# Experiment 5

WAP to perform comparative analysis of iterative soring algorithm (Quick , Randomized Sort)



Fig: 1 (Quick Sort)

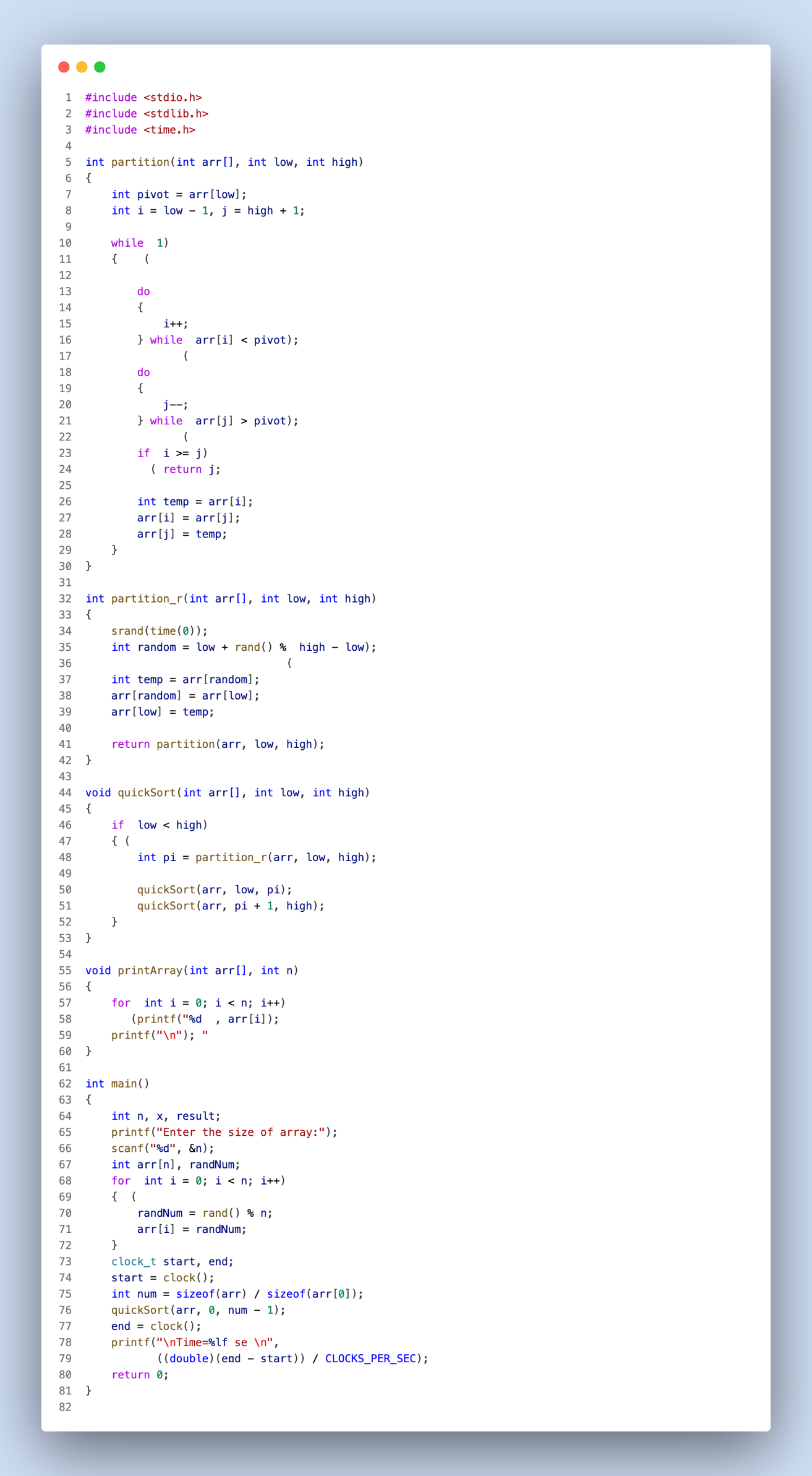
****

Fig: 2 (Randomized Quick Sort)

**Result Analysis and Discussion**

## This experiment is conducted using following specifications. The algorithm is implemented using C language (clang-1400.0.29.202). During this test all the apps were closed to improve the results of the experiment.

## 

Fig: 4 (PC used in experiment)

In this experiment the comparative analysis of iterative sort algorithms has been implemented and executed for different value of n. During this experiment for different value of n the time taken by the algorithm has been measured and tabulated as shown in table below.

|  |  |  |
| --- | --- | --- |
| Arr size (in lakhs) | Time sec  (Quick sort) | Time sec  (Randomized Quick sort) |
| 5 | 0.089221 | 0.199315 |
| 10 | 0.185897 | 0.407216 |
| 15 | 0.283526 | 0.613265 |
| 20 | 0.389866 | 0.821665 |

**Comparative Analysis:**

**Performance:**

In terms of performance, Quick Sort is consistently faster than Randomized Quick Sort for all tested array sizes. This suggests that in this specific experiment, the non-randomized version of Quick Sort outperforms the randomized version.

**Efficiency:**

Quick Sort is more efficient in terms of execution time compared to Randomized Quick Sort. It is able to sort the arrays in less time.

**Growth Rate:**

Both algorithms exhibit similar growth rates in execution time as the array size increases. This is expected because both Quick Sort and Randomized Quick Sort have an average time complexity of O(n log n) and their performance tends to degrade as the input size grows.

**Conclusion:**

Based on the provided data and the observed trends, Quick Sort appears to be the more efficient and faster sorting algorithm compared to Randomized Quick Sort for the given array sizes. However, it's important to note that the choice between these two algorithms may depend on various factors, including the specific input data and implementation details. Further experiments and analysis may be needed to make a conclusive decision about which sorting algorithm is more suitable for a particular use case.