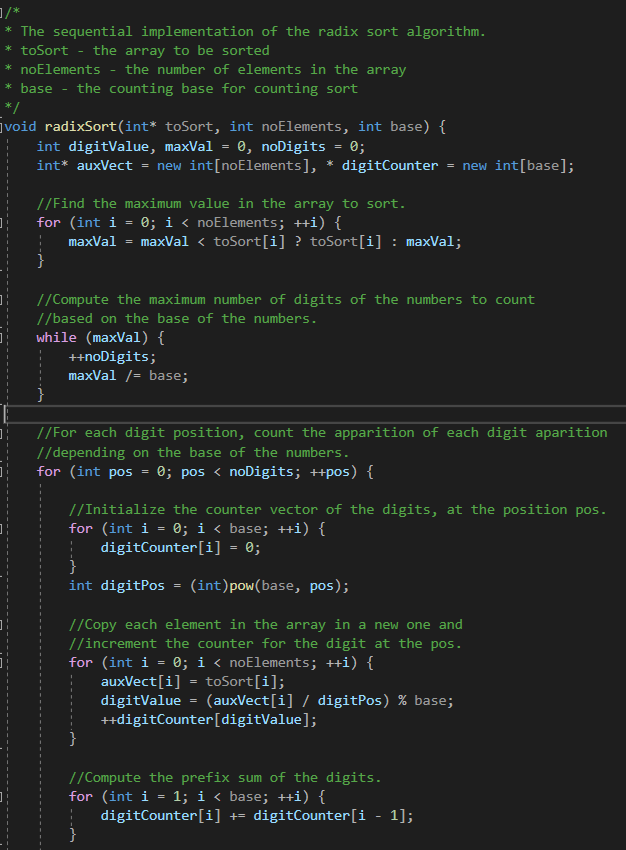
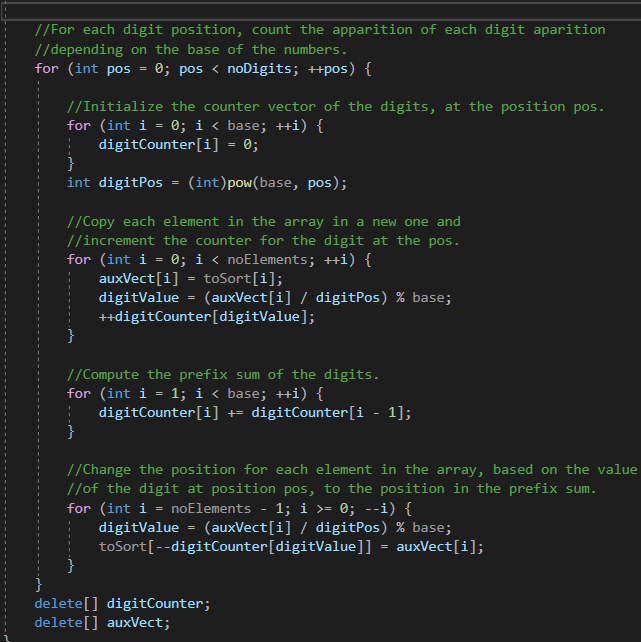
**Sequential version of radix sort**

**Implementation of the algorithm**

For implementation I used the counting sort algorithm with the numeration base 256. Also, I implemented the LSB (least signifiant bit) version of the algorithm. This means that the algorithm sorts from the least signifiant digit of the numbers (the digit of units in the chosen numeration base) to the most signifiant one.

In the following is the documented implementation of the radix sort algorithm.

