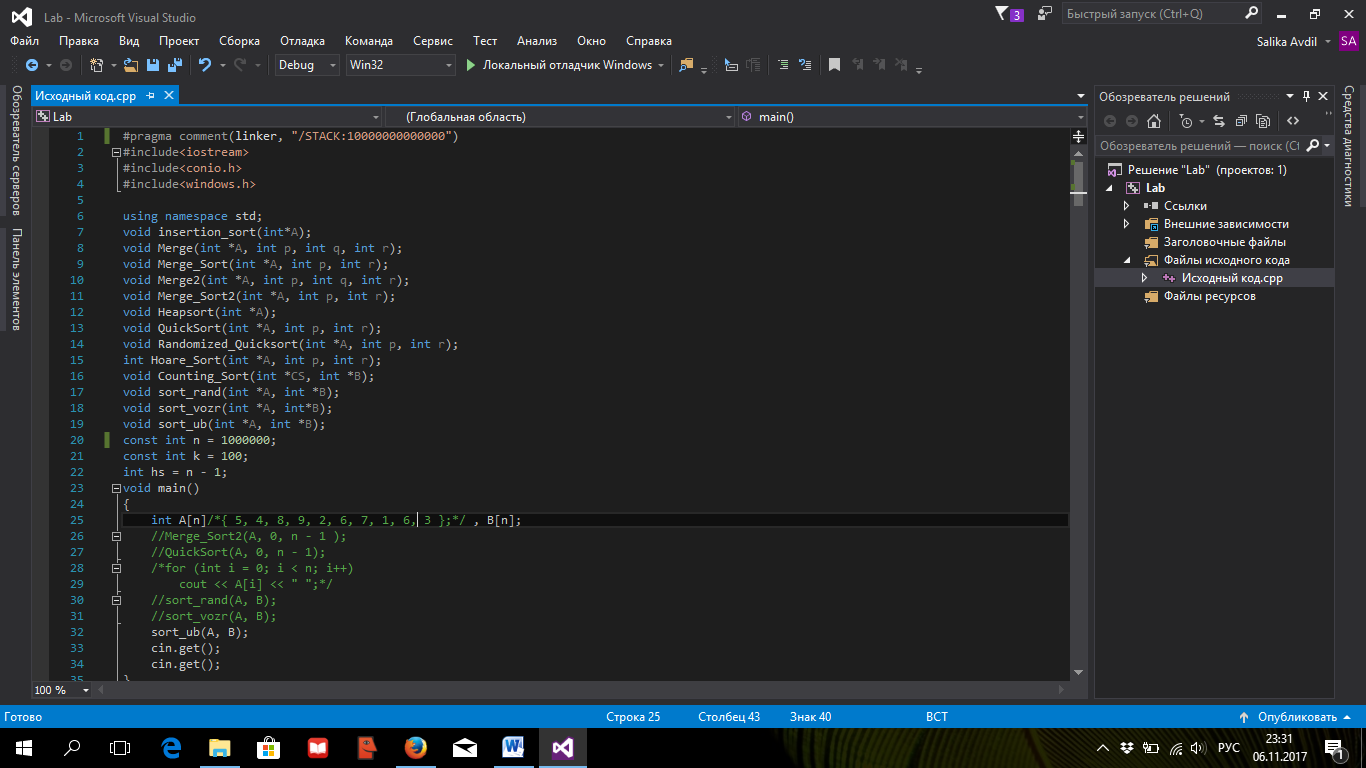
**ОТЧЁТ**

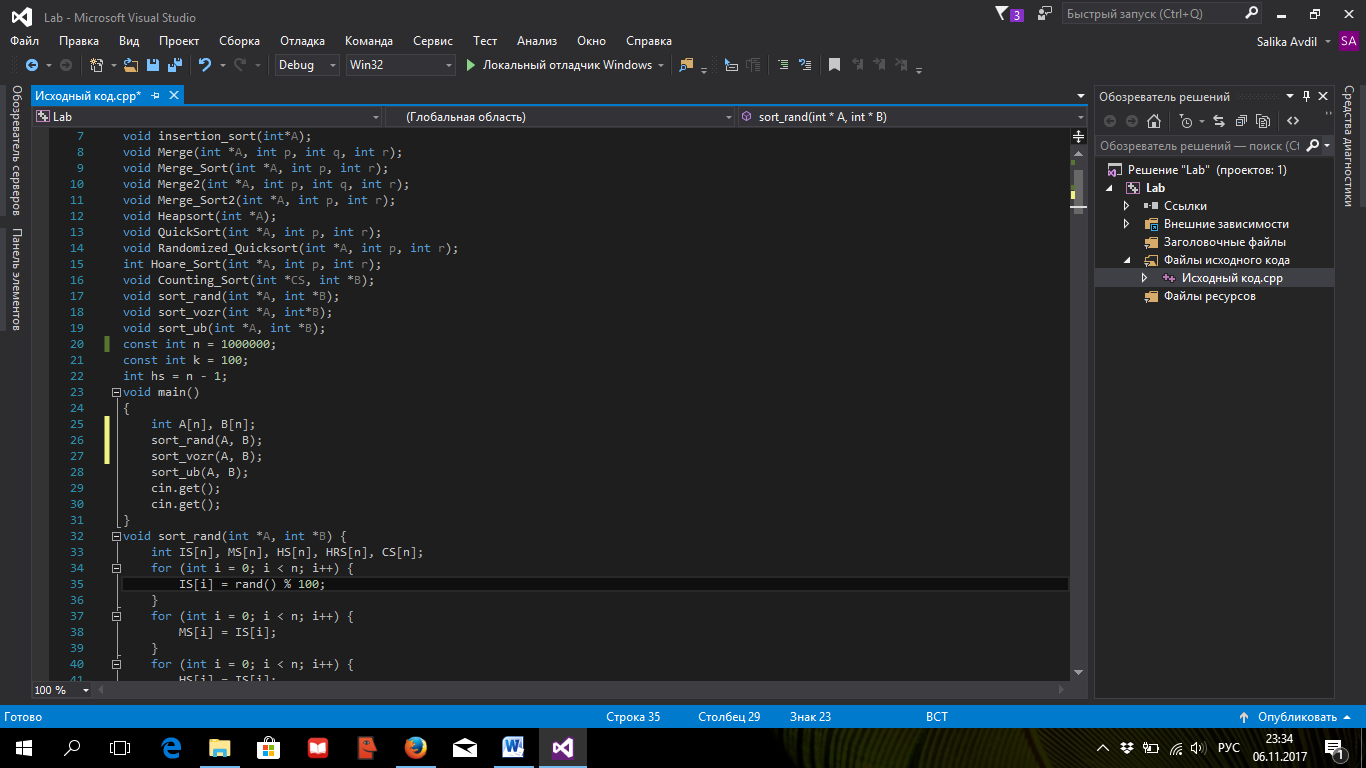
**Анализ алгоритмов сортировки**

**Цель:** провести