CS2002 Practical 3 - C Programming 2

Matriculation Number: 160001362

16/03/2018

Contents

1 Overview

This practical specified the development of a system to perform a card trick that finds a user selected card based on the column in which their card resides among twenty-one random cards. The practical further specified that the implementation should utilise an index (array) based solution and/or a pointer based solution. For this submission both an index based solution has been developed as well as a non-global pointer based solution which makes use of linked lists.

For the pointer based implementation I challenged myself to not use a single set of square brackets to reference an index (even for displaying card suits/ranks) which proved challenging but achievable.

To track changes in the project I used git in conjunction with GitHub with my own private git repository. This proved useful to keep a library of working versions of the program that I could roll back to if local changes caused the program to mysteriously break (such as causing a segmentation fault). **TALK ABOUT EXTENSIONS**

2 Design and Implementation

The index based implementation is contained with a folder 'Index_Based' and the pointer based implementation is contained within a folder 'Pointer_Based', both of which are within the 'ReadMyMind' directory. The code is heavily commented to make it clearn how each of the implementations work.

Both implementations of the program have the same following files within their respective directories:

2.0.1 Files

- **readmymind.c** - This file contains the main function of the program. It is responsible for handling the control flow of the program and declaring the necessary structs and variables required.
- **cards.c** - This file is responsible for initialising the fundamental data structures required for use in the program including the deck, the cards, as well as the columns used to store the 21 cards (using getDeck, getCard, and fill respectively).
- **io.c** - Handles the input and output of the solution. The file contains functions to get the user's column selection, validate this input, and convert this input into an integer format more easily utilised in the program. In addition the file contains a function to print the cards of a columns structure to the terminal as well as print the centre card of a columns structure.

- **actions.c** - Contains functions relating to the actions performed during the card trick. The 'gather' function creates a new columns struct and initialises it with the previous columns rearranged so that the column selected by the user is in the centre. The 'deal' function iterates through the cards of each column in the columns struct returned from 'gather' and deals the cards into the new struct from left to right across the columns.
- **readmymind.h** - Contains definitions for the size of data structure dimensions (e.g. COLUMN_SIZE, SUIT_SIZE), the structs representing the data structures, as well as for the signatures of functions used throughout the program.

2.1 Index Based

2.1.1 Data Structures

- **Card** - Has two integer attributes:
  - **suit** - a number from 0 - 3 representing either spade, heart, diamond, or club.
  - **rank** - a number from 0 - 12 representing and Ace, 2, 3, 4, 5, 6, 7, 8, 9, Jack, Queen, or King respectively.
- **Deck** - Has one attribute 'cards' which is an array of type Card which has the size of the constant DECK_SIZE (in the general case 52).
- **Column** - Like the Deck struct - has one attribute 'cards' which is an array of type Card which has the size of constant COLUMN_SIZE (typically 7).
- **Columns** - Has an attribute 'column' which is an array of columns of size NUM_COLUMNS (typically 3).

2.2 Pointer Based

The pointer based implementation has an additional C file '**linkedlist.c**' which has functions to interface with the linkedlist data structure declared in readmymind.h. These include functions to create and add nodes to a linked list as well as retrieve nodes from a linked list. In addition there is a function to free the memory occupied by nodes of a linked list to avoid memory leakage and a function to update the indices of a given linked list.

A linked list implementation by HackerEarth[1] was used to give an idea of how to implement a basic linked list data structure in C.

Aside from the way the data structure is accessed during the trick, the main difference between the two implementations is the way that the user input and

[1] https://www.hackerearth.com/practice/data-structures/linked-list/singly-linked-list/tutorial/

output works. In the pointer based implementation instead of accessing card icons to be displayed using a constant array, a switch statement containing print statements is used. Similarly, instead of storing a constant array of valid inputs and accessing it to validate the user's input, a switch statement is used for input validation. While the implementation of these features is more elegant in the array based implementation, this alternate implementation demonstrates that arrays are not necessary.

The pointer implementation shares a common Card struct with the index based implementation, however the other following structures differ in implementation:

2.2.1 Data Structures

- **Card** - Has two integer attributes:
  - **suit** - a number from 0 - 3 representing either spade, heart, diamond, or club.
  - **rank** - a number from 0 - 12 representing and Ace, 2, 3, 4, 5, 6, 7, 8, 9, Jack, Queen, or King respectively.
- **LinkedList** - Has the following attributes:
  - **card** - Stores a Card attribute that can be thought of as the 'value' of the node (a card in the deck or columns).
  - **next** - A linkedlist pointer to the next node the in the list.
  - **index** - An integer signifying a node's position in its linkedlist - manually set and used so that it would be easier to translate the functions in the index/array based implementation into the pointer implementation.
  - **chosen** - A boolean used when selecting the random 21 cards from the deck of cards to signify whether a particular card has already been selected.
- **Node** - A typedef that allows a LinkedList pointer to be declared using the keyword 'Node'.
- **Deck** - A struct that has one attribute 'node' which is a LinkedList Node. Used to allow 52 nodes to be added, each of which stores a unique playing card.
- **Columns** - A struct that contains three Node attributes called 'first', 'second' and 'third'. Each attribute stores the heads of their respective linkedlists which represent the three columns of the card trick.

3 Testing

4 Conclusion and Evaluation

5 Extensions

- Freeing memory
- Letting user repeat trick
- Colours for cards using ansi codes
- Compare complexity of each solution and show time comparison.
- unit testing

References

[1] Mohd Sanad, Zaki Rizvi, *"Singly Linked List Tutorial"*, HackerEarth, https://www.hackerearth.com/practice/data-structures/linked-list/singly-linked-list/tutorial/, (2016).