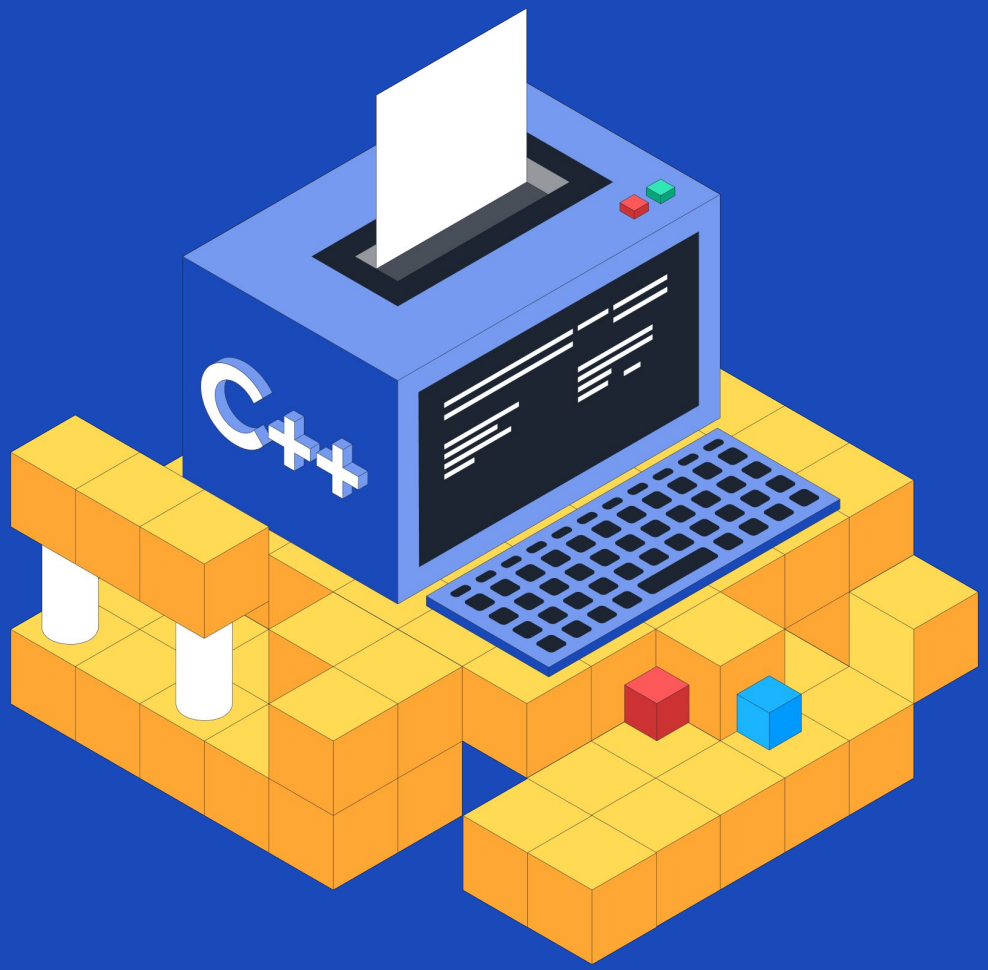
ДЕТСКАЯ ОНЛАЙН-ШКОЛА ПРОГРАММИРОВАНИЯ
HELLO WORLD

#ШПАРГАЛОЧКИ

ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ C++

Материалы подготовлены отделом методической разработки

Средний уровень



Алгоритмы сортировки вставками и Шелла

Алгоритм сортировки вставками

Алгоритм сортировки вставками - еще один несложный алгоритм сортировки. Он берет по одному элементу из исходного массива и перемещает в новый массив, сравнивая с уже хранящимися там элементами и вставляя в нужное место.

На практике, новый массив можно не создавать. Вместо этого можно сортировать исходный, сравнивая элементы в нем (начиная со второго) с соседями слева.

Преимущества и недостатки

Сортировка вставками проста для понимания, легко реализуется. Подходит для работы с частью массива (когда не все его элементы уже известны).

В остальном сохраняет те же недостатки (низкую скорость выполнения), что и сортировка пузырьком и выбором.

