Assignment 3 - Dave Ranola, Ewa Zoladek

How to load a board?

In our project, we decided to to load a board from a binary file (.dat).

The function used to load a board is:

```
void LoadBoard(Board **header, char FILENAME[80])
```

Paramaters:

Board** header – the head of the linked list, double pointer because it needs to be edited

char FILENAME[80] – the name of the file, which the data is loaded from

How it works:

- Open the file with *fopen*(FILENAME, "rb"), rb as it is a binary file, r for read
- If this is NULL, it means there is an error opening the file
- Otherwise, fread(&boardCount, sizeof(int), 1, fptr); -> the first item in the file is the boardCount, hence why we read it and assign it to &boardCount
- After, the binary file format is the boardName, itemCount and itemNames, which repeats until there is no data left
- This is why in the following order we *fread*(boardName, sizeof(char), 80, fptr); so we know the boardName of the item to be loaded. Then we add this board to the end of the list for the correct format: *addBoardEnd*(header, boardName);
- We then initialize the variables of the board, *fread*(&itemCount, sizeof(int), 1, fptr);, how many items the board has, and in the for loop, with itemCount as the guard, we read the item names *fread*(itemName, sizeof(char), 80, fptr);, and assign it to the board either, *addItem*(&((*header)->firstItem), itemName); (if the list is empty) or *addItemEnd*(&(current->firstItem), itemName);, if there are existing items in the list.
- We then simply just *fclose*(fptr); to close the file.

How to edit lists and items:

To edit lists (boards), we made the function:

```
void editBoard(Board **header, char boardName[80], char newBoardName[80])
```

Paramaters:

Board** header – the head of the linked list, double pointer because it needs to be edited

char boardName[80] – the name of the board to be edited

char newBoardName[80] - the name of the new board, which is switched with the old name

How it works?

- We make a current pointer, Board *current = *header;, which points at the current item the program is currently at (at the beginning it points to the first board)
- We traverse the list until we find an item with the same name as boardName[80] or until it reaches NULL
- If it reaches NULL, it implies that the current pointer reaches the end of the list meaning there doesn't exist a board with boardName[80]. We simply just return.
- Otherwise, we copy the new name to the old board, *strcpy*(current->boardName, newBoardName);

To edit items, we made the function:

```
void editItem(Item **firstItem, char itemName[80], char newName[80])
```

Parameters:

Item **firstItem – the head of the linked list, double pointer because it needs to be edited

char itemName[80] – the name of the item to be edited

char newName – the name of the new item, which is switched with the old name

How it works:

- We make a current pointer, Item *current = *firstItem;, which points at the current item the program is at (at the beginning it points to the first item)
- We traverse the list until we find an item with the same name as itemName[80] or until it reaches NULL
- If it reaches NULL, it implies that the current pointer reaches the end of the list meaning there doesn't exist an item with itemName[80]. We simply just return

- Otherwise, we copy the new name to the old board *strcpy*(current->itemName, newName);

How do you save a board:

To save a board we made the function:

```
void saveBoard(Board *header, char FILENAME[80])
```

Paramaters:

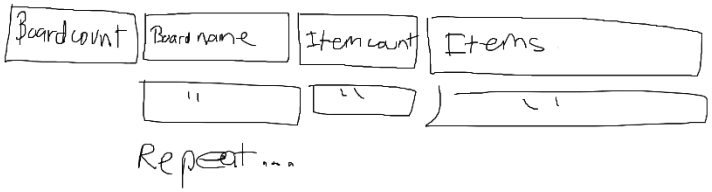
Board *header - the head of the linked list, single pointer because it doesn't need to be edited

char FILENAME[80] – the name of the file, which the data is loaded from

How it works:

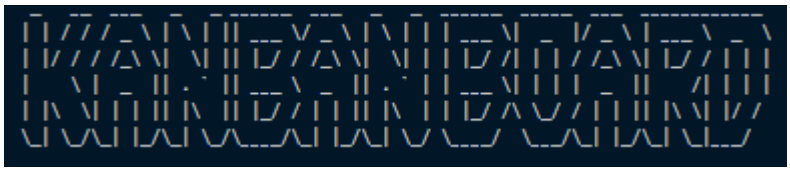
- Open the file with *fopen*(FILENAME, "wb"), wb as it is a binary file, w for write
- If this is NULL, it means there is an error opening the file
- We intitialise a current pointer, so that we can traverse the list, Board *current = header;
- We count how many items there are, with a while loop, and write this into the binary file, *fwrite*(&boardCount, sizeof(int), 1, fptr);
- We reset the current to header, current = header;, after the boardcount loop, the current pointer will point at the last item
- We then save the name of the board, with *fwrite*(current->boardName, sizeof(char), 80, fptr);
- We initialize another pointer, this time an item to count how many items the board has, Item *currentItem = current->firstItem; As we did last time we count how many items there are with a while loop, and then write this into the binary file, *fwrite*(&itemCount, sizeof(int), 1, fptr);
- We reset the currentItem pointer to firstItem again as it will point at the last item of the board after the while loop
- We then write down all the names of the items with the following while loop,
  ```
  while (currentItem != NULL) {
      fwrite(currentItem->itemName, sizeof(char), 80, fptr);
      currentItem = currentItem->next;
  }
  ```
- We then point the current board pointer to the next item, and repeat the process above starting from saving the boardName until we reach the end of the list
- We then simply just *fclose*(fptr); to close the file.

Format of the binary file:



What ASCII art about:

The ascii art lets the user know they are going to use a Kanban Board like program



Git Repo:

https://csgitlab.ucd.ie/DaveRanola/comp10050-assignment-3.git - “2025 Assignment 3 AA” + “Assignment 3”