сравнительный анализ алгоритмов сортировки за нелинейное и линейное время

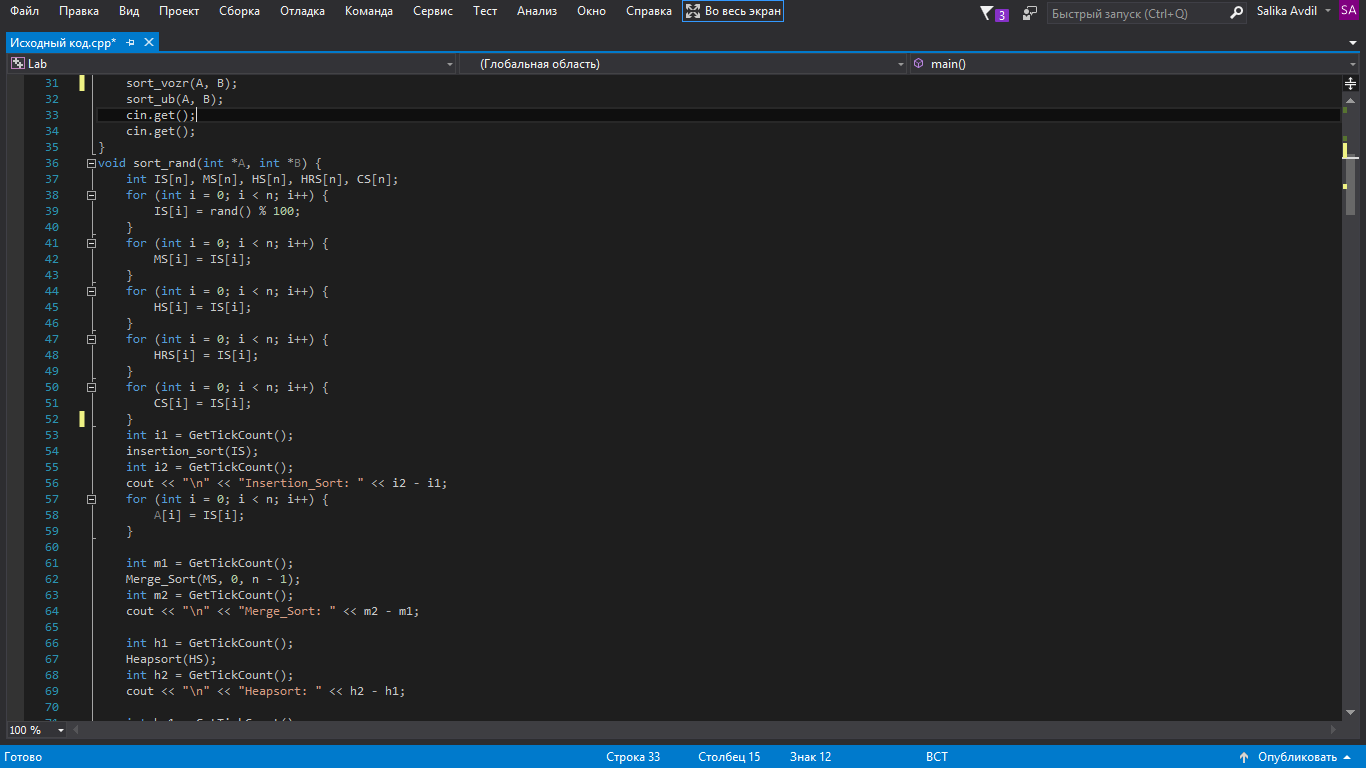
1. **Объявление функций:**

