# Experiment 3

WAP to perform comparative analysis of iterative soring algorithm (Bubble, Insertion, Selection)

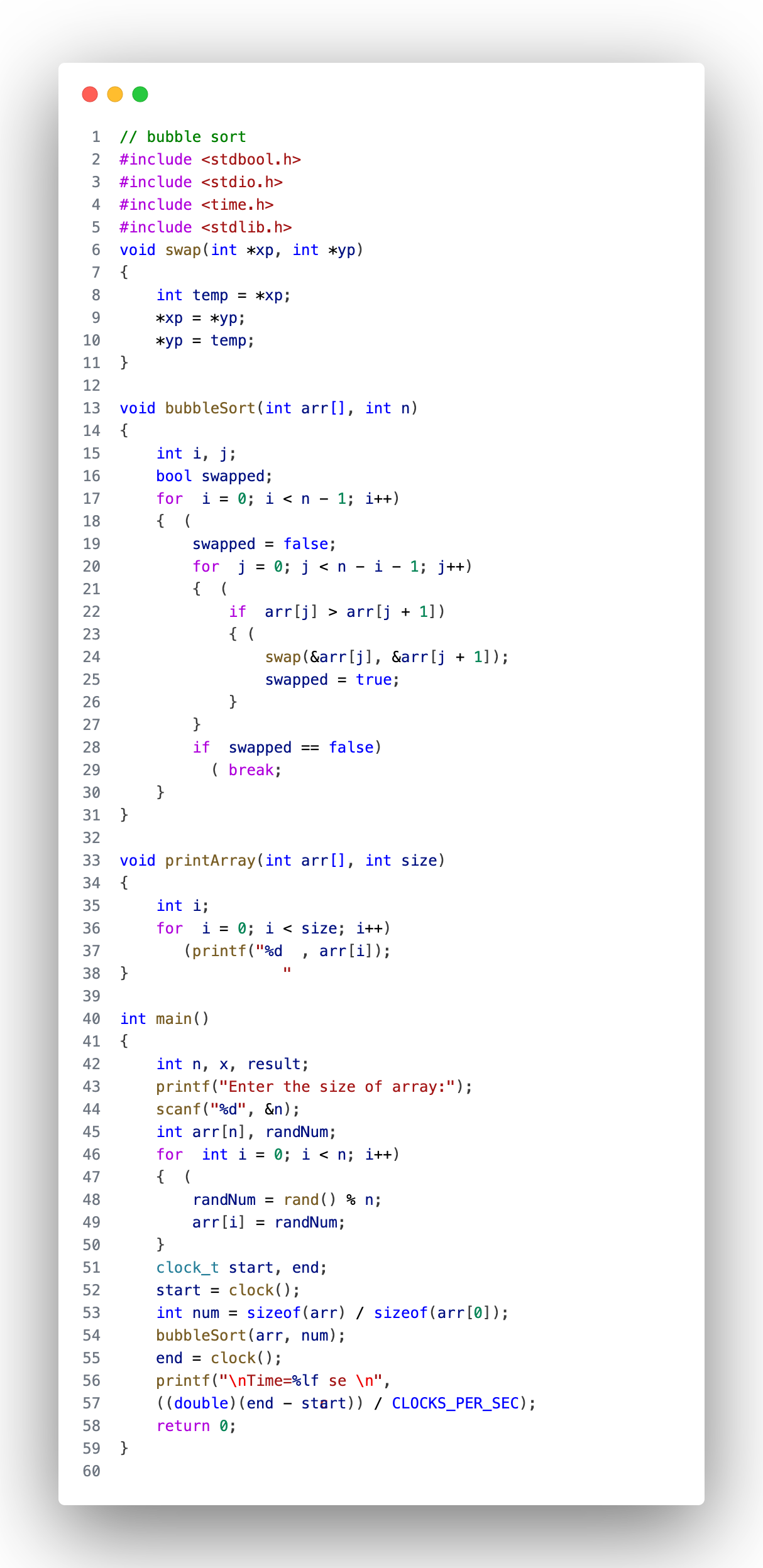
****

Fig: 1 (Bubble sort)

## 

Fig: 2 (Insertion sort)

## 

Fig: 3 (Selection 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 thousand) | time (sec)  (bubble sort) | time (sec)  (selection sort) | time (sec)  (insetion sort) |
| 30 | 3.1929 | 1.0289 | 0.6471 |
| 50 | 8.9243 | 2.8556 | 1.7969 |
| 70 | 17.514 | 5.6911 | 3.5501 |
| 100 | 41.62 | 11.438 | 7.2447 |

The graph shown below is the plot of input n and the time in milliseconds taken by the algorithm while running on a system recorded in table above.

**Comparative Analysis**

1. **Bubble Sort**:
   * Bubble Sort exhibited the highest execution times among the three algorithms for all array sizes.
   * It demonstrated poor performance, particularly as the array size increased, with a substantial increase in execution time.
   * Bubble Sort is not recommended for sorting large datasets efficiently.
2. **Selection Sort**:
   * Selection Sort performed better than Bubble Sort but was still relatively slow.
   * Execution times increased with larger array sizes, but it was generally faster than Bubble Sort.
   * It is not the most efficient choice for sorting larger datasets.
3. **Insertion Sort**:
   * Insertion Sort outperformed both Bubble Sort and Selection Sort for all array sizes.
   * It consistently had the lowest execution times among the three algorithms.
   * Although execution times increased as the array size grew, Insertion Sort remained the fastest choice.

**Conclusion**

In conclusion, based on the experimental data, we can recommend the following insights:

* For small to moderate-sized arrays, **Insertion Sort** is the most efficient sorting algorithm among the three considered.
* **Bubble Sort** and **Selection Sort** are not suitable for sorting large datasets efficiently due to their poor performance with increasing array size.
* For significantly larger datasets, more advanced sorting algorithms such as Merge Sort or Quick Sort should be considered to achieve better performance.

This experiment highlights the importance of choosing the appropriate sorting algorithm based on the dataset size and performance requirements.