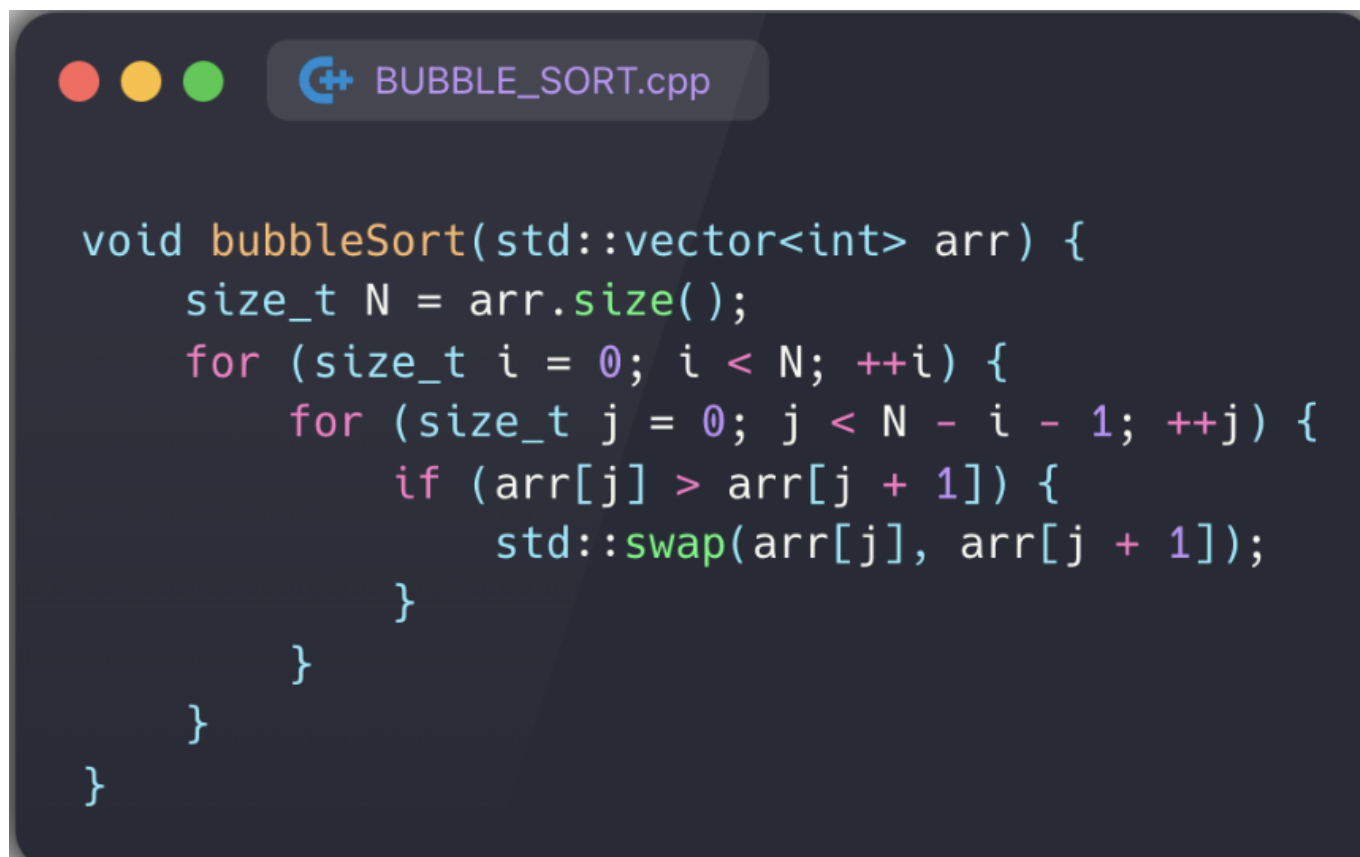


ДЗ 1 Анализ алгоритмов

Сортировка пузырьком

A screenshot of a code editor window titled "BUBBLE_SORT.cpp". The code implements the bubble sort algorithm using a vector. It features two nested loops: an outer loop for 'i' from 0 to N-1, and an inner loop for 'j' from 0 to N-i-1. Inside the inner loop, it compares adjacent elements and swaps them if they are in the wrong order. The code is as follows:

```
void bubbleSort(std::vector<int> arr) {
    size_t N = arr.size();
    for (size_t i = 0; i < N; ++i) {
        for (size_t j = 0; j < N - i - 1; ++j) {
            if (arr[j] > arr[j + 1]) {
                std::swap(arr[j], arr[j + 1]);
            }
        }
    }
}
```