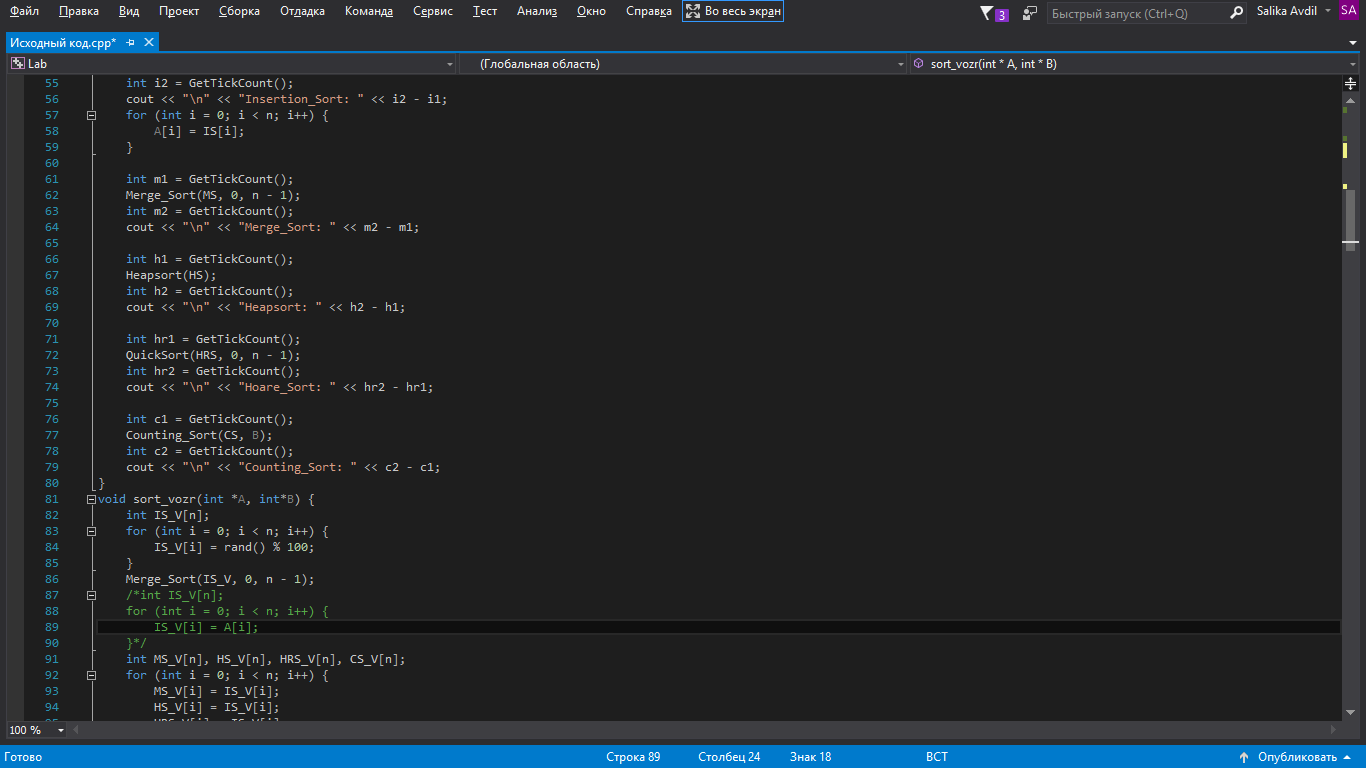


1. **Главная функция:**

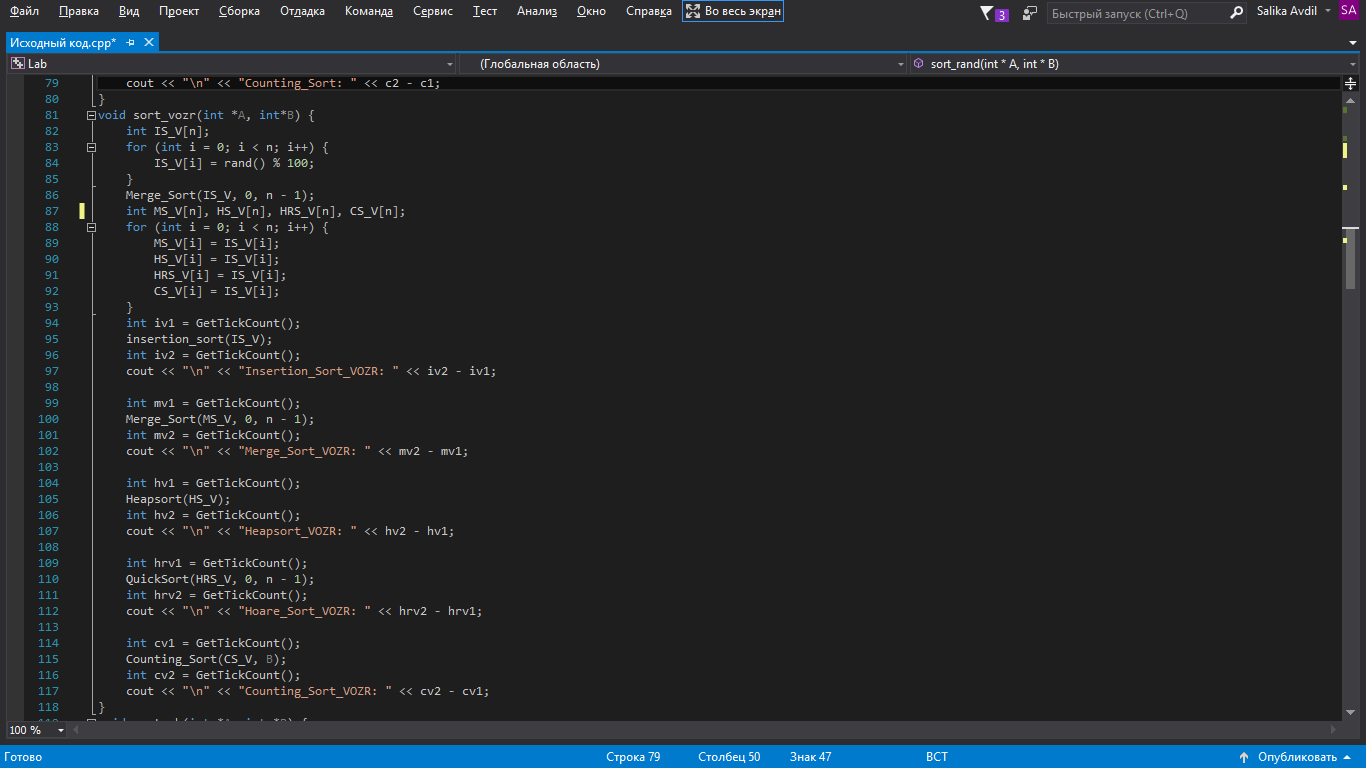


1. **Реализация сортировки случайных чисел:**

