Тема 3.1-3.5. Алгоритмы. Работа в VS Code с алгоритмами сортировки и поиска

Выполнил студент группы 15.27Д-ПИ03/25б

Малышева Дарья Андреевна

1. **Сортировка выбором (Selection Sort):**

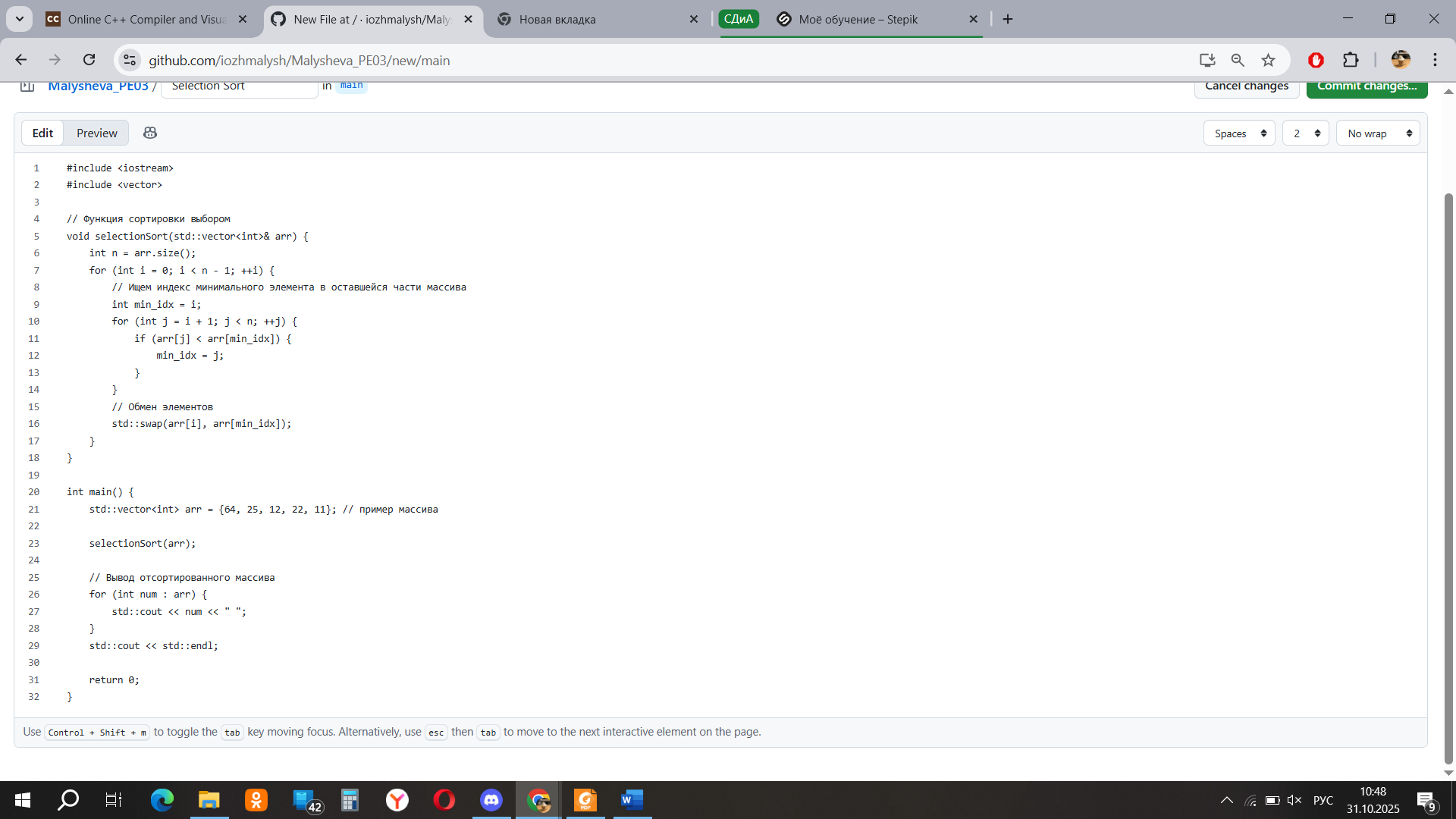


Рисунок 1 – Код сортировки выбором

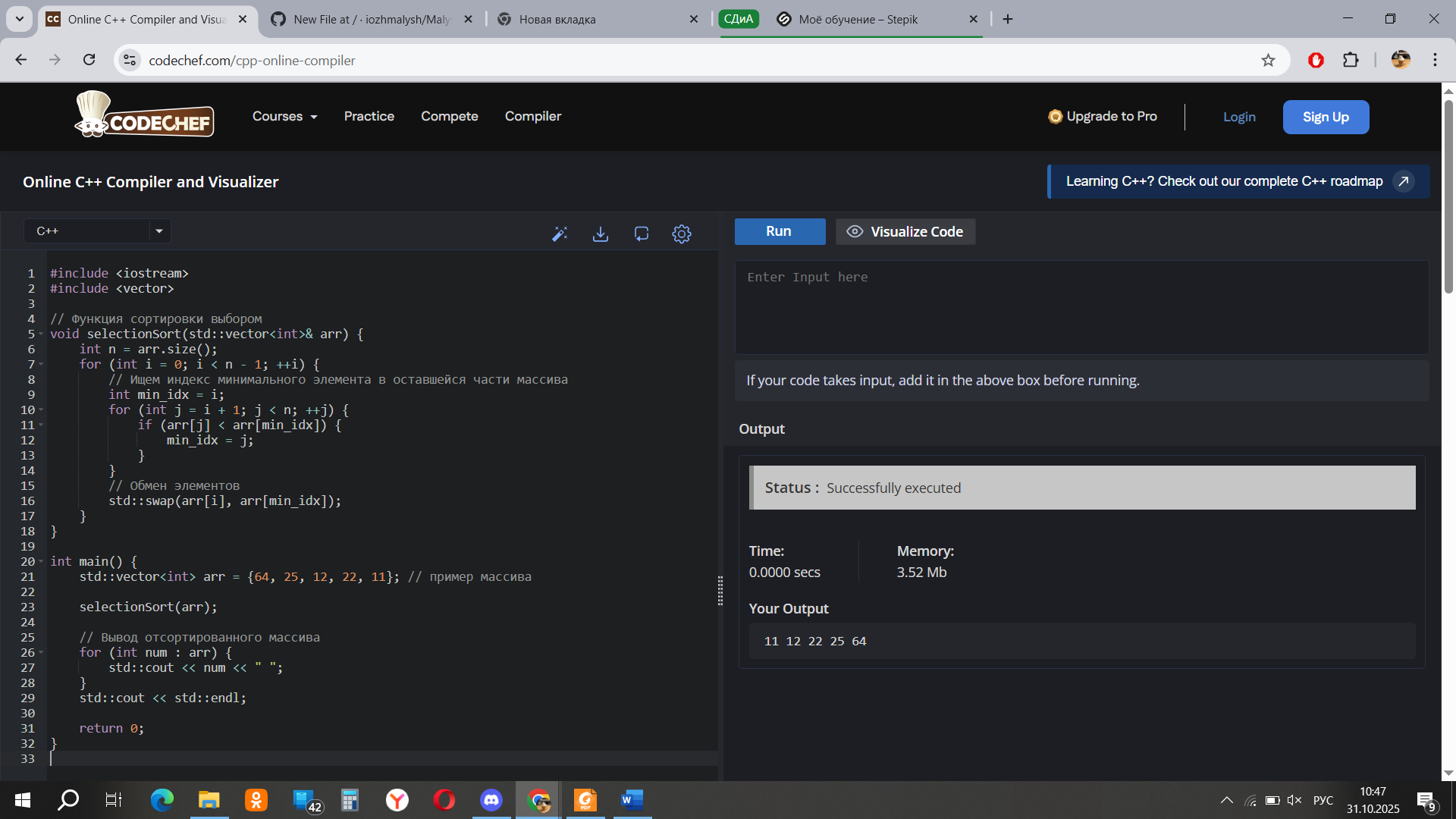


Рисунок 4 – Проверка кода сортировки выбором

Временная сложность алгоритма:

* O(n²) в худшем, среднем и лучшем случаях, где n -количество элементов в массиве.

1. Сортировка обменом (пузырьком) (Bubble Sort):