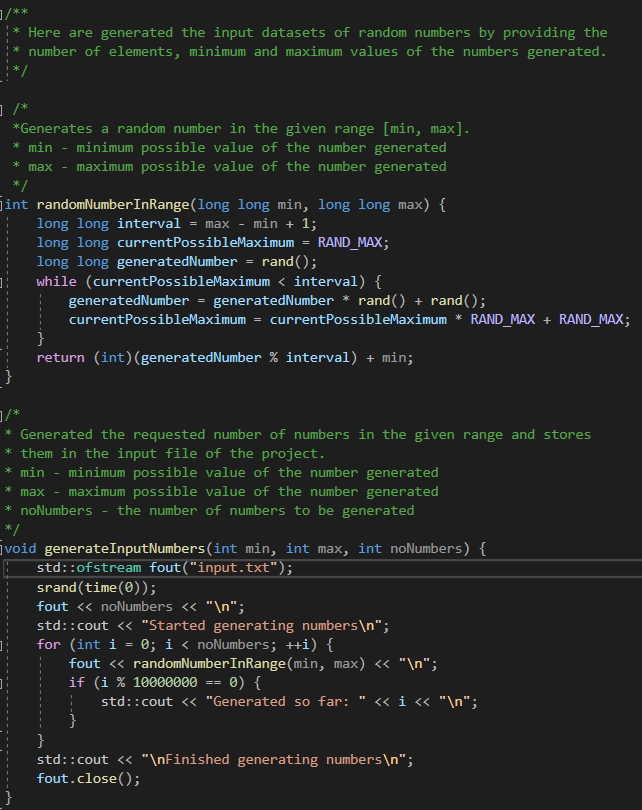


**Generating the input data**

For generatin the numbers that will be sorted, I used library random and time.h and for setting the seed of the rand function and for generating short numbers.

In order to generate larger numbers, I used multiple short numbers generated by the rand function to create a larger number. In my version, the numbers generated can take atmost the INT\_MAX value, but this can be easily changed to generate even larger numbers.

In the following will be shown the 2 methods for generating input numbers and for writing them in the input file.



**Graphs for execution times**

For the execution times, I used the libraries ctime, chrono and ratio for the timer; and for comparison the standard sort and the quick sort algorithms. The time is displayed in milliseconds.

|  |  |  |  |
| --- | --- | --- | --- |
| No. elements | Std sort(ms) | Quick sort(ms) | Radix sort(ms) |
| 10 | 0.0081 | 0.0052 | 0.0213 |
| 100 | 0.0584 | 0.0212 | 0.0263 |
| 1000 | 1.2922 | 0.2999 | 0.0665 |
| 10000 | 12.5598 | 3.3251 | 0.5466 |
| 100000 | 163.37 | 41.5596 | 5.398 |
| 1000000 | 1965.39 | 499.703 | 54.4107 |
| 10000000 | 21695.7 | 5563.04 | 533.558 |
| 100000000 | 250260 | 63553 | 5416.76 |

To be able to make a visual comparison, I also created a table for the graph using the logarithmic values (base 10).

|  |  |  |  |
| --- | --- | --- | --- |
| No. elements(lg) | Std sort | Quick sort | Radix sort |
| 1 | -2.091514981 | -2.283996656 | -1.671620397 |
| 2 | -1.233587153 | -1.673664139 | -1.580044252 |
| 3 | 0.111329737 | -0.523023534 | -1.177178355 |
| 4 | 1.098982724 | 0.521804711 | -0.262330373 |
| 5 | 2.213172309 | 1.618671359 | 0.73223288 |
| 6 | 3.293448742 | 2.698711957 | 1.735684313 |
| 7 | 4.336373667 | 3.745312183 | 2.727181636 |
| 8 | 5.39839144 | 4.803136056 | 3.733739594 |

It can be observed that the time needed by the Quick sort for sorting a large amount of numbers is similar to the time needed by the Radix sort to sort 10 times more numbers. Also, Standard sort takes approximatively 40 times the time needed by the Radix to sort the same amount of numbers.

**Alternatives for parallelization**

For implementation when using MPI, the numbers can be scattered between all processes used in order to count each number locally. After that, the locally computed counters will be summed in a global counter. For counting locally can be used counting sort of bucket sort. The advantage for counting sort is that it will have an array of fixed size, but will be harder to parallelize and redistribute de repositioned number. When using bucket sort, the size of the arrays of the counters will be variable, but will be easier to distribute the numbers.

For Java Threads can be used synchronized methods, volatile containers and cyclic barrier.

**References**

[Fastest sorting algorithm for distributed systems (Parallel Radix Sort) [Difficulty: Medium] – PRACE Summer Of HPC (prace-ri.eu)](https://summerofhpc.prace-ri.eu/fastest-sorting-algorithm-for-distributed-systems-parallel-radix-sort-difficulty-medium/#:~:text=The%20parallel%20Radix%20Sort%20is%20one%20of%20those,it%20is%20the%20word%20often%20used%20in%20HPC.)

[srt (buffalo.edu)](https://cse.buffalo.edu/faculty/miller/Courses/CSE633/Narendran-Sankaran-Spring-2014-CSE633.pdf)

[Partitioned Parallel Radix Sort (semanticscholar.org)](https://pdfs.semanticscholar.org/7c72/dc25d41e358b8e4508de34c61482e5c54326.pdf)

[sorting - Parallel radix sort, how would this implementation actually work? Are there some heuristics? - Stack Overflow](https://stackoverflow.com/questions/26206544/parallel-radix-sort-how-would-this-implementation-actually-work-are-there-some)

[AMD FidelityFX - Parallel Sort - GPUOpen](https://gpuopen.com/fidelityfx-parallel-sort/)