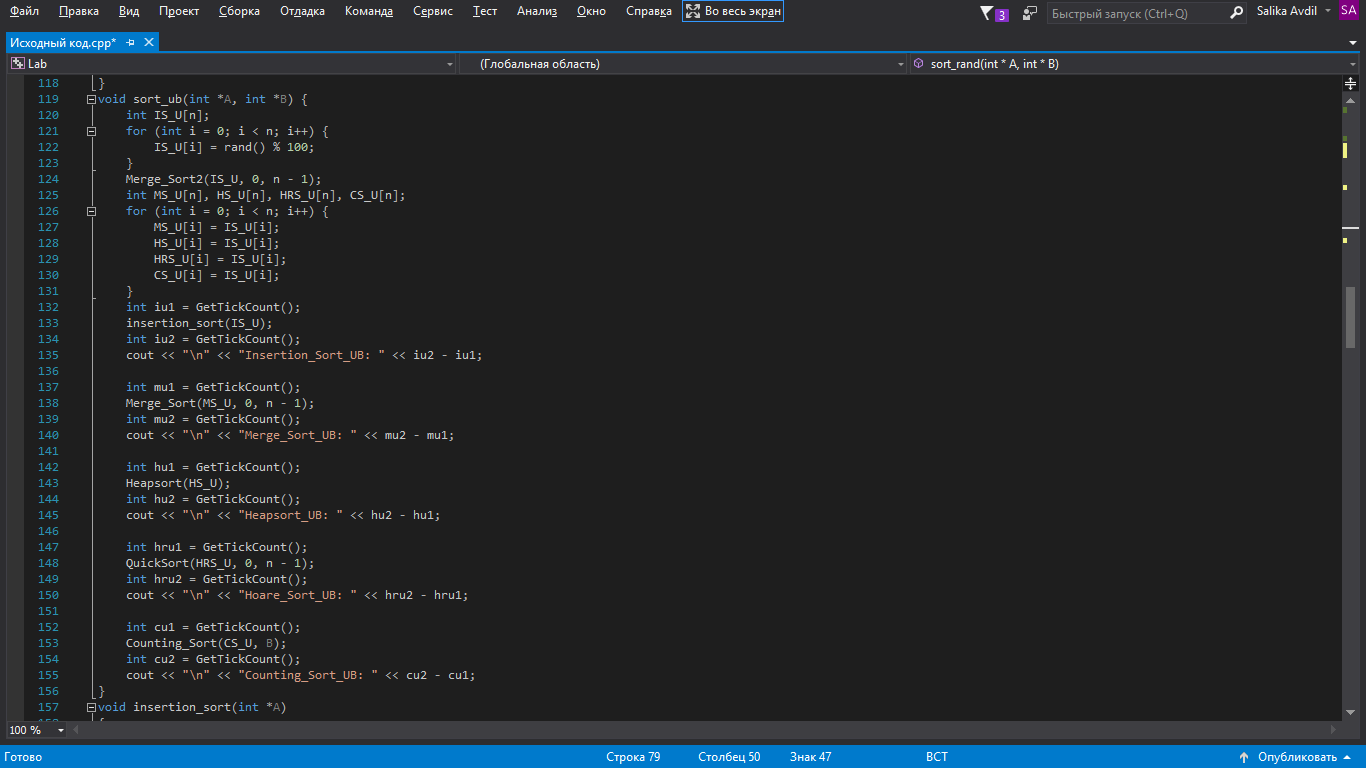




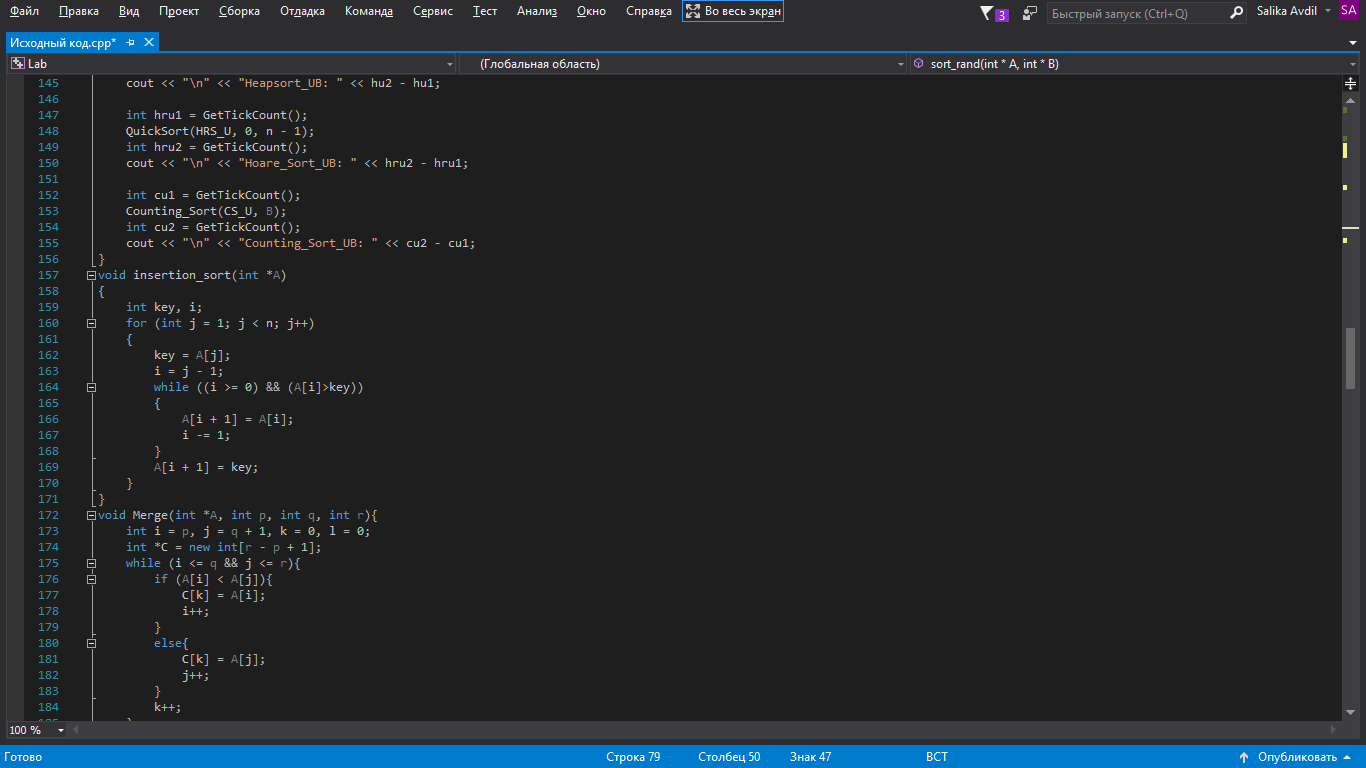
1. **Реализация сортировки отсортированных чисел:**