Реализация

Для реализации сортировки выбором достаточно следующих шагов:

1. Взять первый элемент из исходного массива и перенести в конец нового.
2. Сравнить перенесенный элемент с соседом слева, если перенесенный элемент меньше - поменять элементы местами.
3. Повторять шаг 2, пока перенесенный элемент не будет больше или равен соседу слева или не дойдет до начала массива.
4. Повторить шаги 1-3.

Реализация

```
for (int i = 1; i < length; i++) {

    // поскольку сортируем без второго массива, начинаем перебирать исходный со второго элемента

    int current = array[i];

    int prevIndex = i - 1;

    while (prevIndex >= 0 && current < array[prevIndex] ) {

        array[prevIndex + 1] = array[prevIndex];

        array[prevIndex] = current;

        prevIndex--;

    }

}
```

[3, 1, 7, 5, 2]

[1, 3, 7, 5, 2]

[1, 3, 7, 5, 2]

[1, 3, 5, 7, 2]

[1, 3, 5, 2, 7]

[1, 3, 2, 5, 7]

[1, 2, 3, 5, 7]

Сортировка Шелла

Сортировка Шелла - это усовершенствованная сортировка вставками, которая использует предварительные “**грубые**” проходы. Грубые - потому что сравниваются и меняются местами не соседние элементы (находящиеся в одном шаге друг от друга), а те, которые находятся в б**о**льшем количестве шагов.

Этот алгоритм позволяет “причесать” массив перед окончательной сортировкой, что уменьшает число перестановок. Начальный шаг обычно равен половине длины массива. После каждого прохода по массиву шаг уменьшается, пока не дойдет до одного.

Реализация

12	8	14	6	4	9	1	8	13	5	11	3	18	3	10	9

Попробуем рассмотреть реализацию на примере массива из 16 элементов. Сначала нужно определить **размер шага** (16 / 2 = 8). А дальше начинаем сравнивать элементы, находящиеся на расстоянии 8 шагов друг от друга (**array[i] и array[i + 8]**). Как только дойдем до конца массива (**i + 8 > 15**), уменьшим шаг в 2 раза.

Проходим по массиву с шагом 8

Первая группа {12, 8, 14, 6, 4, 9, 1 ,8 ,13, 5, 11, 3, 18, 3 ,10, 9} – не меняем

Вторая группа {12, 8, 14, 6, 4, 9, 1 ,8 ,13, 5, 11, 3, 18, 3 ,10, 9} – меняем

{12, 5, 14, 6, 4, 9, 1 ,8 ,13, 8, 11, 3, 18, 3 ,10, 9} - поменяли

Третья группа {12, 5, 14, 6, 4, 9, 1 ,8 ,13, 8, 11, 3, 18, 3 ,10, 9} – меняем

{12, 5, 11, 6, 4, 9, 1 ,8 ,13, 8, 14, 3, 18, 3 ,10, 9} - поменяли

Четвертая гр. {12, 5, 11, 6, 4, 9, 1 ,8 ,13, 8, 14, 3, 18, 3 ,10, 9} – меняем

{12, 5, 11, 3, 4, 9, 1 ,8 ,13, 8, 14, 6, 18, 3 ,10, 9} - поменяли

Пятая группа {12, 5, 11, 3, 4, 9, 1 ,8 ,13, 8, 14, 6, 18, 3 ,10, 9} – не меняем

Шестая группа {12, 5, 11, 3, 4, 9, 1 ,8 ,13, 8, 14, 6, 18, 3 ,10, 9} – меняем

{12, 5, 11, 3, 4, 3, 1 ,8 ,13, 8, 14, 6, 18, 9 ,10, 9} - поменяли

Седьмая гр. {12, 5, 11, 3, 4, 3, 1 ,8 ,13, 8, 14, 6, 18, 9 ,10, 9} – не меняем

Восьмая гр. {12, 5, 11, 3, 4, 3, 1, 8, 13, 8, 14, 6, 18, 9, 10, 9} – не меняем

ИТОГ {12, 5, 11, 3, 4, 3, 1, 8, 13, 8, 14, 6, 18, 9, 10, 9}

Проходим по массиву с шагом 4

Первая группа {12, 5, 11, 3, 4, 3, 1, 8, 13, 8, 14, 6, 18, 9, 10, 9} – нужна сортировка

