Anthony Browness

CSIS 2420-001

Reflection Paper

After reading the section I can see how people may want to take sides and try to decide once and for all which implementation is better outright. The tables at the end of the section that show the Time/Complexity of each implementation can bias one towards arrays. This is the wrong way to think about it, and instead one must consider the specific situation that is being programmed for and thus the correct implementation to use in that situation.

Consider a system that has very little memory, using an array may take too much of a chunk from the memory pool and cause an error, in this case it may be best to use a linked list especially if there are many objects that need to be stored.

Conversely a project that will have data *accessed* on a very regular basis would favor an array due to its ability to calculate memory locations with very little complexity whereas a linked list must transverse the list to access a memory location. If the project must *manipulate* the store of objects on very regular basis a linked list might be a better implementation due to its ability to store objects in a dynamic fashion, though, the location of storage be it head/bottom, tail/top, or middle of the store should be considered due to the differing complexities of each implementation.

Keeping the pros/cons of each implementation in mind while deciding the most efficient means for a project will surely lead to more efficient programming practices, and in the end, a better program.