нужное место в уже отсортированном списке, сдвигая на один всех соседей справа. На каждом шаге мы будем увеличивать отсортированную область на один элемент. Такой алгоритм называется **сортировка вставками**. Именно она часто используется в жизни при сортировке материальных предметов.

Рассмотрим на примере списка:

$$[5, 3, 9, 2, 4]$$

Проиллюстрируем работу алгоритма таблицей:

Отсортированная часть списка	Рассматриваемый элемент списка	Список
[]	a_0	[5, 3, 9, 2, 4]
[5]	a_1	[5, 3, 9, 2, 4]
[3, 5]	a_2	[3, 5, 9, 2, 4]
[3, 5, 9]	a_3	[3, 5, 9, 2, 4]

[2, 3, 5, 9]	a_4	[2, 3, 5, 9, 4]
[2, 3, 4, 5, 9]	-	[2, 3, 4, 5, 9]

Особенностью алгоритма сортировки вставками является то, что он работает быстро на упорядоченном или близком к упорядоченному массиве. Например, если мы запустим его на уже отсортированном массиве, то алгоритм выполнит ровно $n - 1$ сравнение.

Однако, если мы рассмотрим худший случай и запустим алгоритм на упорядоченном по убыванию массиве, то он выполнит $1 + 2 + \ldots + (n - 2) + (n - 1) = n(n - 1)/2$ операций. Следовательно, асимптотика алгоритма составляет $O(n^2)$.

Реализация

```
void insertion_sort(vector<int> &a)
{
    for (int i = 1; i < a.size(); ++i)
    {
        int tmp = a[i];
        int j = i - 1;
        while (j >= 0 && a[j] > tmp)
        {
            a[j + 1] = a[j];
            --j;
        }
        a[j + 1] = tmp;
    }
}
```