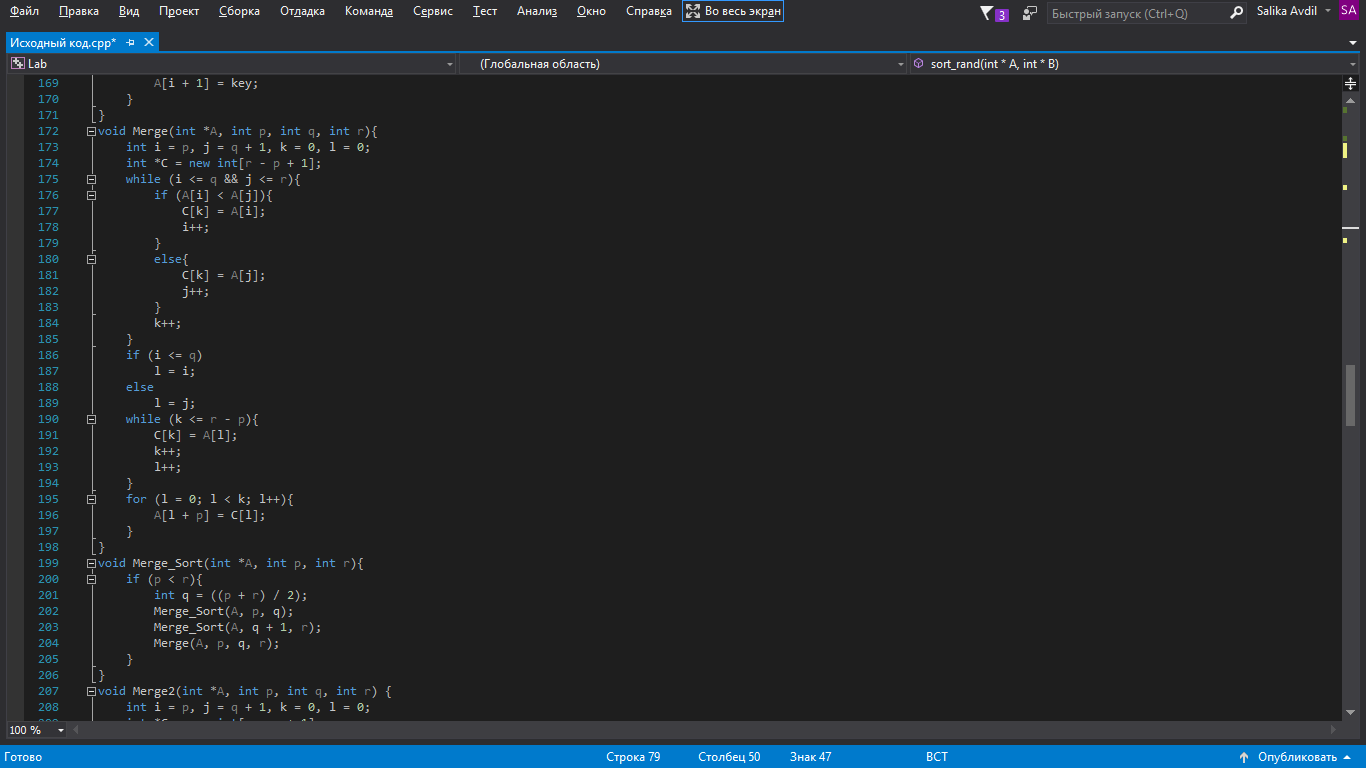
1. **Реализация сортировки отсортированных в обратном порядке чисел:**



