**Act 2.2 - Functionalities of a linear data structure verification**

Diego Partida Romero A01641113

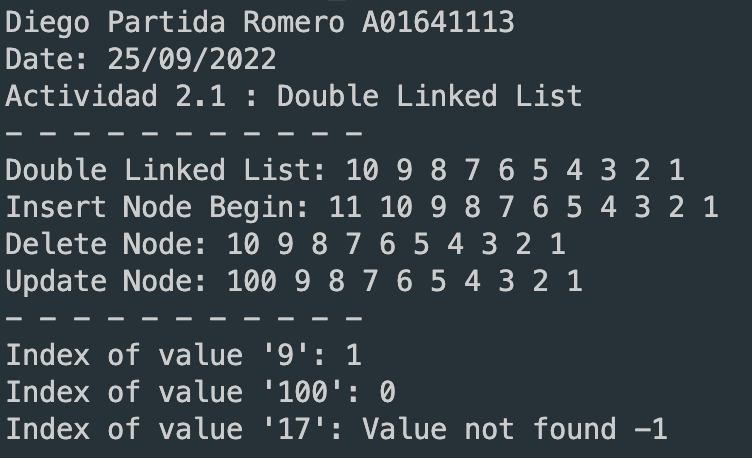
9 oct. 2022

**Explanation of different sorting and search algorithms for this problem situation:**

It is important to use double linked list because it is easier to implement than a singly linked list. While the code for the doubly linked implementation is a little longer than for the singly linked version, it tends to be a bit more “obvious” in its intention, and so easier to implement and debug.

Test cases:

|  |  |  |  |
| --- | --- | --- | --- |
| Function | Input | Output | Time Complexity |
| Create | You pass the head and the value to insert at the beginning of the list  head = insertNodeBegin(head, 1);  …  head = insertNodeBegin(head, 10);  printDoubleLinkedList(head); | 10, 9, 8, 7, 6, 5, 4, 3, 2, 1 | O(1) because it only has one operation |
| Delete | You pass the head and the value you want to delete from the list and it will return the new head  head = delNode(head, 10);  printDoubleLinkedList(head); | 9, 8, 7, 6, 5, 4, 3, 2, 1 | O(n) because it uses a while loop |
| Update | You pass the head, the value you want to update and the new value  head = updateNode(head, 9, 100);  printDoubleLinkedList(head); | 100 8 7 6 5 4 3 2 1 | because it has to go through the whole list |
| Search | You pass the head and the value you want to search and it will return the index of the value  cout << "Index of value '9': " << searchNode(head, 9) << endl;  cout << "Index of value '100': " << searchNode(head, 100) << endl;  cout << "Index of value '17': " << searchNode(head, 17) << endl | Index of value '9': 1  Index of value '100': 0  Index of value '17': Value not found -1 | O(n) because it has to go through all the nodes" |



In this program I use search function to find the index of the node we want to look for I also implemented a function to update the value of a node in the list and a function to delete a node in the list by value of the node, all of this functions have a time complexity of O(n) because we have to go through the list to find the node we want to look for except the insert function that have a time complexity of O(1) because we only have to add the node at the beginning to the list and change the pointers; all of the functions work with pointers to save memory and to make the program more efficient.

I use these test cases to check if the functions work correctly, and becuse it's easy to see the results in the console and understand what is happening.

I can conclude that the functions work correctly and these implementation helps me to understand the double linked list better.