Daniel Loranger

CS-300

Week 2 assignment

* Code Reflection – Overall I found the code fairly straight forward to manipulate, with minor struggles to migrate the code and methods from the zybooks learnings into functional code. Most of the challenge was around the quicksort implementations. Initially I was unable to get started as I was expecting the files inside the folder tree to automatically be brought into the project but compiling failed with unusable cryptic errors that were of no help in triaging the problem. Additionally, my local IDE environment was missing the windows desktop templates which was resolved by adding the additional modules via the installer.  
    
  When running the selection sort algorithm, my console terminal would appear unresponsive for extended periods of time, which would rightly be interpreted as a crash if there is no warning that this could take a long time. To help with this, a notification to the user has been added.  
    
  While the code does not crash, there is no indication that the vector to sort is empty if not loaded beforehand, this could be enhanced to alert the user of an input problem.
* Pseudo-code
  + Handle command line arguments (if any)
  + Initialize variables
  + Display menu tree
    - 1- load raw data from file
      * Capture start time
      * Load data
      * displayDuration (startTime)
    - 2-display raw data that has been loaded
    - 3-perform selection sort algorithm
      * Capture start time
      * Perform sort
      * displayDuration (startTime)
    - 4-perform quick sort algorithm
      * Capture start time
      * Perform sort
      * displayDuration (startTime)
    - 9-exit
  + (void) selectionSort (vector of bids)
    - Increment from index 0 looking for the lowest remaining value
    - Swap lowest value into the next lowest position
  + (void) quickSort (vector of bids, int start pos, int end pos) (recursive capable)
    - split input vector by using partition function
    - quick sort lower partition
    - quick sort higher partition
  + (int high pos) partition (vector of bids, int begin pos, int end pos)
    - Init variables
    - Find midpoint
    - Find low pos
    - Find high pos
    - Swap sort values
  + displayDuration (clock\_t start time)
    - compute elapsed time
    - display elapsed time
* Specifications and Correctness – Code has been tested using the short list via commands (in order) 1,2,3,2 and 1,2,4,2 respectively to show the unsorted values, then sorted values. All testing was performed using locally coded paths, command line arguments are not tested
  + Step 1 – elapsed time 10 ticks (76 records)
  + Step 2 – unsorted records presented
  + Step 3 – elapsed time 10 ticks
  + Step 2 – sorted records presented
  + ----
  + Step 1 – elapsed time 11 ticks (76 records)
  + Step 2 – unsorted records presented
  + Step 4 – elapsed time 0 ticks
  + Step 3 – sorted records presented
* Specifications and Correctness – Code has been tested using the long list via commands (in order) 1,2,3,2 and 1,2,4,2 respectively to show the unsorted values, then sorted values. All testing was performed using locally coded paths, command line arguments are not tested
  + Step 1 – elapsed time 2250 ticks (12023 records)
  + Step 2 – unsorted records presented
  + Step 3 – elapsed time 244804 ticks
  + Step 2 – sorted records presented
  + ----
  + Step 1 – elapsed time 2264 ticks (12023 records)
  + Step 2 – unsorted records presented
  + Step 4 – elapsed time 205 ticks
  + Step 3 – sorted records presented
* Annotation and Documentation – Code is commented inline respective to the immediate commands
* Modular and Reusable - additional helper function displayDuration() was added as there was redundancy in the code when displaying the computed time for each process.
* Readability – White space is autoformatted using the VS IDE tools, variables utilize the camelCase formatting which is the default inherited from existing code.