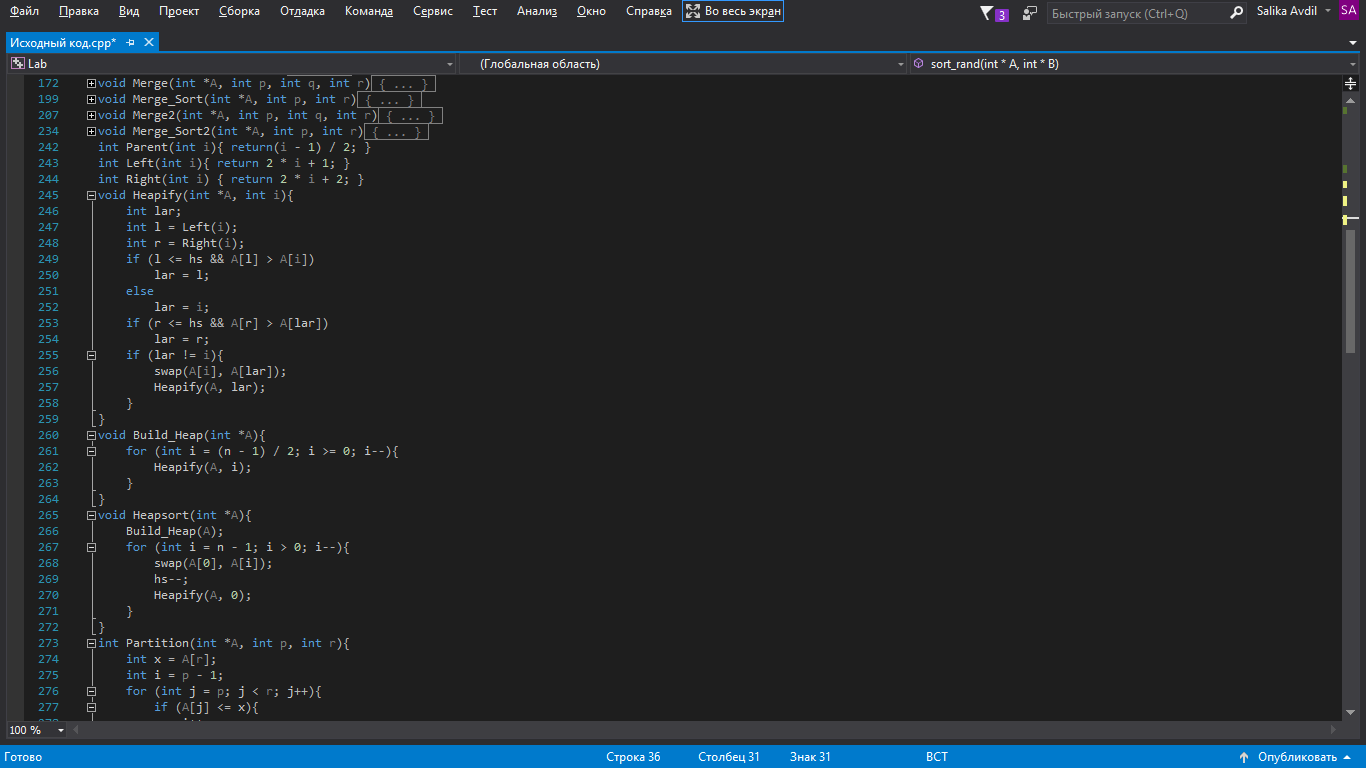
1. **Реализация Insertion Sort на С++:**



