**CS 499 Artifact Two Enhancement: Storing and Sorting Bids Program**

**By Alexander Varljen**

**August 13, 2022**

**Description of the Artifact:**

The original project from which this artifact enhancement was taken from was called “CS 260 Data Structures and Algorithms: Hash Tables and Chaining”, which I created for an assignment back in February of 2021. The goal of the project was to create a program that, via four menu options, allowed the user to load bid data from a csv file into a Hash Table data structure, display all the bids from within this data structure, find one bid from the list stored in the Hash Table, and remove one bid from the Hash Table. My enhancement to this artifact was adding vector sorting capacity with four new menu options, one for loading the bids into the vector that was used for the vector sorting, one for displaying all the bids in this vector, one for sorting the bids in that vector via selection sort, and one for sorting the bids in that vector via quick sort. This enhancement was made in the third week of July in 2022.

**Why This Artifact was Chosen:**

I selected this artifact as my algorithms and data structures enhancement because it demonstrates my ability to store and manipulate data using multiple forms of data structure and different algorithms. These included an unmapped vector containing bid data and a mapped, or “hashed”, vector, also containing the bid data from a csv file. The unmapped vector was used by the vector sorting algorithms “quick sort” and “selection” sort, with selection sort being a simpler, but less efficient, algorithm and selection sort being a more complex but more efficient sorting algorithm, both of which were used to sort the bid data in ascending order. The mapped, or “hashed”, vector is a more complex vector to construct, but provides a much faster and more efficient way of finding an item, like a bid, than compared to using an unmapped vector. By including both types of vectors and their corresponding algorithms in this project, I was able to demonstrate that I can work with multiple algorithms and data structures, which each have their trade-offs, but can be utilized for different problems and desired outcomes.

**Course Objectives:**

While working on this artifact enhancement, I was able to employ strategies for building collaborative environments that enable diverse audiences to support organizational decision making in the field of computer science. Those who want to interact with the source file, download it for themselves, make desired changes and even, potentially, recommit changes on an updated version of the file can easily do so via my public GitHub repository. Also, the source code of the artifact enhancement has been made modular, which means that others change modules, delete modules, or add their own modules with a few code changes and without much impact on the other modules. Also, the code comments that I had written for the project make it easier for others to add to, change, navigate, or learn from the code. For example, when one opens the source file, they’ll notice a short description of the source code’s functionality at the top of the file, two boxed-in code comments highlighting two main divisions in the program, which are the modules for the hashed table data structure and those for the vector sorting algorithms, and code comments describing each module. These help others working with and navigating through the source code understand how everything works together so that they can better participate.

While working on this artifact enhancement, I was able to design, develop, and deliver professional-quality oral, written, and visual communications that are coherent, technically sound, and appropriately adapted to specific audiences and contexts. The second artifact covered in the code review video is the original artifact for this enhancement, where I discussed the original source code, module by module, and assessed my own previous work. I’ve created two flowcharts for this artifact enhancement, one of which is a depiction of the functionality of the artifact before the enhancements, and the other a depiction of its functionality after the enhancements where made. I also wrote this descriptive narrative which helps explain my artifact, its enhancements, and my intent behind the enhancements.

While working on this artifact enhancement, I was able to Design and evaluate computing solutions that solve a given problem using algorithmic principles and computer science practices and standards appropriate to its solution, while managing the trade-offs involved in design choices. To solve the problem of sorting the bids, as well as show competence in creating functional algorithms, I created two vector sorting algorithms. This, however, was met with a trade-off in the form of having to add an extra data structure other than the hash table, which meant that the bids needed to be loaded by the program a second time. This led me to add two more menu options other than the two for the vector sorts, one for loading bids to the new vector and one for displaying the bids from this new vector, along with changing the menu labels to help the user understand the difference. This solution allowed me to manage this trade-off and let the user decide when to load the bids a second time while also allowing me to add two new functional sorting algorithms and meet my artifact enhancement goals.

While working on this artifact enhancement, I was able to develop a security mindset that anticipates adversarial exploits in software architecture and designs to expose potential vulnerabilities, mitigate design flaws, and ensure privacy and enhanced security of data and resources. This was done by adding exception handling in the artifact enhancement’s source in key areas, which are in the modules that contain the code used to load bid data. These let the user know if there is an issue with the program when loading the bids. As the csv file containing bid data could, potentially, hold thousands of “bid id’s”, “titles”, “funds”, and “amounts”, which could be filled with the wrong type of data or come from a corrupted file, this seemed to be the area of the most vulnerability. I also made sure that the code ran without any errors.

**Reflection:**

As part of the process of completing my enhancements to this artifact, I had to look over the program thoroughly to understand how it works, what enhancements where possible without changing the whole program, and, when those enhancements were decided on, what programming decisions would be necessary to complete those enhancements. This was made easier by my familiarity with C++, which is the programming language that I am most familiar with. While going through this process of reviewing the projects code, I noticed that the Hash Table data structure doesn’t easily lend itself to vector sorting, so I added an additional vector. This became central to many of my other programming decisions, which includes four new menu options and an extra algorithm for loading the new vector with the csv file data. I don’t believe that a set of menu options like this would normally take place. However, I wanted to display multiple forms of data structures and algorithms while using the same source of data and the same theme of the program, which was storing, adding to, removing, and looking up bids. So, to have a coherent program with one main theme while still demonstrating multiple data structures, I added a second set of options that were labeled differently for the users benefit.