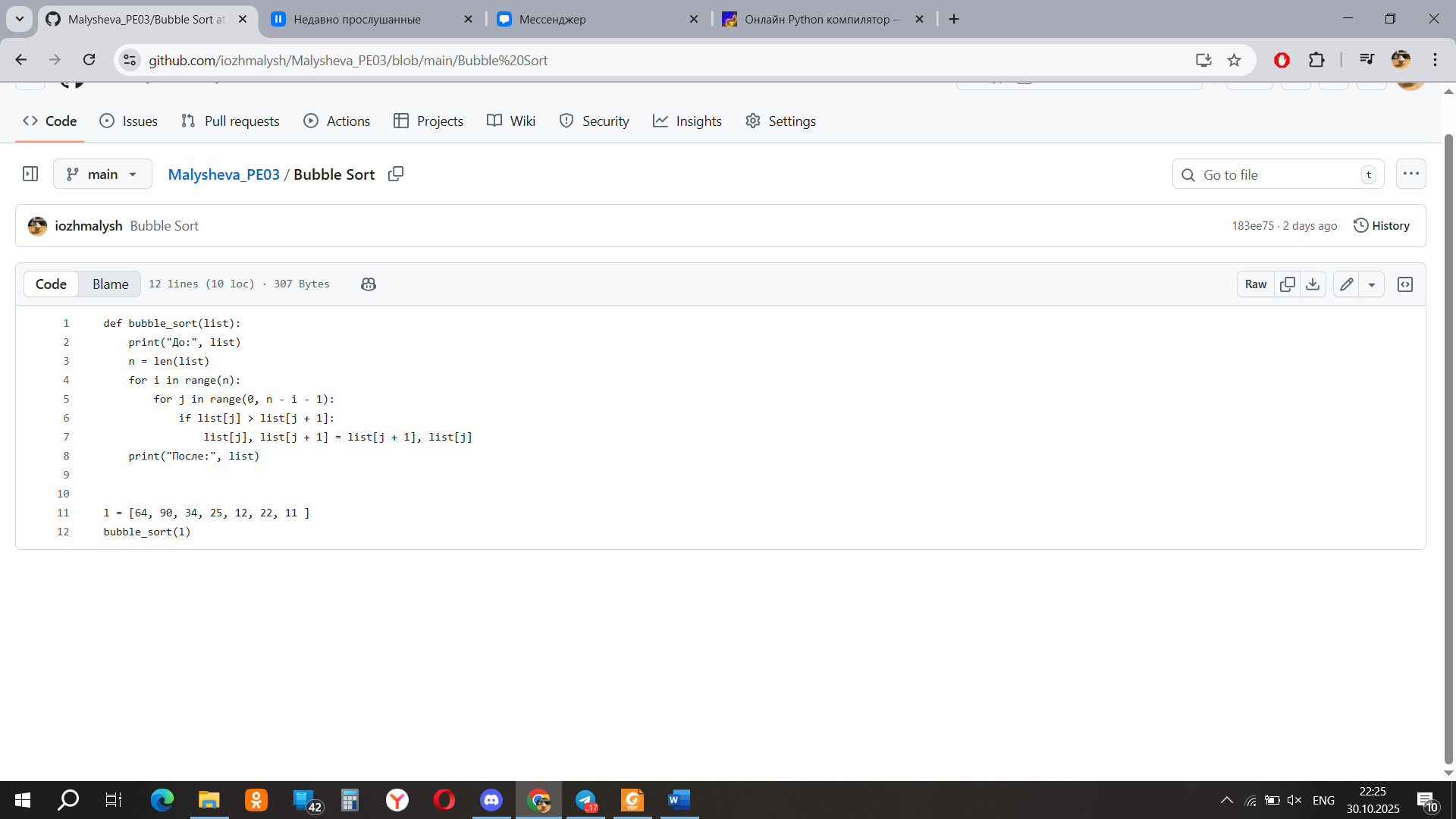


Рисунок 3 – Код сортировки обменом

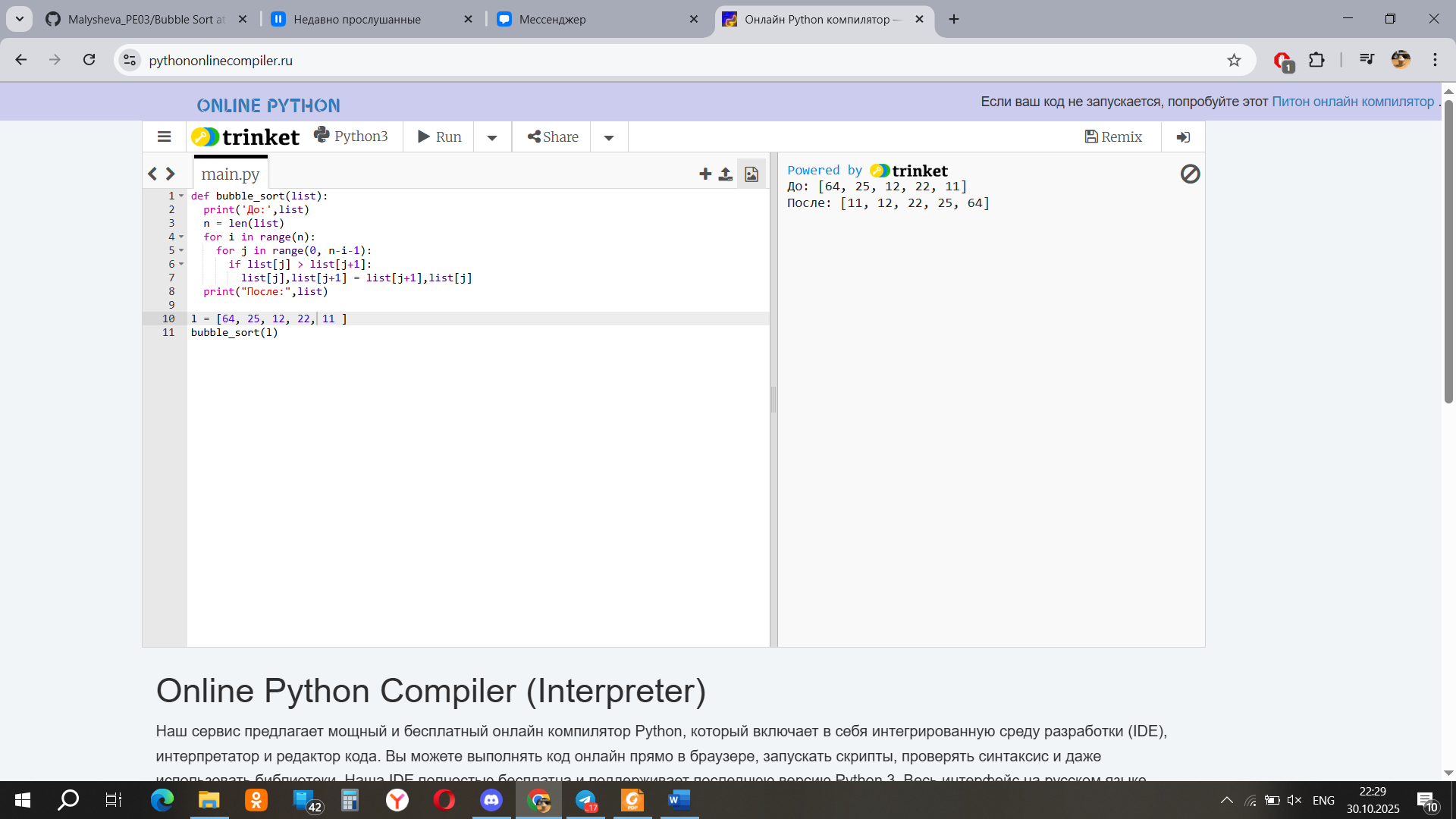


Рисунок 4 – Проверка кода сортировки обменом

Временная сложность алгоритма:

* О(n) – при отсортированном массиве (лучший случай)
* О(n2)- массив расположен в случайном порядке (средний случай), когда массив отсортирован в обратном порядке (худший случай)

1. Сортировка вставками (Insertion Sort):

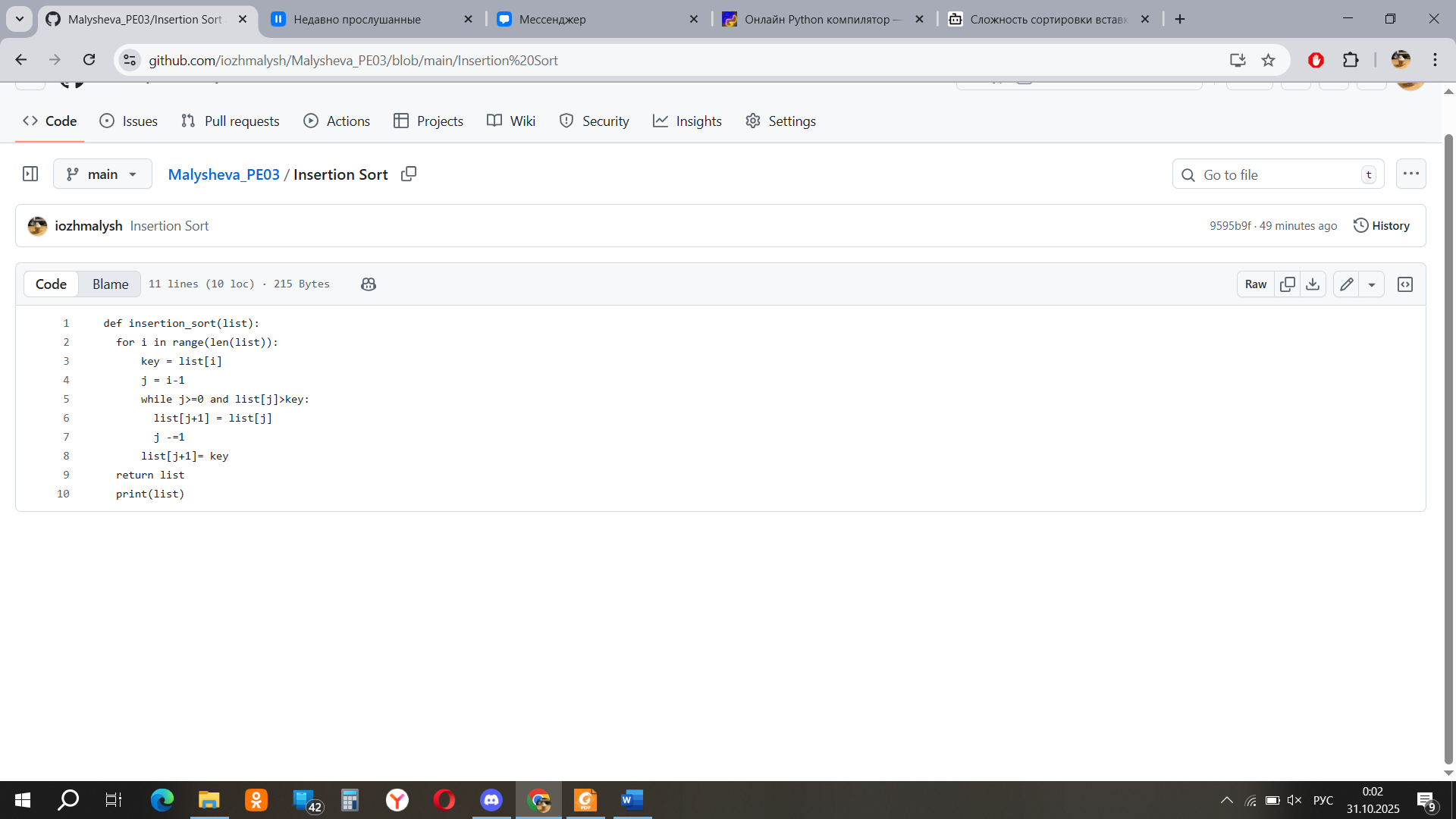


Рисунок 5 – Код сортировки вставками



Рисунок 6 – Проверка кода сортировки вставками

Временная сложность алгоритма:

* О(n) – при отсортированном массиве (лучший случай)
* О(n2)- массив расположен в случайном порядке (средний случай), когда массив отсортирован в обратном порядке (худший случай)