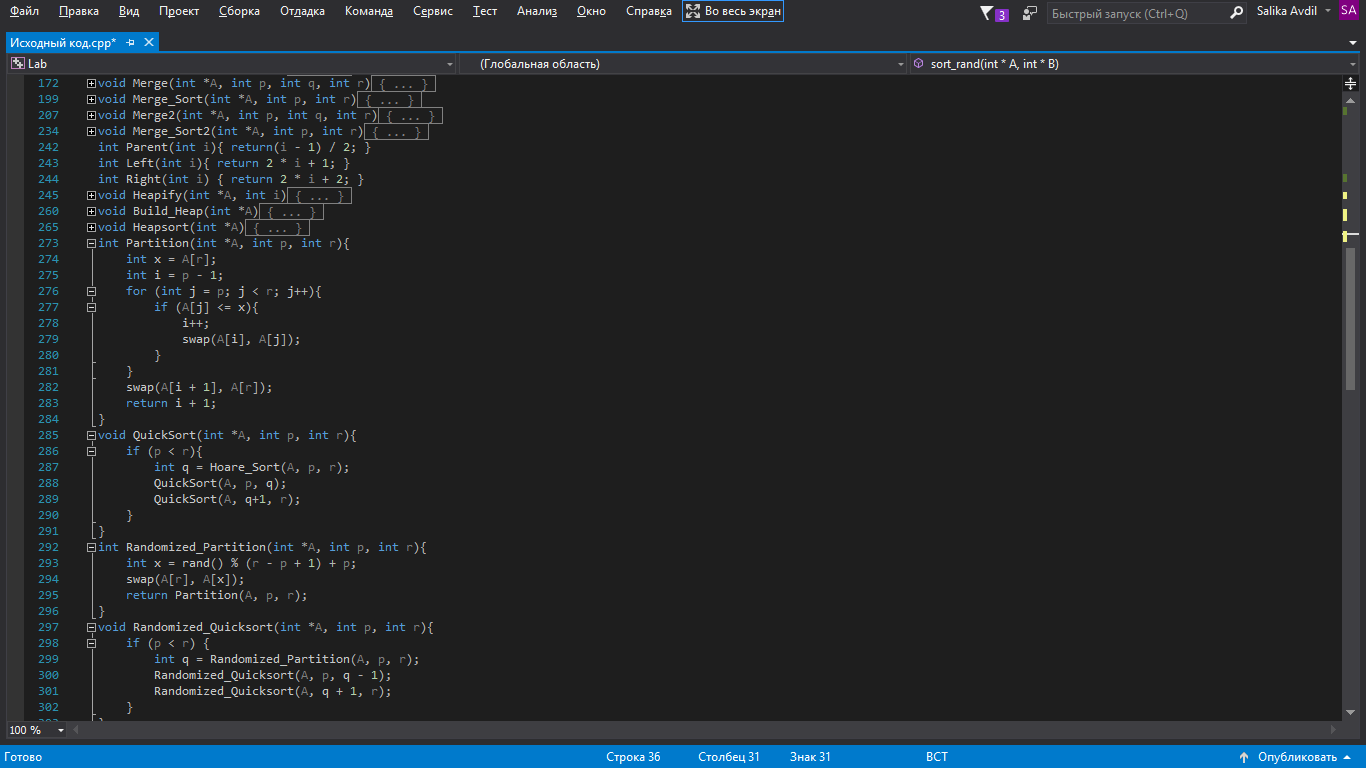
1. **Реализация Merge Sort на С++:**



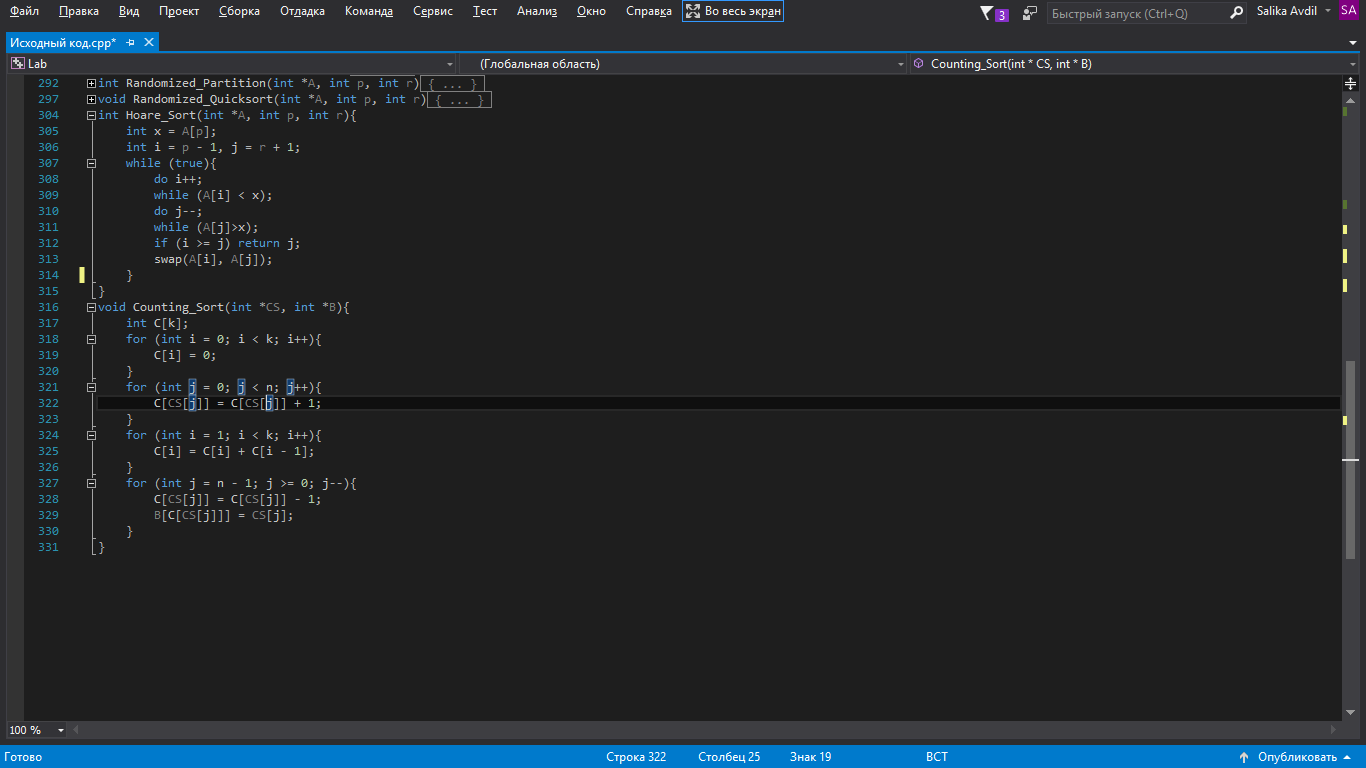
1. **Реализация Heap Sort на С++:**



1. **Реализация Quick Sort на С++:**



1. **Реализация Hoare Sort на C++:**



**Асимптотическая оценка работы алгоритмов сортировки**

|  |  |  |  |
| --- | --- | --- | --- |
| **Формулы** | **Случайные числа** | **Отсортированный массив** | **Массив, отсорт. в обратном порядке** |
| Insertion Sort | 𝑂(𝑛2) | 𝑂(𝑛) | 𝑂(𝑛2) |
| Merge Sort | 𝑂(𝑛log𝑛) | 𝑂(𝑛log𝑛) | 𝑂(𝑛log𝑛) |
| Heapsort | 𝑂(𝑛log𝑛) | 𝑂(𝑛log𝑛) | 𝑂(𝑛log𝑛) |
| Quicksort  (Hoare Sort) | 𝑂(𝑛log𝑛) | 𝑂(𝑛2) | 𝑂(𝑛2) |

**Время сортировки массива случайных чисел**

|  |  |  |  |
| --- | --- | --- | --- |
| **Время работы** | **n=104** | **n=105** | **n=106** |
| Insertion Sort | 1077 | 26551 | 2495595 |
| Merge Sort | 31 | 110 | 1232 |
| Heapsort | 141 | 592 | 7910 |
| Quicksort  (Hoare Sort) | 47 | 156 | 2152 |