{4, 5, 11, 3, 12, 3, 1, 8, 13, 8, 14, 6, 18, 9, 10, 9} – отсортировали

Вторая группа {4, 5, 11, 3, 12, 3, 1, 8, 13, 8, 14, 6, 18, 9, 10, 9} – нужна сортировка

{4, 3, 11, 3, 12, 5, 1, 8, 13, 8, 14, 6, 18, 9, 10, 9} – отсортировали

Третья группа {4, 3, 11, 3, 12, 5, 1, 8, 13, 8, 14, 6, 18, 9, 10, 9} – нужна сортировка

{4, 3, 1, 3, 12, 5, 10, 8, 13, 8, 11, 6, 18, 9, 14, 9} – отсортировали

Четвертая гр. {4, 3, 1, 3, 12, 5, 10, 8, 13, 8, 11, 6, 18, 9, 14, 9} – нужна сортировка

{4, 3, 1, 3, 12, 5, 10, 6, 13, 8, 11, 8, 18, 9, 14, 9} – отсортировали

ИТОГ {4, 3, 1, 3, 12, 5, 10, 6, 13, 8, 11, 8, 18, 9, 14, 9}

Проходим по массиву с шагом 2

Первая группа {4, 3, 1, 3, 12, 5, 10, 6, 13, 8, 11, 8, 18, 9, 14, 9} - нужна сортировка

{1, 3, 4, 3, 10, 5, 11, 6, 12, 8, 13, 8, 14, 9, 18, 9} – отсортировали

Вторая группа {1, 3, 4, 3, 10, 5, 11, 6, 12, 8, 13, 8, 14, 9, 18, 9} - не нужна сортировка

ИТОГ {1, 3, 4, 3, 10, 5, 11, 6, 12, 8, 13, 8, 14, 9, 18, 9}

И последний этап, проходим по массиву с шагом 1

{1, 3, 4, 3, 10, 5, 11, 6, 12, 8, 13, 8, 14, 9, 18, 9}
{1, 3, 3, 4, 10, 5, 11, 6, 12, 8, 13, 8, 14, 9, 18, 9}
{1, 3, 3, 4, 5, 10, 11, 6, 12, 8, 13, 8, 14, 9, 18, 9}
{1, 3, 3, 4, 5, 10, 6, 11, 12, 8, 13, 8, 14, 9, 18, 9}
{1, 3, 3, 4, 5, 10, 6, 11, 8, 12, 13, 8, 14, 9, 18, 9}
{1, 3, 3, 4, 5, 10, 6, 8, 11, 12, 13, 8, 14, 9, 18, 9}
{1, 3, 3, 4, 5, 10, 6, 8, 11, 12, 8, 13, 14, 9, 18, 9}
{1, 3, 3, 4, 5, 10, 6, 8, 11, 8, 12, 13, 14, 9, 18, 9}
{1, 3, 3, 4, 5, 10, 6, 8, 8, 11, 12, 13, 14, 9, 18, 9}
{1, 3, 3, 4, 5, 10, 6, 8, 8, 11, 12, 13, 9, 14, 18, 9}
{1, 3, 3, 4, 5, 10, 6, 8, 8, 11, 12, 9, 13, 14, 18, 9}
{1, 3, 3, 4, 5, 10, 6, 8, 8, 11, 9, 12, 13, 14, 18, 9}
{1, 3, 3, 4, 5, 10, 6, 8, 8, 9, 11, 12, 13, 14, 18, 9}
{1, 3, 3, 4, 5, 10, 6, 8, 8, 9, 11, 12, 13, 14, 9, 18}
{1, 3, 3, 4, 5, 10, 6, 8, 8, 9, 11, 12, 13, 9, 14, 18}
{1, 3, 3, 4, 5, 10, 6, 8, 8, 9, 11, 12, 9, 13, 14, 18}
{1, 3, 3, 4, 5, 10, 6, 8, 8, 9, 11, 9, 12, 13, 14, 18}
Итоговый массив:
{1, 3, 3, 4, 5, 10, 6, 8, 8, 9, 9, 11, 12, 13, 14, 18}

Реализация

```
for (int increment = size / 2; increment > 0; increment = increment / 2) {

    for (int i = increment; i < size; i++) {

        int j = i;

        int current = num[i];

        while ((j >= increment) && ( current < num[j - increment] )) {

            num[j] = num[j - increment];

            j = j - increment;

        }

        num[j] = current;

    }

}
```