1. **Сортировка слиянием**

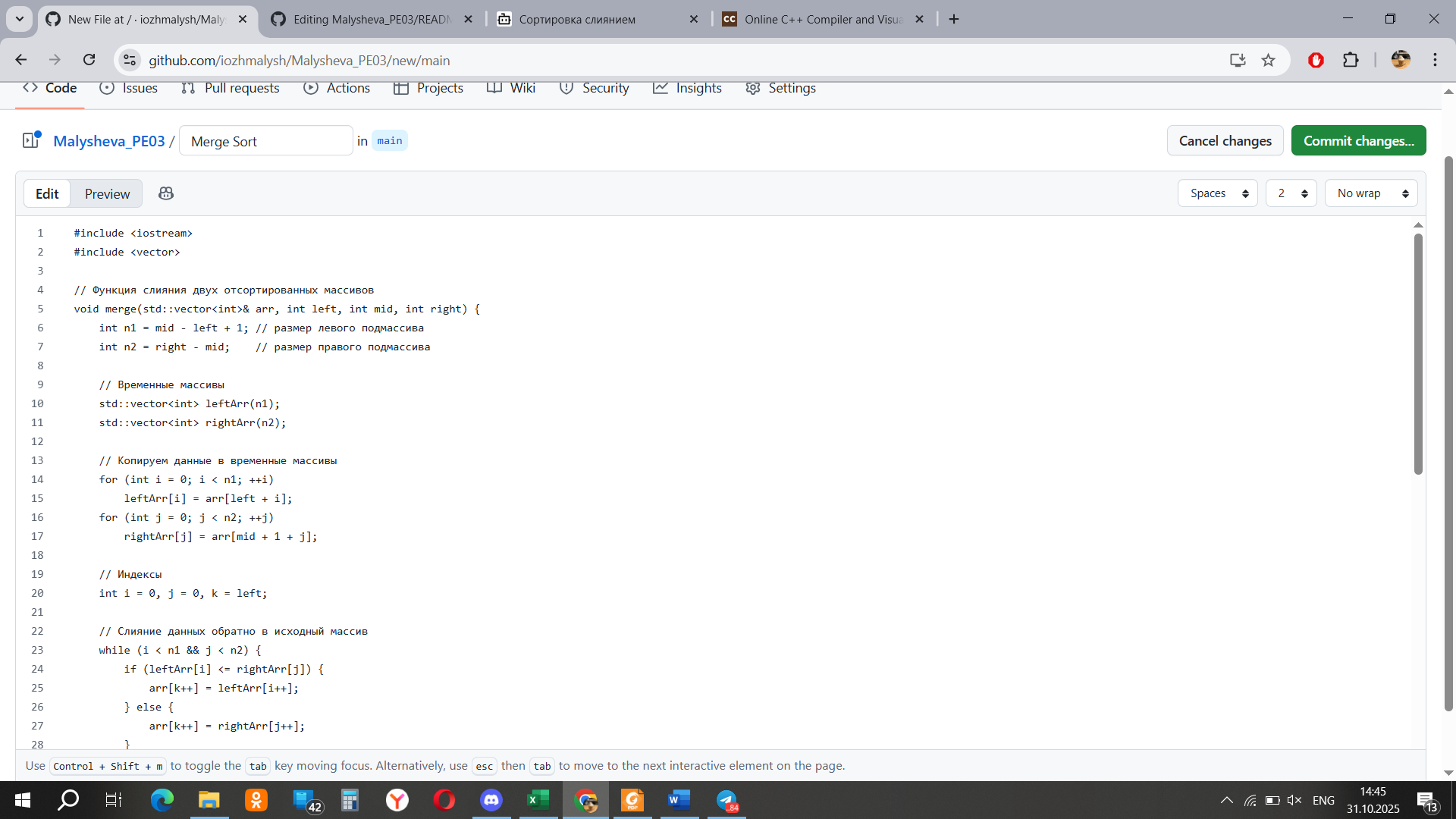
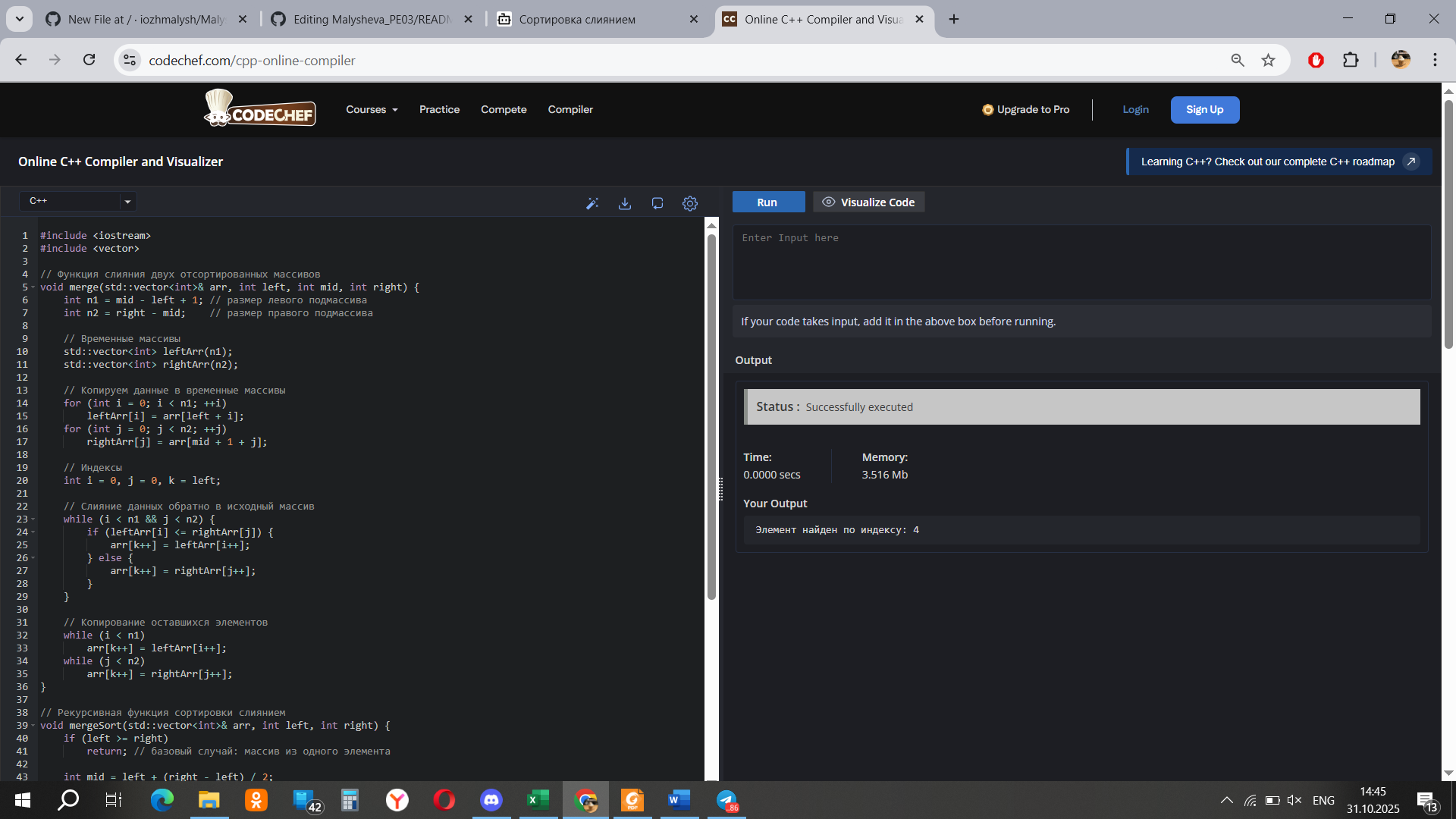


Рисунок 7 – Код сортировки слиянием



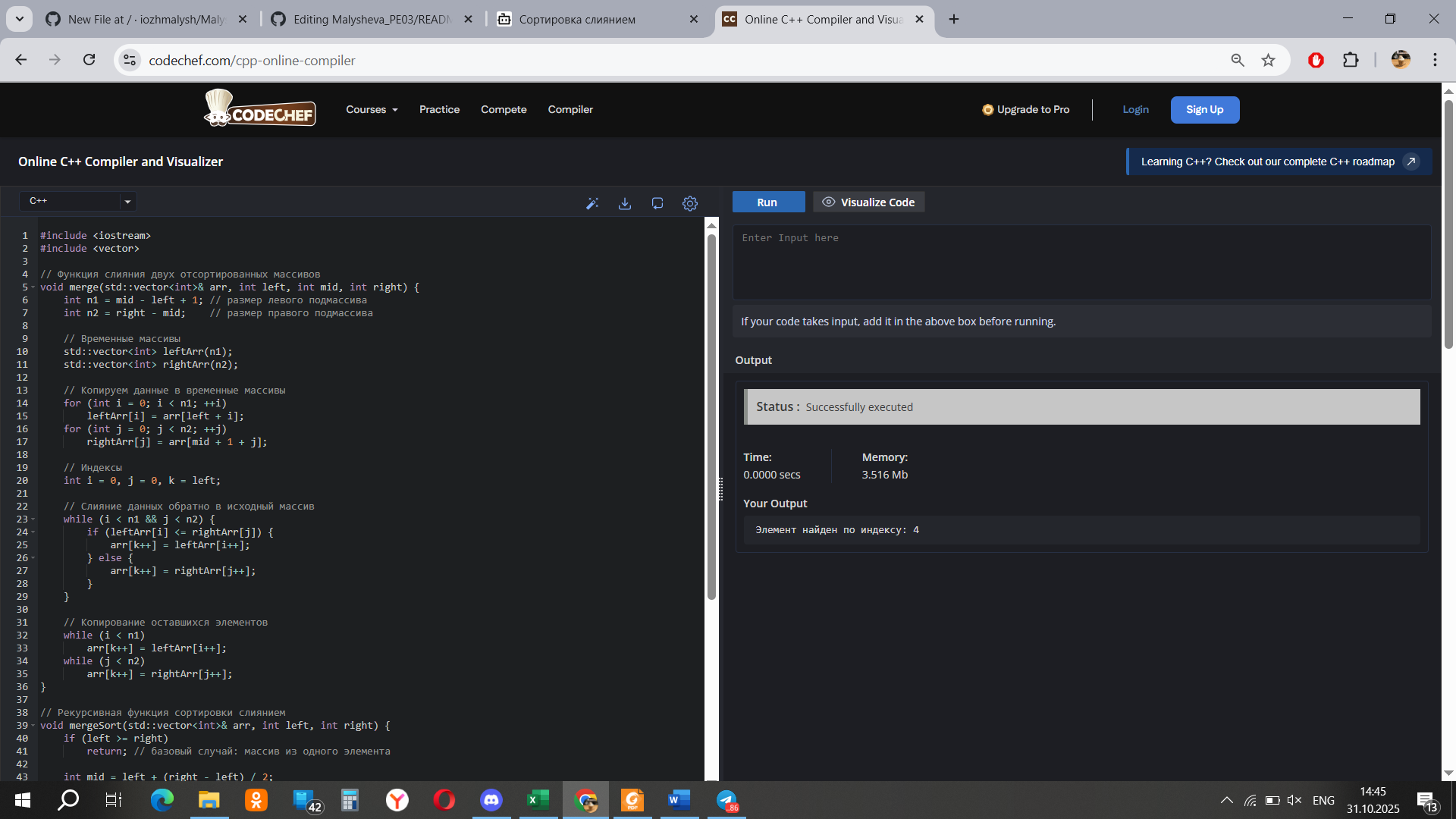


Рисунок 8 – Проверка кода сортировки слиянием

Временная сложность алгоритма:

