CS23820 Individual Assignment: Sailing Club Membership Service

Veronika Karsanova, vek1@aber.ac.uk

12th of December 2019

1 Introduction

This document details the implementation of the Sailing Club Membership Service, outlined in the assignment brief [1] for the CS23820 module.

It consist of the analysis of the given tasks and self-evaluation.

All of the requirements for Membership program were fulfilled, but no attempt at the Race program (15% of the assignment) was undertaken.

2 Self-evaluation

The requirements of the brief [1] gave a head start to the beginning of this coursework. Knowing that all the information needs to be stored in a Binary Search tree, it was obvious for me to go through the Case Study slides[2][3] once again, knowing that the data structure used in the assignment from couple of years ago was, in fact, the Binary Tree I could refer to while working on the code.

The first couple of steps were quiet straightforward, with the very first one being the creation of the struct "member". It takes in six variables, which together represent the, hopefully, unique participants of the Sailing club. Apart from those six variables, we also have two pointers, which point at the children of the current node (*left* and *right* child respectively). That is, if you wish, a "feature" of building a Tree yourself.

Creating and populating our tree was the next required step. In order to do that I am allowing user to insert the path to the file, which will provide us with the data. This data will be used in order to create the nodes. First-off, I initiate a variable of type "member_node_ptr" *tree_root* as a NULL. The if statement located a bit lower in the code will check whether or not the root is definied, and if it is not then the first node read from the file will become the root of our tree. However, once the root is definied – we need to insert the follow-up tree nodes in the right order. In our case, they are sorted by family names. I did remind myself how similar insertion worked in Linked List[4], before starting on that function of the program. What we are doing, is comparing the family names using *strcmp()*, and then updating the left or right child pointer, depending on the subtree the new node should belong to.

With that, our binary search tree is succesfully build. The function for printing is using in-order traversal implementation, which allows us to have everything being printed from the smallest value, or, in your case, in alphaberical order. We are first visiting left subtree of the root, printing all the nodes there one by one, and then doing same thing for the right subtree.

When it came down to implementing the search options for ID and Boat class, I had quiet a few designs to choose from. At first, I was using a post-order traversal and simple compairason to check whether the node is matching the one we looking for: if it did, it was printing out the result. It worked perfectly fine, but, however, I had 2 separate func-

tions: one handled the ID search (which was accepting int as a parameter), another one handled boat class search (pointer to char as a parameter). That was causing some code duplication and I was thinking that it would be rather inefficient, if we would want to have search methods for everything. That is why I updated the code to have a function, which accepts as one of the parameters a pointer to another function. By doing that, we are not repeating the same traversal code multiple times, but just calling different functions to perform the various compairasons following the requirements. Furthermore, one of the parameters of "searchForTheNode" is a char value, which, depending on what we are looking for, can be converted to, for example, an int. Also, by swapping my design and making my functions of type *int*, I am able to make sure that if the node is not in the tree – we are getting a print statement to tell us so.

Creating new member is not any different from adding new nodes at the beginning of the file: the only thing which is different, is that we are scanning in the information from the command line.

Once the user chooses to save the file, we are putting the file from which we read in the data in the write mode[5] and passing the tree node (signifies the root) in. Once file is ready to be used, we calling function which is overwriting the contents of the file with our current binary tree, using pre-order traversal. You can use both the in-order traversal and post-order traversal, but, if I remember correctly from the Algorithms module, doing it in-order can sabotage the tree, since we, essentially, will be building an array, if we build the Binary Search ordered from the very beginning.

Couple of warning remain in the code, but they do not seem fatal: all of them are referring to the "strtol" function, which is preferable to use, when it comes to reading in ints. However, since we were not introduced to it over the period of our practicals and it was considered as acceptable practice to use "scanf" and "fscanf" for decimals, I decided to keep them, though now I am more aware of problems with them after reading quiet a few articles online.

What I would like to improve is the lack of error-checking. For example, despite the variables holding the membership ID data being integers, we are getting neither error nor warning if we input a char of a kind. In the menu, if you insert a char instead of an int – you end up in an infinite loop. Another thing is double-checking for duplicates, aka making sure that do not input the same details twice. Furthermore, if it was up to me, I would add the removing of nodes to the functionality of the program, since in the situation of accidental duplication, we would be able to remove the node directly, while running the program.

Taking everything in the consideration, based on the version 1.2 of Assessment Criteria for Development, even though my work most certainly has room for improvement and is not exceptional in any way, I think that I managed to show a good understanding of the problem and was able to present it as such. Due to that, I believe that I have earned myself around 70 percent of the total mark.

References

[1] Dave Price, CS23820 Assignment - C and C++, 2019
`shorturl.at/enCW4`

[2] Dave Price, Case Study One, 2019
`shorturl.at/htPU2`

[3] Dave Price, Case Study Two, 2019
`shorturl.at/ginZ8`

[4] Dave Price, Worksheet Four (Linked Lists), 2019
`shorturl.at/zCQSZ`

[5] Tutorialpoint, C – FileI/O, 2019
`shorturl.at/cmOQV`