Group-BW

Assignment-3-Implementation

Git Repository Link:

https://csgitlab.ucd.ie/Ider/comp10050-assignment-3-kanban-style-board

How we implemented the load board function:

The load board function is declared as “boardPtr loadDataFromFile()”

Which returns a pointer variable to the start of the board struct linked list and takes no parameters. First we declare our variables such as boardCounter, itemCounter and pointer variables that point to board and item linked lists.

We ask the user to enter the filename to load the board data from.

This is done using “Fgets” with a maximum buffer of 30 characters from standard input. We used “fgets” because it allows us to prevent character buffer overflow from entering a large amount of characters. We then made sure to remove the newline character from the inputted file name and replace it with the string termination character (‘\0).

We use a while loop with the condition that “fgets(string, MAX\_CHAR, loadFile)!=NULL)”. This will continue the loop until it reaches the end of the file. “String” is just a placeholder value for the scanned string from the file.

We then use an if statement that checks if the scanned string contains a colon (“:”) using the “strchr” function. If the scanned string contains a colon then we know its a board name and allocate memory to the struct pointer variable “newBoardNode”. We then increment boardCounter and then store it into the board struct linked list.

While doing so we make sure to set the pointer to the previous node of the linked list to NULL for the first node (i.e when boardCounter==1).

Otherwise in the next iterations of the loop, we link the new board with previous board using the “->” operator for struct pointers.

We then have an if statement checking if the boardCounter is equal to 2, if so then we need to store the head of the list in “fileData\_StartNode” . This will be returned by the load function and set it equal to a “start node” variable in main that will allow other functions to access the linked list as well.

Else if the scanned string doesn’t have a colon then we do the same thing we did with the board. We increment itemCounter, then initialise our first node if itemCounter is equal to 1. We also close the previous item list by checking if boardCounter is greater than 1.

Then we link the item to the board and copy our item name into the item list node name.

At the end of our function we return the fileData\_StartNode.

How we implemented the edit and lists and items function:

Our function editList(BoardPtr startNode) edits the list and items.

First we declare our bool values that check whether the list is found and if the initial null node is set.

Fpurge then clears the stream. We prompt the user to enter the file name using “fgets” and we remove the newline character.

We then declare our linked list pointers and while(currentNode!=NULL) loop will continue until it reaches the end of the linked list. This way we can search the whole linked list using if statements with “strcmp” function. If it does exist then we bool value equal to true. And break the loop.

Else, it will move onto the next node in the list.

An if statement checks if the listFound bool value is false then it will print a message saying it cant find the list.

Else if the list name is found we print a menu detailing various options such as editing an item, adding and deleting or returning to the menu.

We use a switch statement to check what the user chooses. If they choose

To edit an item then we use another while loop to check if it exists and set our bool value equal to true if it does.

If it exists then we ask user to enter the new name for the item and replace the name using “strcpy”.

If the user chooses to add a new item

How we implemented the saving board to a file function:

We implemented the function with the name “void savingBoardToFile(boardPtr startNode)” that returns void and takes as a parameter a pointer to the board struct linked list start node. This start node is defined in the main.c “main” function and allows other functions to access the same starting node for the linked list. We set a pointer variable to board struct called “head” equal to the startNode so that we may access the linked list data without changing the value of start node for other functions

First it asks the user to enter the filename they wish to save the board to. We used the “Fgets” function to take in the string input as it allows us to set a maximum buffer limit of 30 characters to be taken in. This prevents the overflow of memory from excessive character inputs.

We also made sure to remove the newline character at the end of the string and replace it with the null string terminator.

We then use “fopen” in “write” mode and set it equal to a FILE pointer type. An if statement checks if the file couldn’t be opened.

If the file can be opened, then it will declare a pointer variable to the item struct. A while loop is used with the condition that “head!=NULL”.

We use the function “fprintf” to save the current board nodes board name into the file. And then it will move onto the next node in the linked list. This will keep looping until the “while” condition is broken (when the end of the linked list is reached).

Then itemListNode is set equal to the item list struct of the current “head” value and another while loop with a similar condition “while(itemListNode!=NULL)”. It uses “fprintf” to save the item list name into the file and moves onto the next node in the linked list.

This keeps looping until the end of the item linked list is reached.

Our Ascii Art:

Since the group members have a shared interest in music, we decided to reflect that in our Ascii Art through musical notes such as the treble clef, the quaver and the beam note.