* O (n log n) в худшем, среднем и лучшем случаях

1. **Сортировка Шелла (Shellsort):**

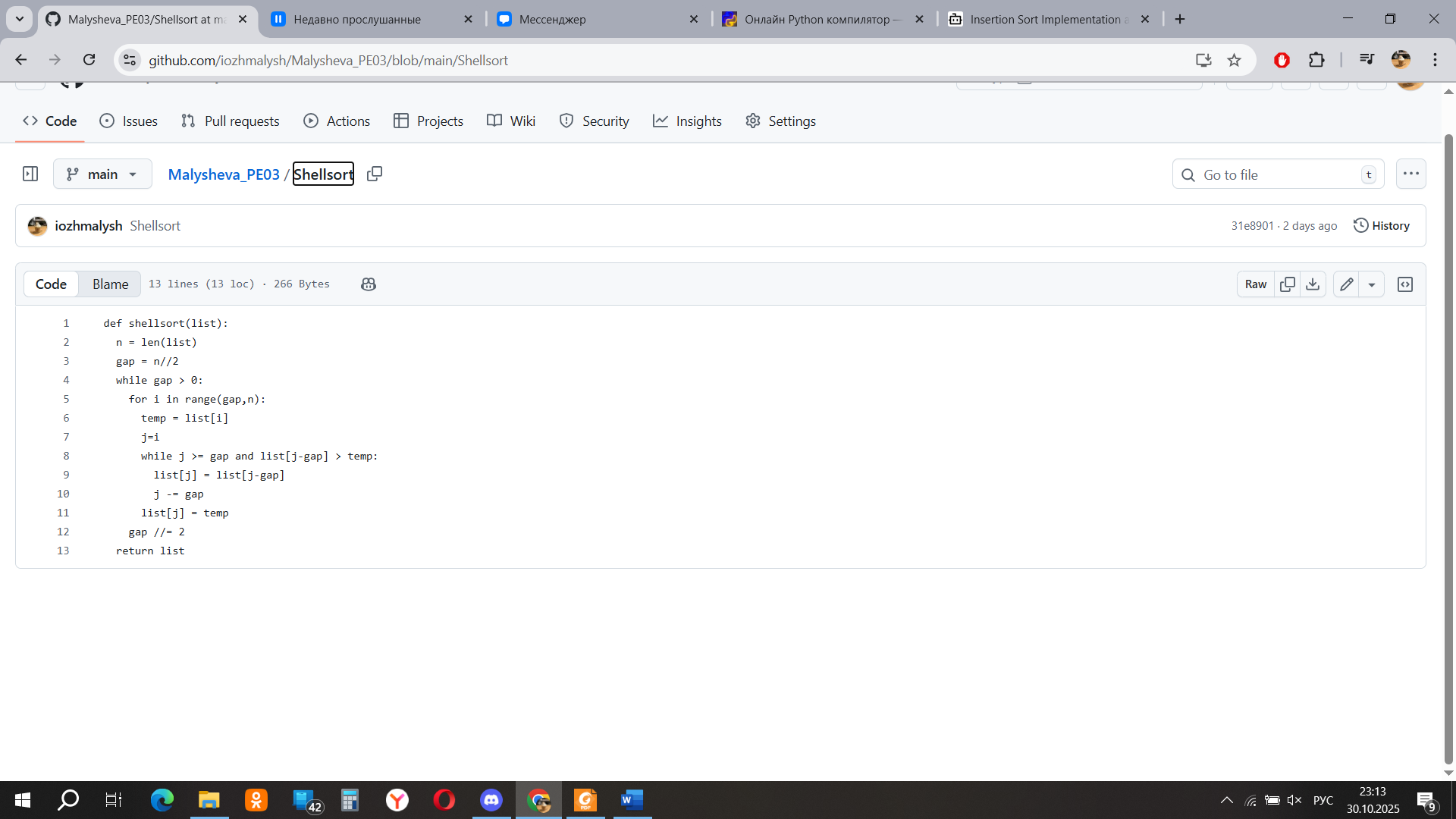


Рисунок 9 – Код сортировки Шелла

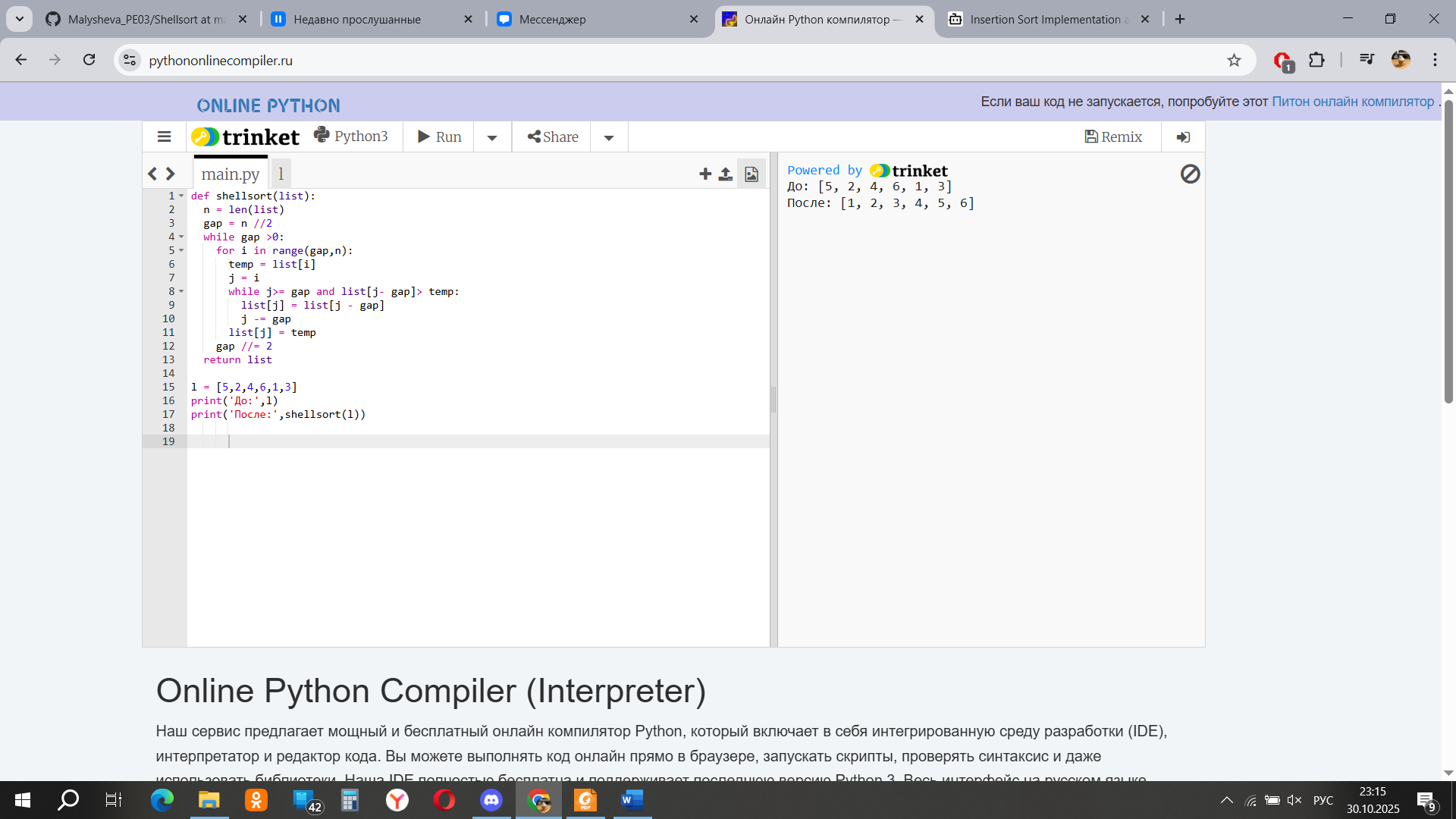


Рисунок 10 – Проверка кода сортировки Шелла

Временная сложность алгоритма:

* О (n log2 n) – при отсортированном массиве (лучший случай)
* O (n log n) - массив расположен в случайном порядке (средний случай)
* О (n2) - когда массив отсортирован в обратном порядке (худший случай)

1. **Быстрая сортировка (Quick Sort):**

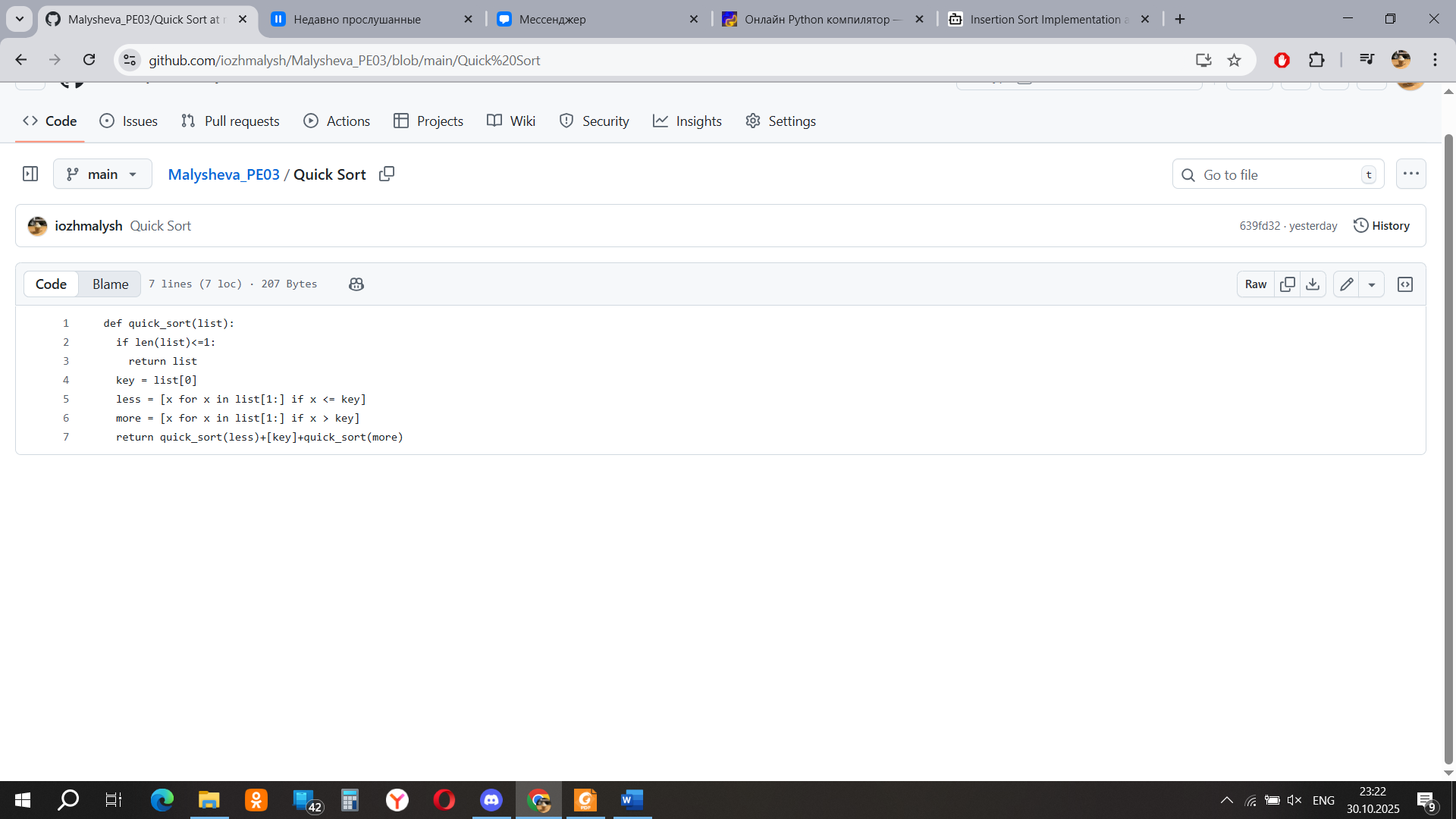


Рисунок 11 – Код быстрой сортировки

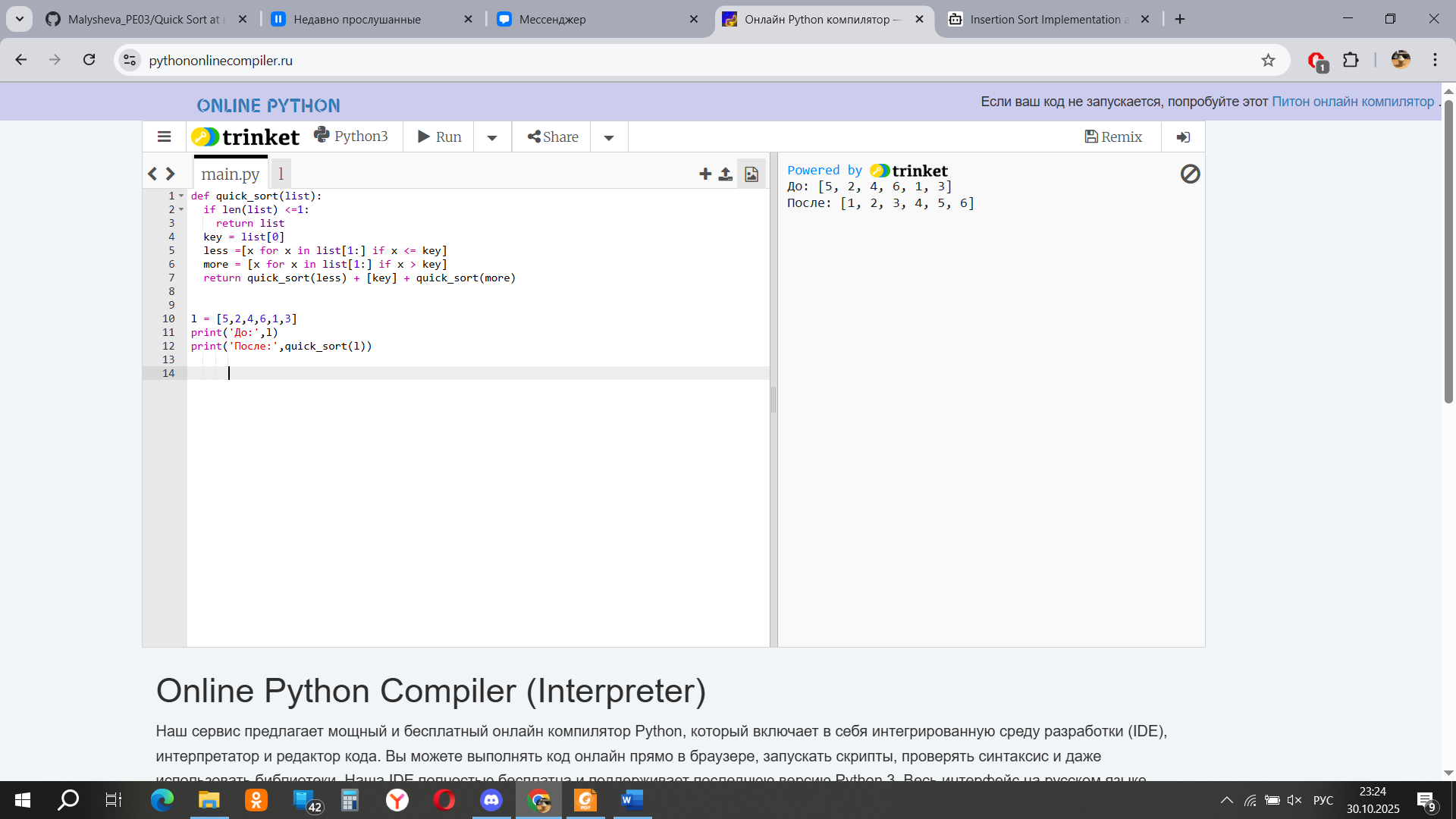


Рисунок 12 – Проверка кода быстрой сортировки

Временная сложность алгоритма:

* О (n log n) – при отсортированном массиве (лучший случай), массив расположен в случайном порядке (средний случай)
* О (n2) - когда массив отсортирован в обратном порядке (худший случай)

1. **Пирамидальная сортировка:**

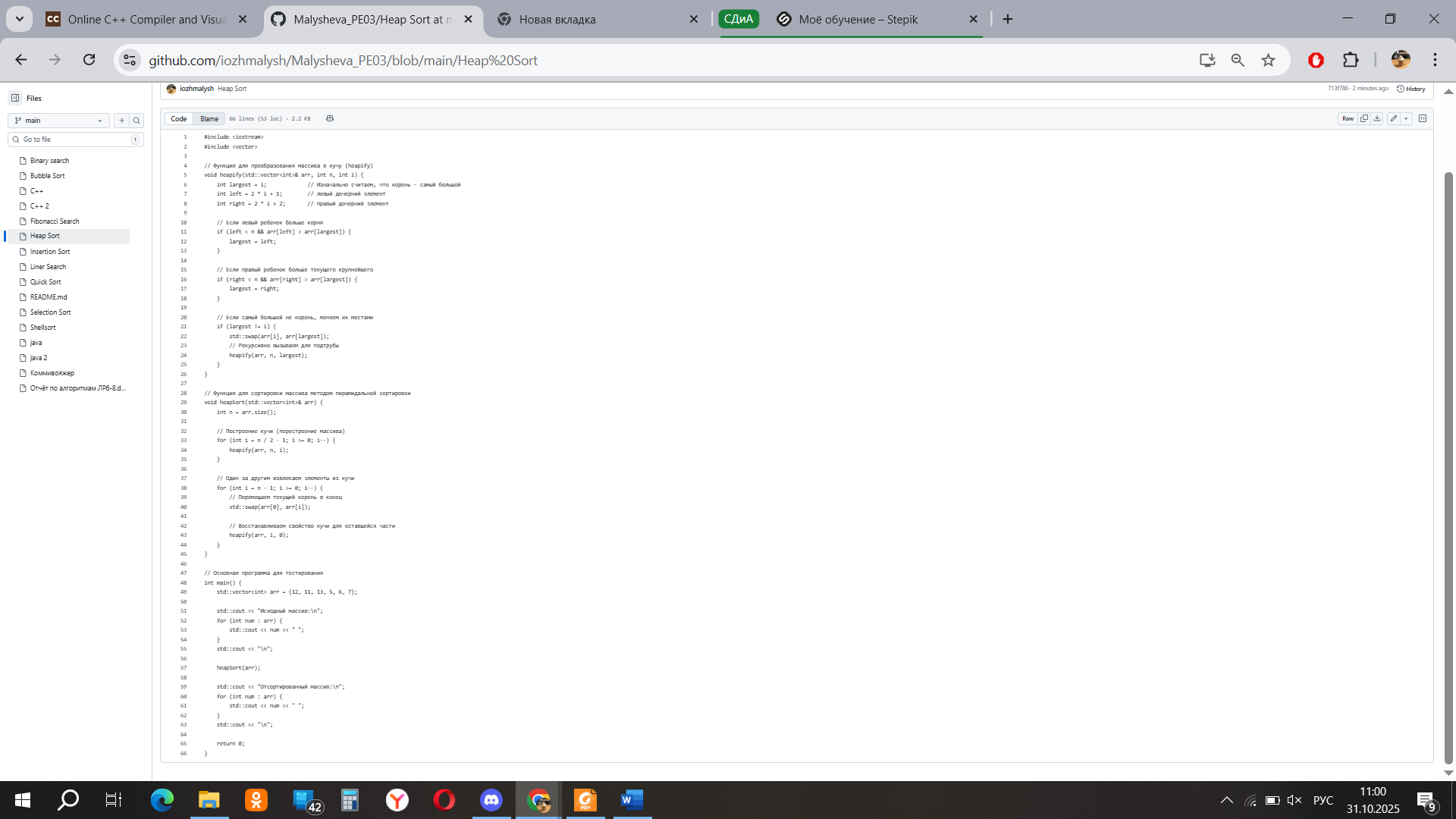
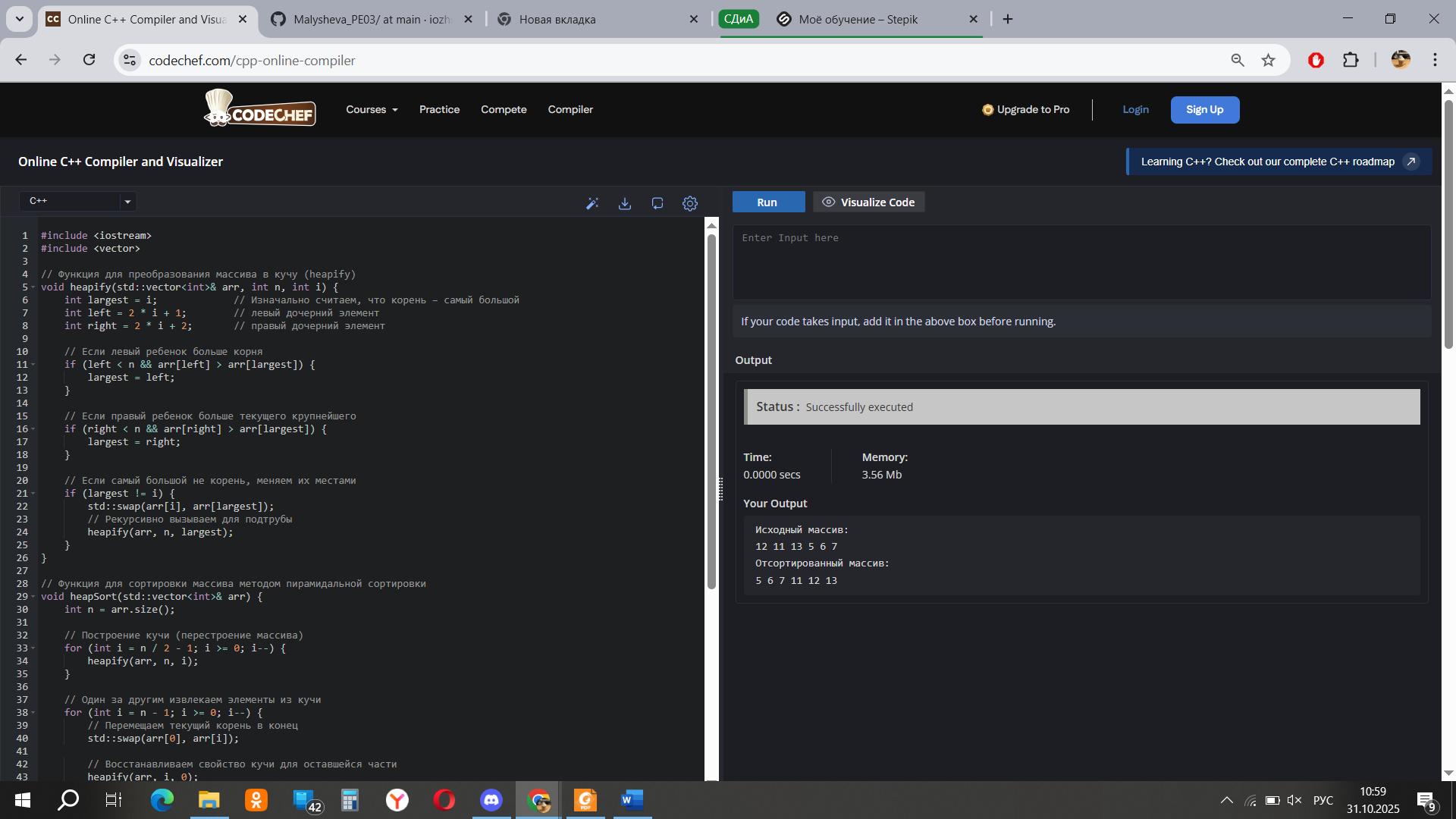


Рисунок 13– Код пирамидальной сортировки



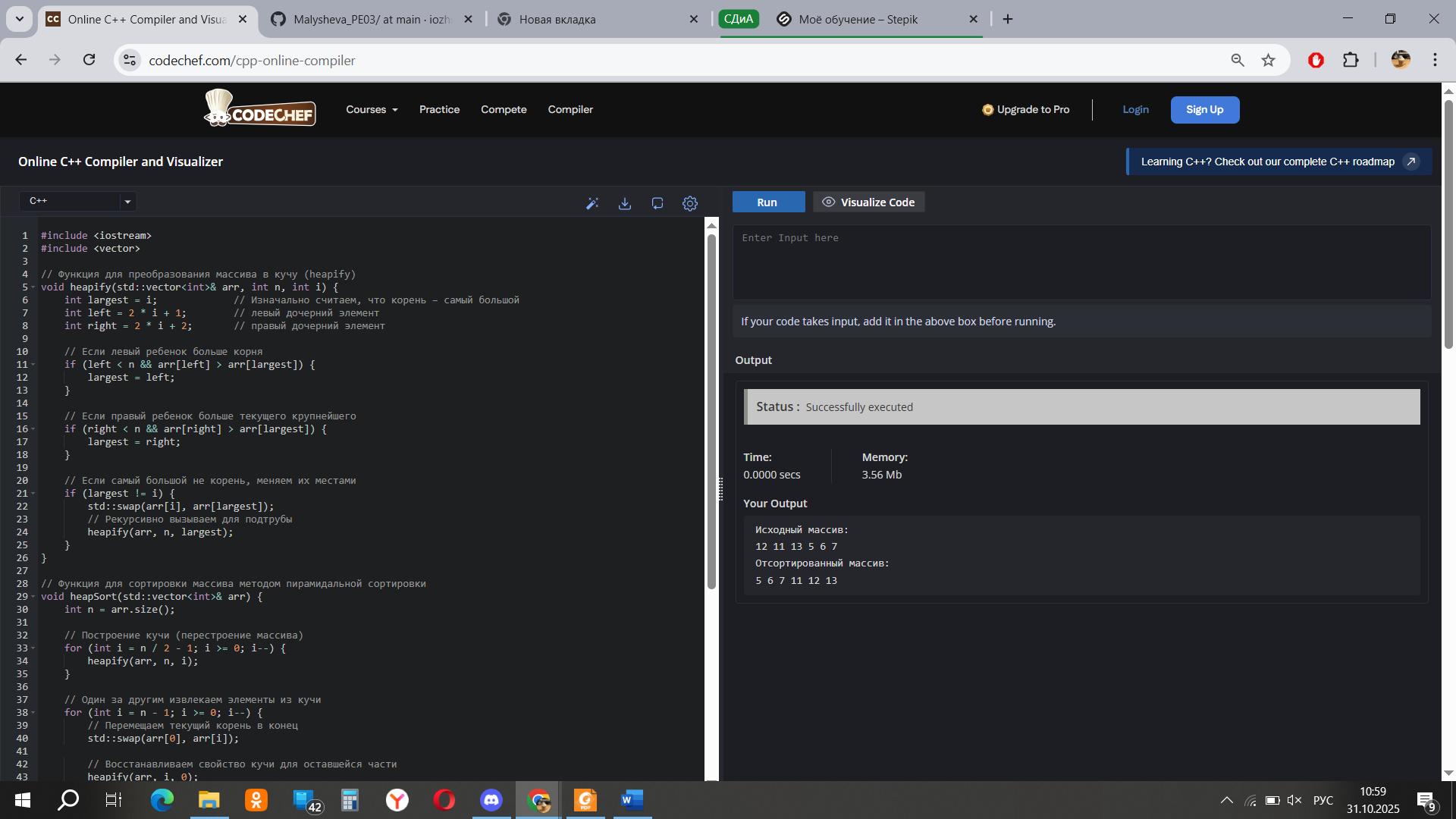


Рисунок 14– Проверка кода пирамидальной сортировки

Временная сложность алгоритма:

* O (n log n) - в худшем, среднем и лучшем случаях, где n -количество элементов в массиве

1. **Последовательный/линейный поиск**

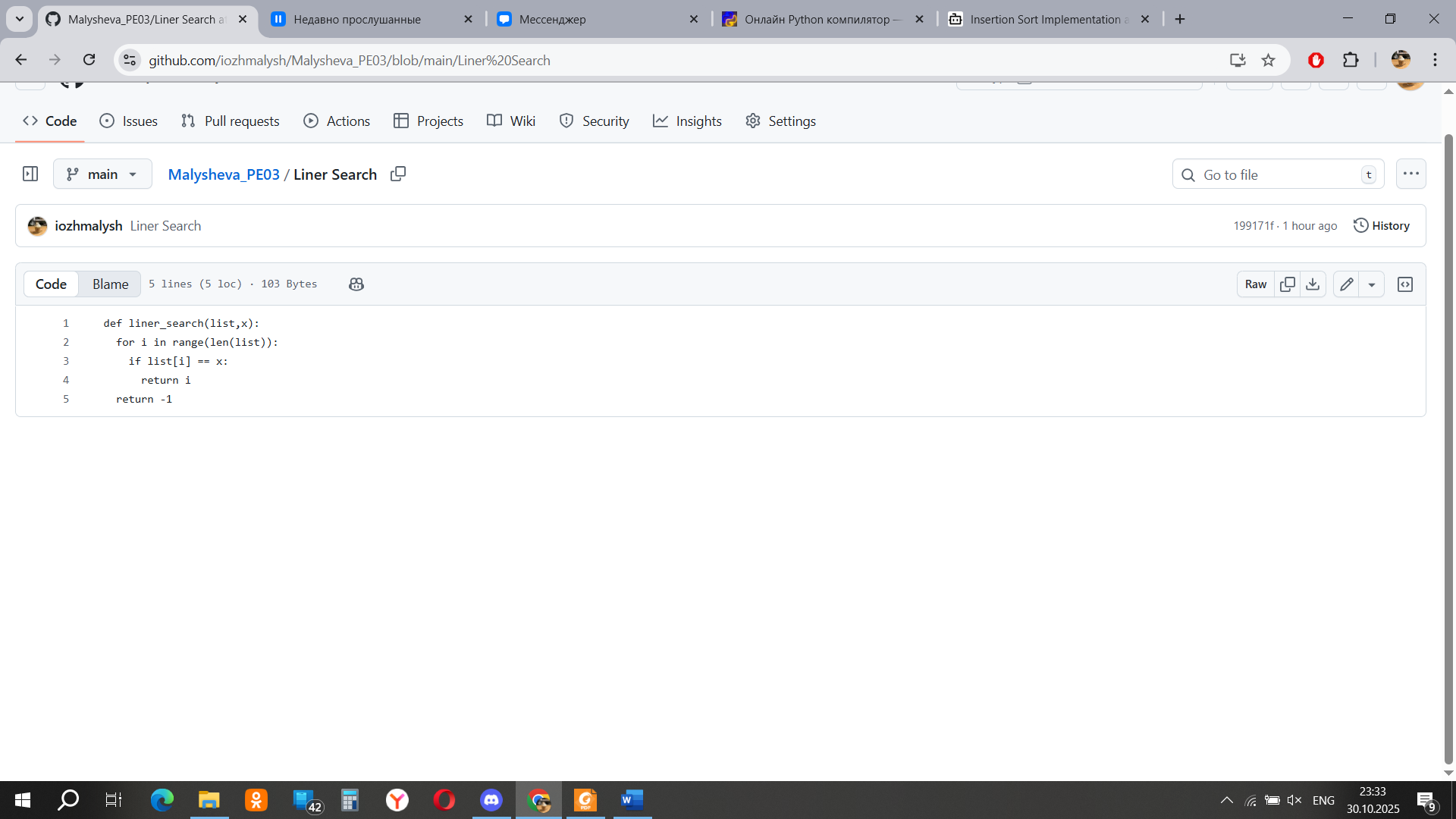


Рисунок 15 – Код линейного поиска

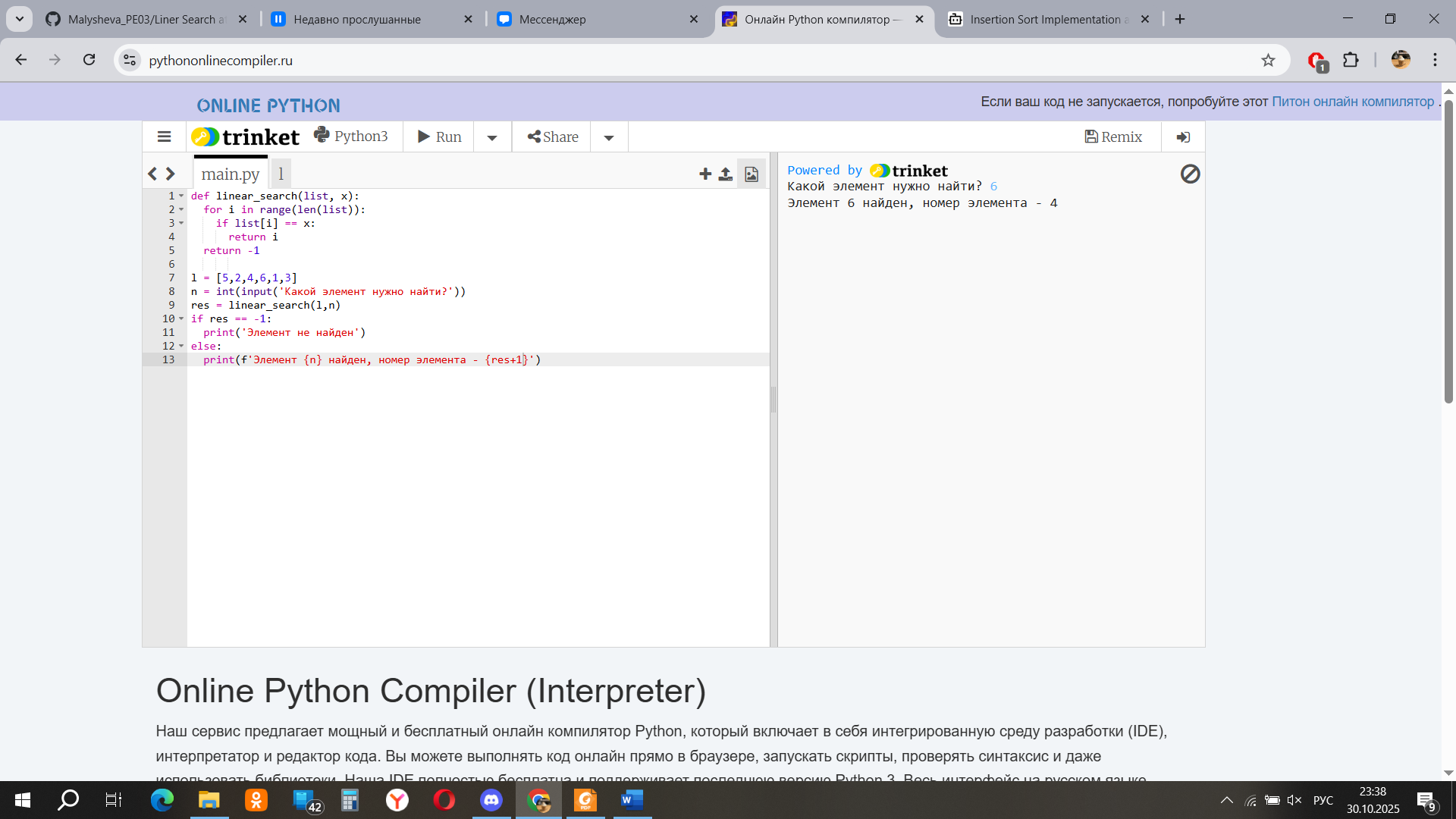


Рисунок 16 – Проверка кода линейного поиска

Временная сложность алгоритма:

* О(n) – в любом случае

1. **Бинарный поиск (Binary search):**

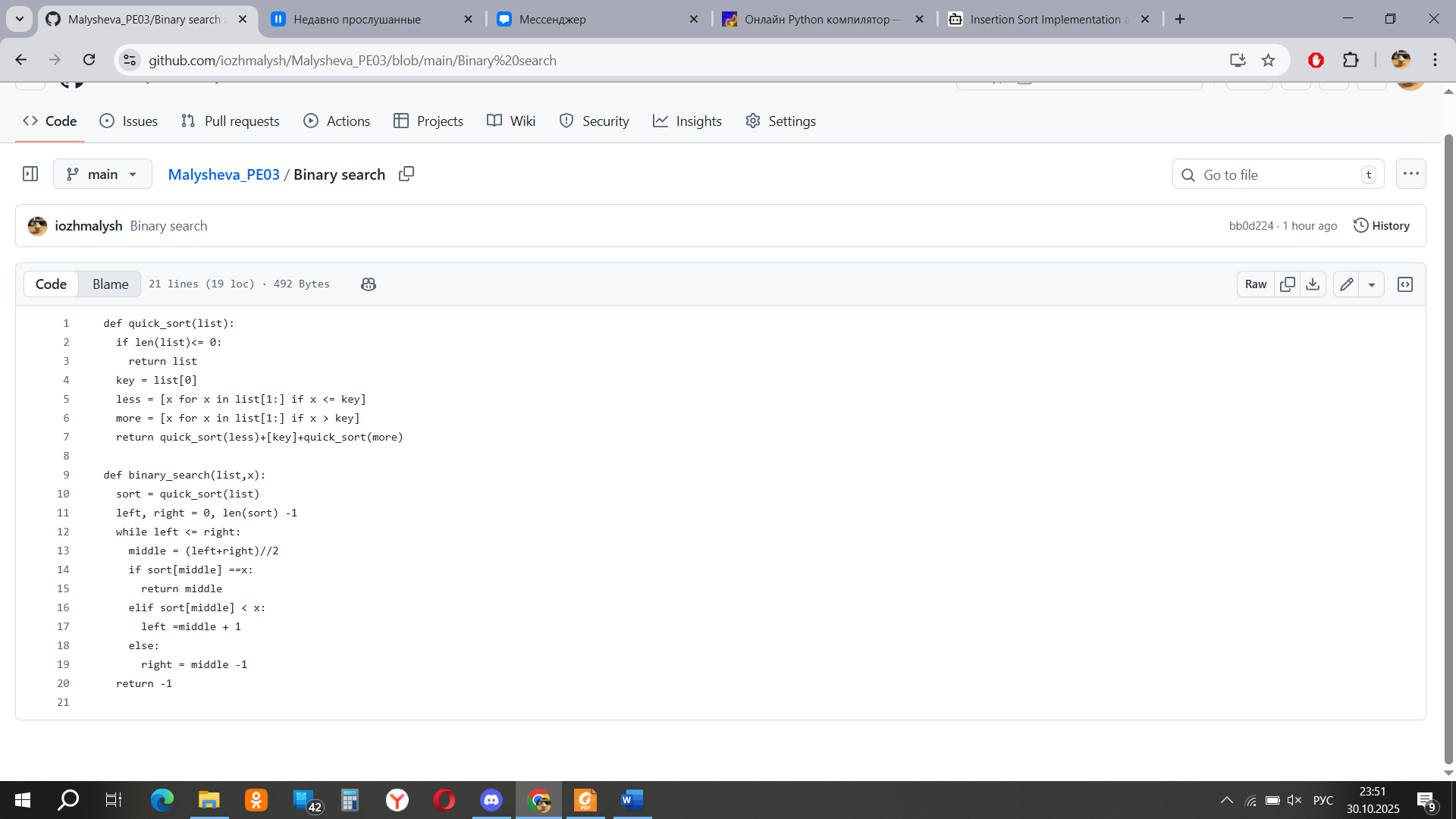


Рисунок 17 – Код бинарного поиска

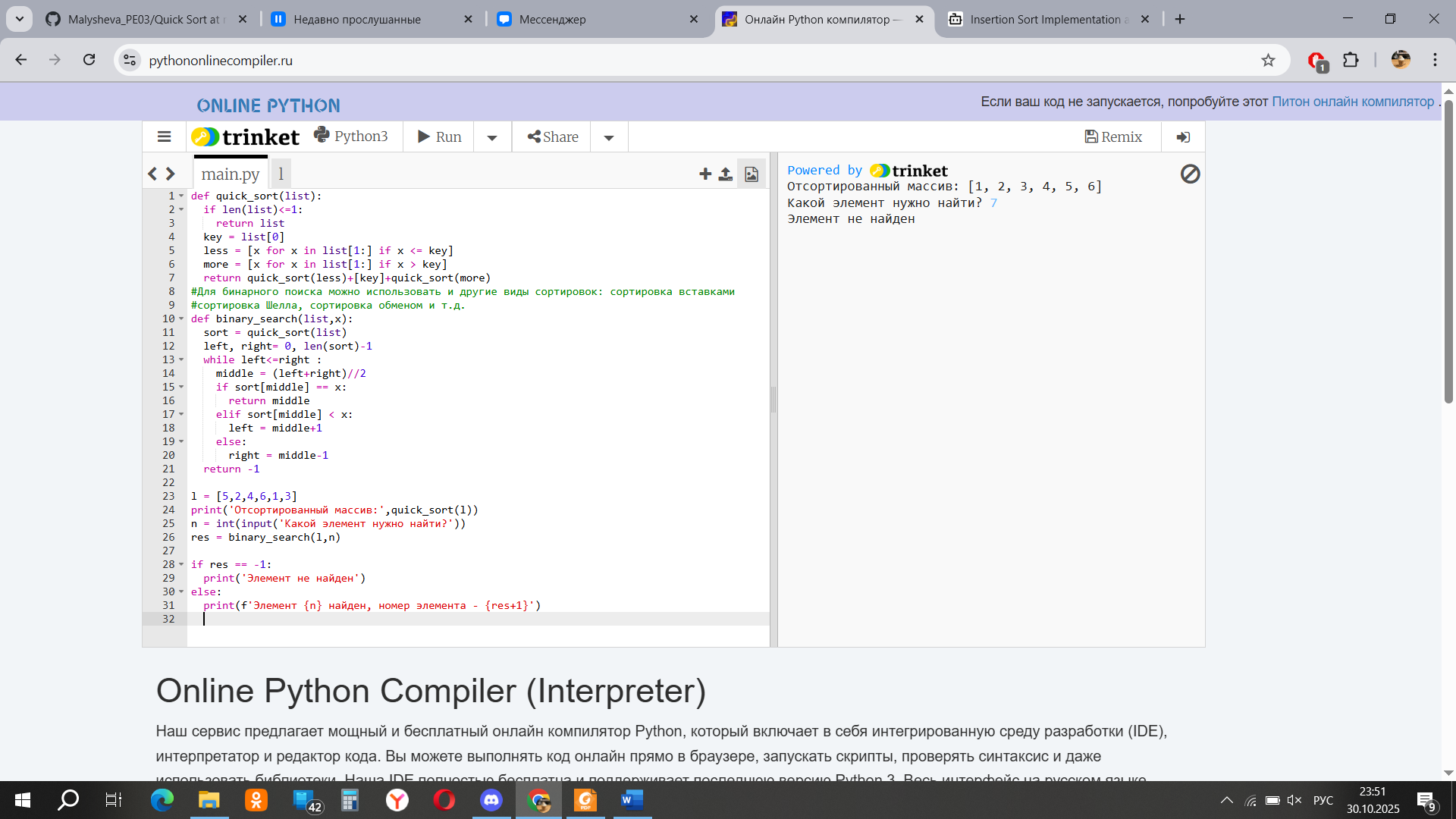


Рисунок 18 – Проверка кода бинарного поиска

Временная сложность алгоритма:

* О (log n) – когда вводят отсортированный массив
* О (n log n) - когда массив отсортировывают перед поиском (худший случай)

1. **Интерполирующий поиск**

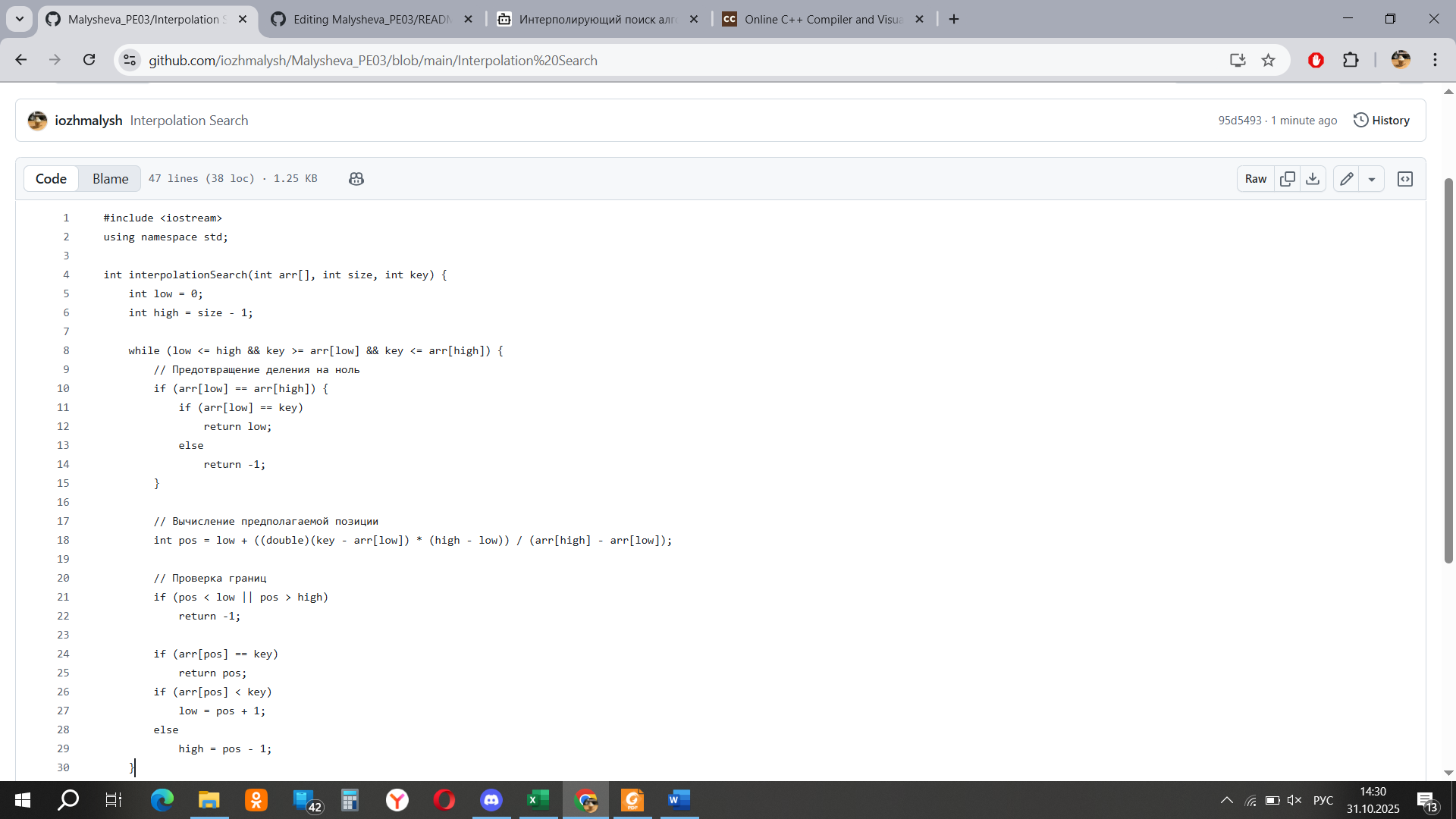
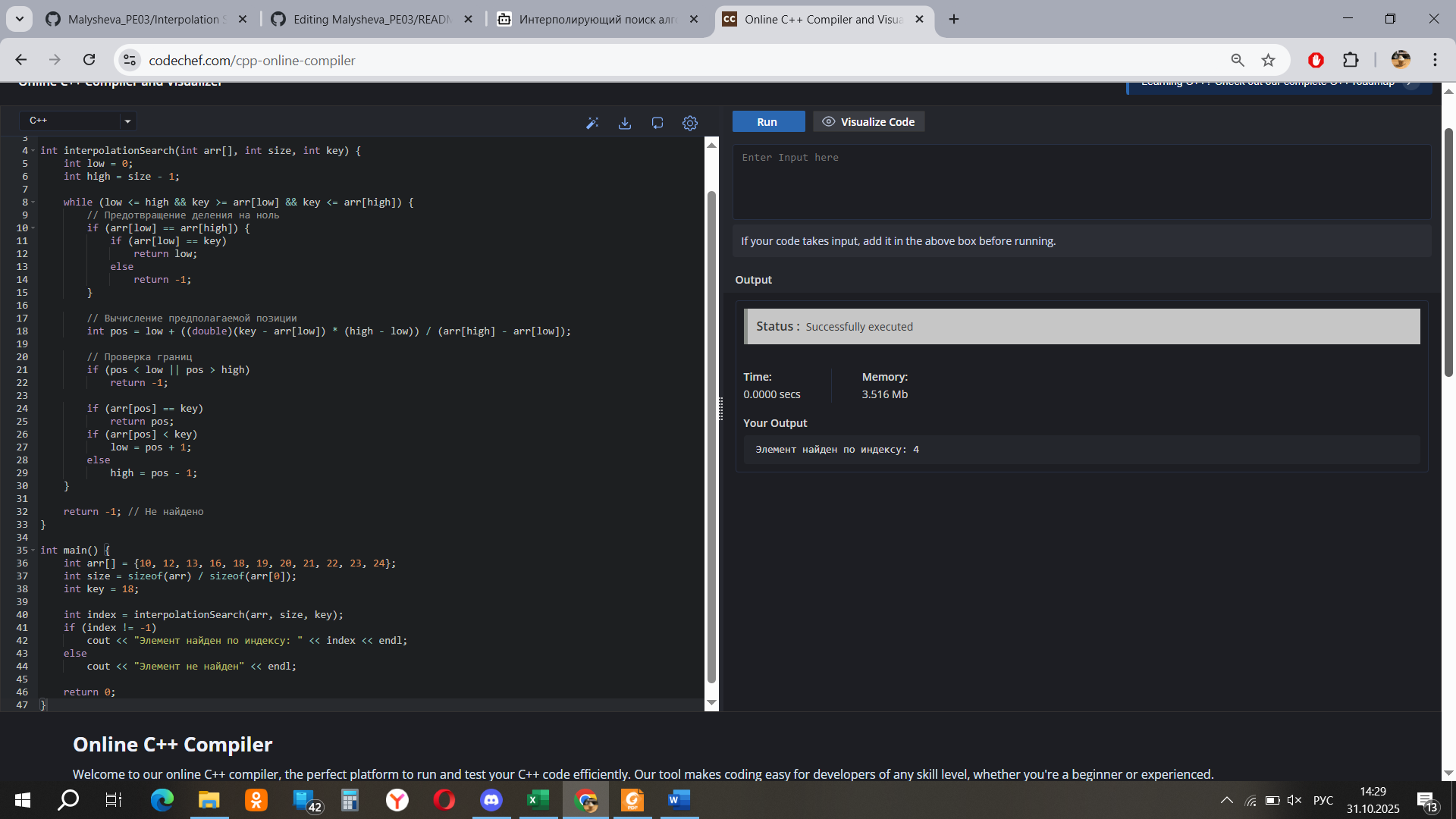