Доказательство через инвариант

Инвариант возьмем такой:

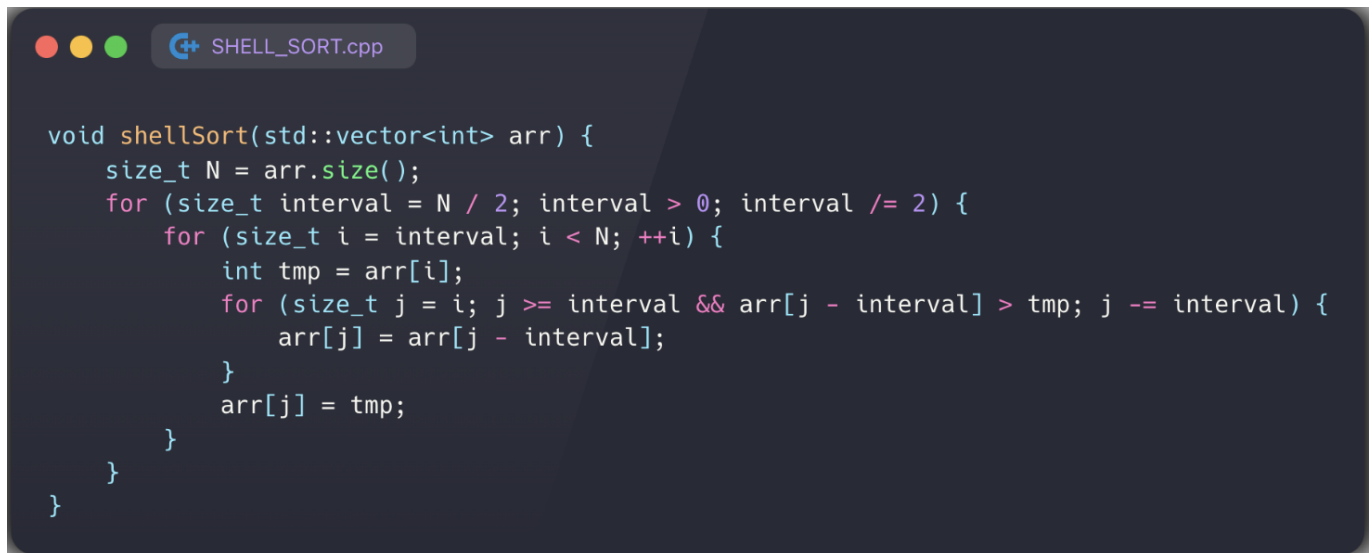
В конце i -того цикла последние $i + 1$ элементов стоят на своих местах (т.е. стоят так, как стояли бы в отсортированном массиве)

на 0 шаге это выполняется (в цикле по j мы отправим самый больший элемент в конец)

на i -том шаге у нас неотсортированными остаются $n - i$ первых элементов (потому что инвариант работал на $i - 1$ шаге). Максимальный из них встанет на $N - i - 1$ место благодаря циклу по j . Инвариант выполняется.

В конце последнего шага окажется, что последние $i + 1 = N$ элементов отсортированы. инвариант выполняется.

Алгоритм Шелла



```

void shellSort(std::vector<int> arr) {
    size_t N = arr.size();
    for (size_t interval = N / 2; interval > 0; interval /= 2) {
        for (size_t i = interval; i < N; ++i) {
            int tmp = arr[i];
            for (size_t j = i; j >= interval && arr[j - interval] > tmp; j -= interval) {
                arr[j] = arr[j - interval];
            }
            arr[j] = tmp;
        }
    }
}

```

Временная оценка сложности

рассмотрим худший случай

худший случай для 8 выглядит так:

8, 4, 6, 2, 7, 3, 5, 1

Массив выстроен так, что для каждой последовательности (подстрока соседние элементы которой удалены на интервал) длины k количество свопов для сортировки вставками будет равно

$$1 + 2 + \dots + k/2 = \frac{k(k/2+1)}{4}$$

Каждое такое действие умножаем на 3 (помимо свопа надо будет сделать проверку условия и уменьшить j)

тогда

$$T(N) = \sum_1^{\log_2 N} \left(\frac{N}{2^i} 3 \frac{2^{i-1}(2^{i-1}+1)}{4} \right) \rightarrow \frac{3}{4} N \log N$$