Dylan Lovin

SNHU CS-300

1/26/25

Assignment 2-3

This program is designed to sort a collection of bids stored in a vector using two distinct algorithms: selection sort and quick sort.

The first algorithm, selection sort, operates by identifying the smallest value (min) in the list and comparing it to the subsequent values. If a smaller value is found, it becomes the new min, and the two values are swapped. This process is repeated for all elements in the list, ensuring that the items are arranged in ascending order by the end.

The second algorithm, quick sort, works by dividing the list into two partitions: low and high. A pivot value, chosen as the middle element of the collection, is used as the dividing point. Elements smaller than the pivot remain on its left, while elements greater than the pivot are moved to its right. The quick sort method is then called recursively on the left and right partitions until the entire collection is sorted in ascending order.

During development, the primary challenges stemmed from syntax errors and issues with the Integrated Development Environment (IDE) configuration. Debugging with Visual Studio was instrumental in resolving these problems. For instance, a common mistake occurred when the program continued running in the background, resulting in linker errors during recompilation. The solution was straightforward: exiting the running program before rebuilding and recompiling the project. These errors, though typical for beginners, provided valuable learning experiences.

Psuedocode:

START PROGRAM

WHILE choice IS NOT equal to 9

EXECUTE

PRINT user menu:

1. Load Bids

2. Display All Bids

3. Selection Sort All Bids

4. Quick Sort All Bids

9. Exit

READ user input and store in variable `choice` for menu selection.

IF choice IS 1 (Load Bids)

SET `ticks` to clock() to record starting time.

INITIALIZE CSV file.

PARSE CSV file and store data into bid object.

DEFINE vector to store a collection of bids.

RETURN bids.

PRINT the size of the bids collection.

CALCULATE elapsed time: clock() - `ticks`.

PRINT elapsed time.

ELSE IF choice IS 2 (Display All Bids)

LOOP through the bids vector.

PRINT each bid.

INSERT a break.

ELSE IF choice IS 3 (Selection Sort All Bids)

SET `ticks` to clock() to record starting time.

INVOKE the selection sort method, passing in the bids vector.

CALCULATE elapsed time: clock() - `ticks`.

PRINT elapsed time.

INSERT a break.

ELSE IF choice IS 4 (Quick Sort All Bids)

SET `ticks` to clock() to record starting time.

INVOKE the quick sort method, passing in the bids vector, start position, and end position (bid size - 1).

PRINT the size of the bids collection.

CALCULATE elapsed time: clock() - `ticks`.

PRINT elapsed time.

INSERT a break.

ELSE

PRINT "Goodbye".

RETURN 0

END PROGRAM