Рисунок 19 – Код интерполирующего поиска



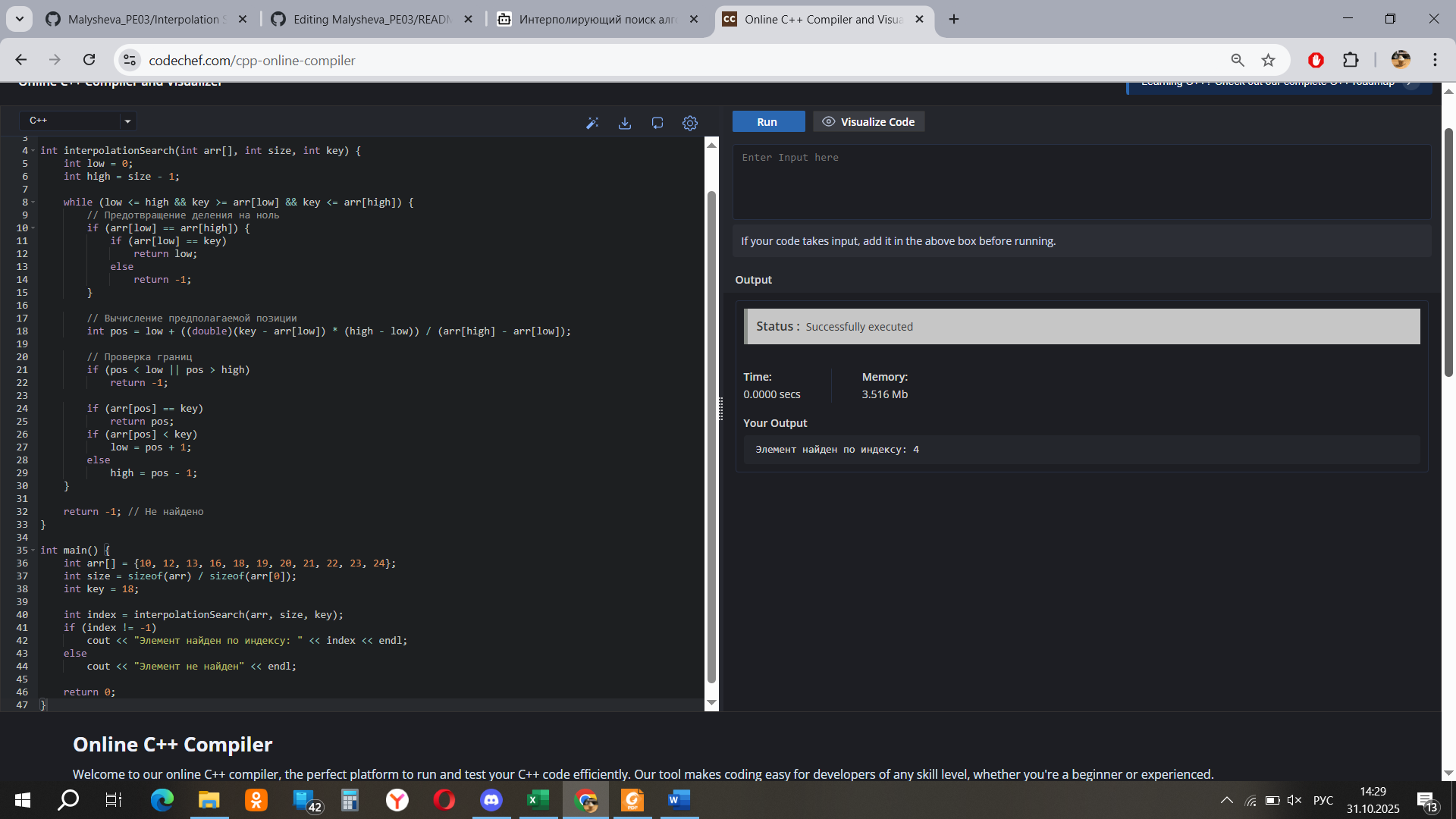


Рисунок 20 – Проверка кода интерполирующего кода

Временная сложность алгоритмов:

* O (log log n) — когда значения распределены равномерно;
* O ( n) — если значения распределены неравномерно (наихудший случай).

1. **Поиск по Фибоначчи**

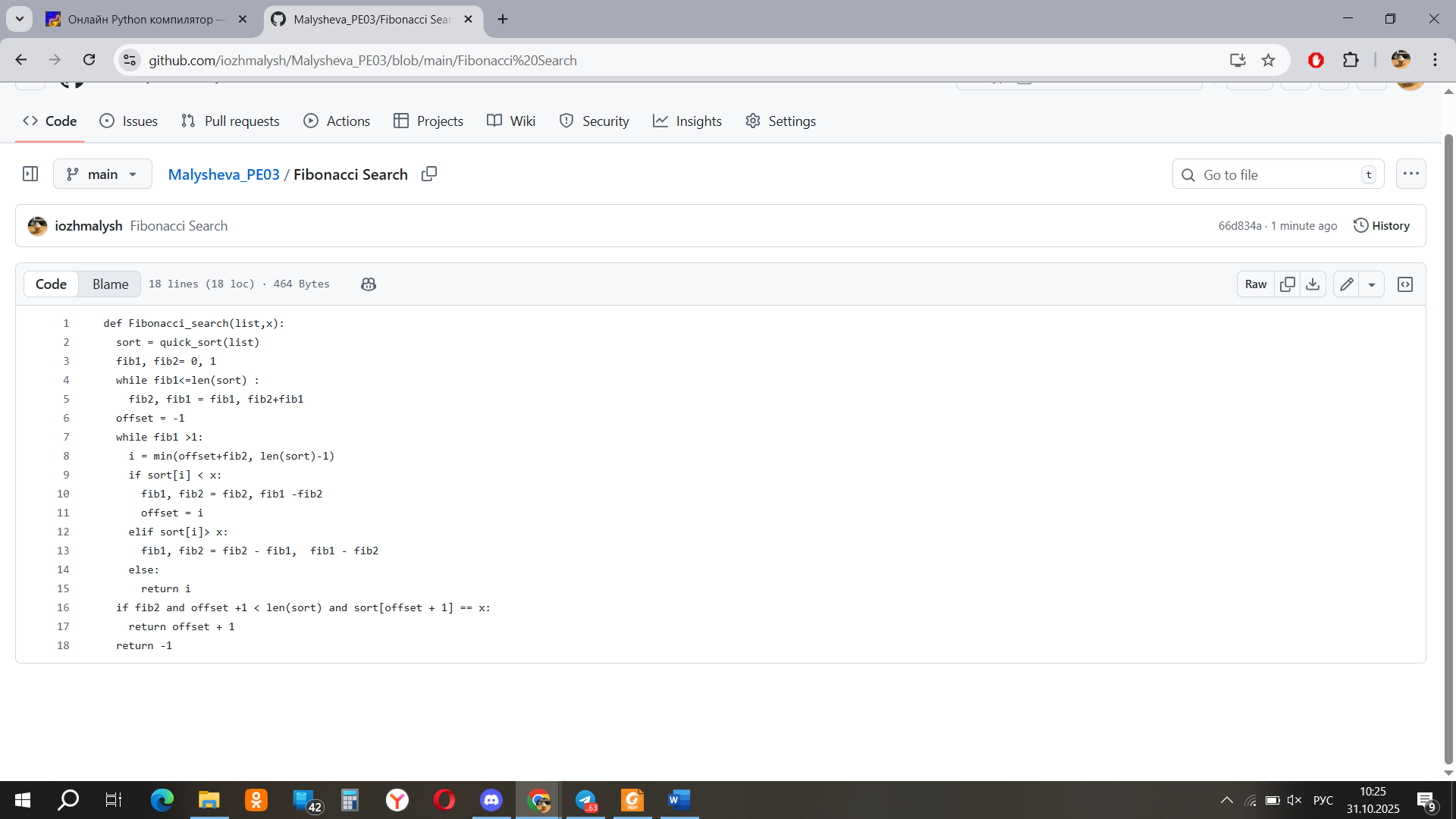


Рисунок 21 – Код поиска по Фибоначчи

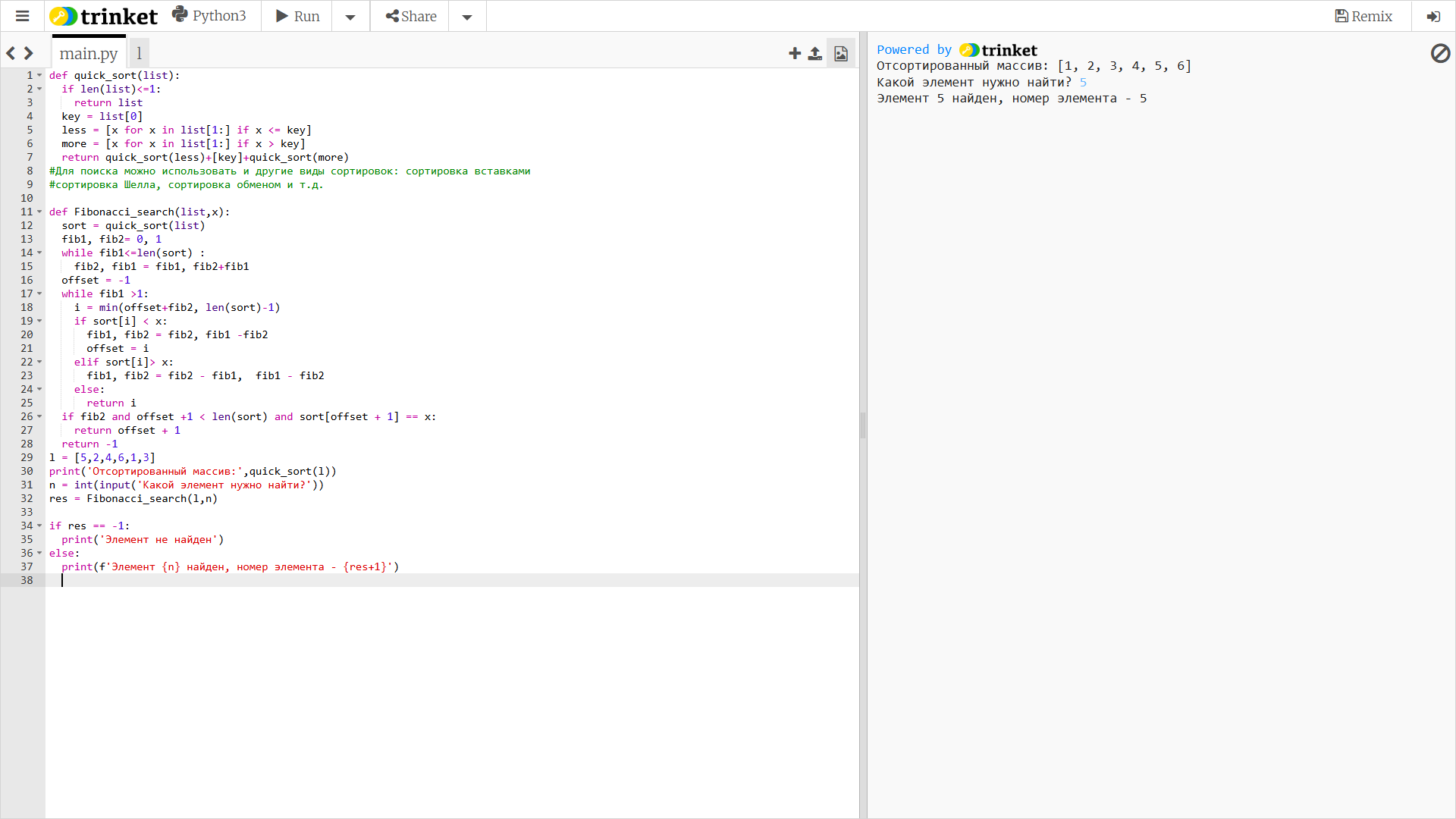


Рисунок 22 – Проверка кода поиска по Фибоначчи

Временная сложность алгоритма:

* О (log n) – когда вводят отсортированный массив
* О (n log n) - когда массив отсортировывают перед поиском (худший случай)