Lab 2 Questions

Draven Schilling

CS3841

9-20-19

1.All of the list functions need a “reference” to the list structure, and according to this design, that list reference is passed as a pointer. Why is this necessary? Do all of the list functions need this to be passed as a pointer? Any exceptions? Be specific in your answer.

All the list functions that we used in this lab required passing their arguments by pointer. In our case, the list specifically needed to be passed by pointer because it was an instance of a struct and in order to avoid duplicating the contents of the struct on the stack (passing by value) a pointer was needed. The same argument is also made for passing nodes. Another side benefit of passing node structs by reference is that we can determine if their data is equal by simple equality (node1 == node2?) which is convenient for our use. We also had to pass strings by pointer because of the same aforementioned benefits but also because we don’t know the length of the string, and a pointer will allow us to avoid that conundrum because C style strings are null terminated.

Overall, because the data we were passing were all either structs, or strings, pointers were preferred. If the data were simple types such as characters or integers, pointers would have been unnecessary.

2.Unlike a Java or C++ implementation, this implementation cannot “hide” any of the internal structure of the list. That is, users of the list could mess up the next and prev pointers if they are careless. Can you think of any way we could hide the structure of the list to lessen the chances a user will mess up the list? Describe in brief detail.

Since there are no objects in C, you cannot embed the list in any wrappers. You could make its fields constant which would make it so the user could not edit them, but that would also not allow methods to edit them either.

Given these considerations C is just a language that places a lot of trust in the user and is under the assumption that they know what they are doing and will make educated decisions. Therefore there may be a way to “hide” the structure of the list but it’s not obvious or seemingly straightforward.

3.What if all llClear() did was assign NULL to head and tail in the list structure and nothing else. Would the program crash? Would there be any side effects? Try it and report results.

The program would not crash nor would there be any immediately noticeable side effects. Instead, the nodes and associated strings would be lost in memory leaks which if left running over a long period of time with many resets may result in the heap running out of allocable memory. This is why it’s important to “free the malloc’s”.

4.This design requires the user to iterate the list somewhat manually as demonstrated in the sample driver. Propose the design of an iterator for this list. What data items would the iterator need to store (in a structure, perhaps)? What functions would the iterator supply?

The iterator could store the current position within the list as well as the node associated with that position. Methods to set the position and increment/decrement the position would be useful. The intended usage would be easier access to getting nodes within the list.