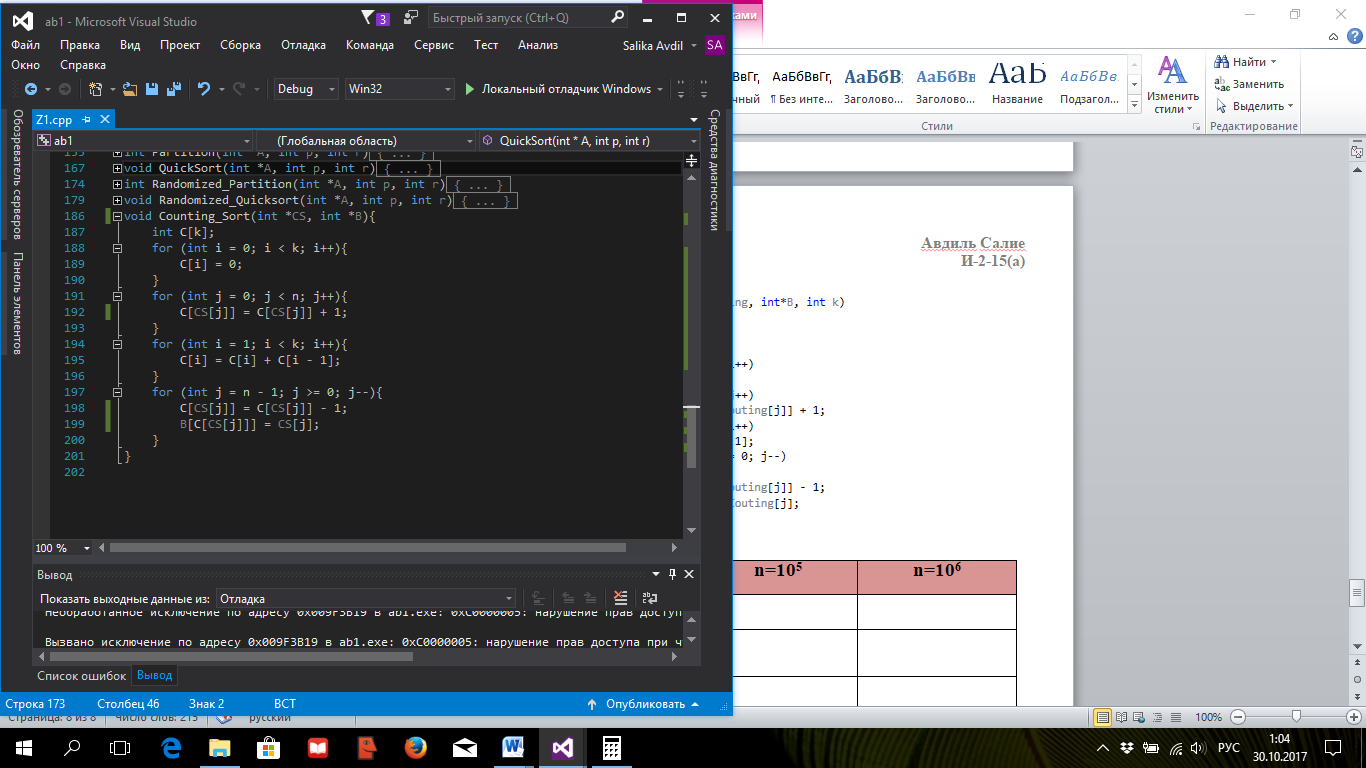
**Время сортировки массива, в котором числа упорядочены по возрастанию**

|  |  |  |  |
| --- | --- | --- | --- |
| **Время работы** | **n=104** | **n=105** | **n=106** |
| Insertion Sort | 0 | 0 | 16 |
| Merge Sort | 31 | 125 | 1107 |
| Heapsort | 468 | 749 | 6240 |
| Quicksort  (Hoare Sort) | 109 | 546 | 5554 |

**Время сортировки массива, в котором числа упорядочены по убыванию**

|  |  |  |  |
| --- | --- | --- | --- |
| **Время работы** | **n=104** | **n=105** | **n=106** |
| Insertion Sort | 1529 | 3500 | 3477312 |
| Merge Sort | 15 | 94 | 1047 |
| Heapsort | 125 | 563 | 6031 |
| Quicksort  (Hoare Sort) | 47 | 234 | 2594 |

1. **Линейная сортировка Counting sort**



|  |  |  |  |
| --- | --- | --- | --- |
| **Время работы** | **n=104** | **n=106** | **n=108** |
| Случайные числа | 0 | 32 | 908 |
| Отсортированный массив | 0 | 31 | 867 |
| Массив, отсортированный в обратном порядке | 0 | 31 | 886 |

**Вывод:** В ходе реализации различных алгоритмов и вычисления времени их работы, можно сделать вывод, что в массиве случайных чисел и в отсортированном в обратном порядке массиве лучшей сортировкой является Merge Sort (сортировка слиянием), а в отсортированном массиве по возрастанию – Insertion Sort (сортировка вставками).

Наихудшем временем работы Insertion Sort будет, если числа в массиве будут упорядочены по убыванию. Наилучшем – если числа отсортированы по возрастанию.

Merge Sort и Hoare Sort работают стабильно независимо от того как расположены числа в массиве.

Пирамидальная сортировка (HeapSort) работает стабильно, если мы рассматриваем массив случайных чисел и массив, в котором числа упорядочены по убыванию, а в массиве, в котором числа упорядочены по возрастанию HeapSort работает немного дольше.

Линейная сортировка является самым быстрым алгоритмом из всех рассматриваемых. Независимо от расположения чисел в массиве и размера этого массива, линейная сортировка всегда работает стабильно и быстро.

*Заметка:* при возникновении ошибки Stack Overflow (когда сортируем большие массивы), необходимо увеличить стек, с помощью:

***#pragma comment (linker, “/STACK:10000000000000”).***