Programozói dokumentáció

Menu: This is a simple menu used to navigate between the different functions, using a do-while loop and switch-cases (each case goes to different function).

- 1 (To list all books)
- 2 (to add a new book)
- 3 (To find a specific author/book)
- 4 (Delete old book)
- 5 (Edit the information of an old book)
- 6 (For Readers Functions)
- 9 (Exit)

As we can see over here, we use numbers as inputs, so if the user gives characters, the program reads it as a string and converts it to the number 0 using (int)strtol(str, NULL, 10), and since the default case is when none of these numbers are inputted, it displays that this is an invalid input.

Olvasok.h:

typedef struct reader

- typedef struct reader
- void reader_info()
- void read_file(void)
- void create()
- int number_of_lines()
- void rent2(reader r)
- void rent()

Struct:

Reader: This structure is used to store the readers information. This includes a 4 character long ID, integer type age, the book rented which is used in the rent() function and a name structure which has to subtypes: first name and last name.

olvaso_info_kezeles:

Reader_info:

This is also a menu for the different functions of the program. It has no return value or parameter and works exactly the same as the main menu using a do-while loop and (int)strtol(str, NULL, 10).

Functions:

- 1 (Display all readers)
- 2 (Rented books)
- 3 (Add new reader)
- 4 (Rent a book)

Read_file: This function is used to read each line of a file letter by letter. Its parameter is the filename we want to read (char*filename). If the file was opened successfully, we assign the c character variable the first letter of the file using fgetc. Then until it hasn't reached the end of the file (EOF), we print it and go to the next. After we are done we close the file and return to the menu (as this function has no return value, void).

Create: Adds a new reader by reading from the user information simply using scanf, and storing into a reader r variable. After all data has been read, we print it into the olvasok.txt file and close it.

Number_of_lines: We use this function to count the number of lines of a file. Parameter is filename, and we go through each and every single line of this file, until the end (EOF). While doing so we increase the value of the count variable and return it - 2 (as each file starts with a header and a breakline).

Rent: We will be using 2 different files in this function. First we have the original file which we will be deleted and have its edited information stored in a second temporary file which will be renamed. In this function every line is read read in a buffer variable and stored, and we ask the user to choose a row number (which book to rent) as well a reader ID, and use a do-while loop with 2 conditions. First is to check whether we have reached the end of the file, if so then the (boolean) keep_reading variable is set to false and we exit the loop. Second is to check whether we have reached the row to be modified. If so, then we store the book we want in the (reader) r.book_rented variable and check if the book has been rented before or not. If it hasnt then we modify the row by adding the ID and [R] symbol. Finally, if neither of these conditions are met, then we just simply write the line onto the second file. After exiting this loop, we give the r information to the rent2 function which continues this.

Rent2: This function works very similarly to the rent function, where we have 2 files, one is being read and the other is being modified. The do-while loop is the same, except for the 2nd condition, where we check if the id matches whats read in the buffer, if so write into the 2nd file. In the else part we go to the following line and read its info.

Konyv.h:

- typedef struct date
- int num_lines()
- char* dinamikus_tomb()
- void foglal(char*** ptomb, int szeles)
- void view(void)
- void delete_book(int delete_line)
- void add(int which_line)
- void choose_book()
- void edit()
- void search(void)

Struct:

Date: Structure used for storing date information (év, hónap,nap)

Delete and edit:

Num_lines: This function is exactly the same as number_of_lines. So we use this function to count the number of lines of a file. Parameter is filename, and we go through each and every single line of this file, until the end (EOF). While doing so we increase the value of the count variable and return it - 2 (as each file starts with a header and a breakline).

Delete_book: This function is used to delete books (rows) from the original file. We open 2 files (one in reading mode and the other in writing) where the parameter is the row to be delete, and copy all of the information from one file to the other using a do-while loop with the bool keep_reading variable. When we have reached the end of the file(feof) then we set keep_reading to false and exit the loop. If we have not reached the end, we copy the contents 1 file to the other with the condition that the current line does not match the chosen line, and then we increment the current_line variable. Once we are done we close all files, remove the original file as it is not needed anymore, and rename the new file (to the first files name).

Edit: Over here we use the previously mentioned delete function and the add function (next part). If we reach the line chosen by the user, then we delete it and request the user to enter the information.

View and add book:

Dinamuskis_tomb: (This function has no paramters). We dinamically assign an array (tomb), read a new (ujkar) character from the user, create a new dynamic array with

the modified length, copy the information we have so far into it using a for loop, free the tomb named array, add the new character to ujtomb, add the \0 character as the last element of the array, and increment the new size by 1. Keep doing this until an enter (\n character) is read. Finally, return the dynamically allocated tomb array.

Foglal: This has 2 input parameters. A 2D array, and the width of it (how many strings it is going to contain). We dinamically allocate memory for double pointer uj named variable and set the maximum size to 100 (height). Then we allocate memory for each individual element of the array, read from the user a word, store it inside of the newly allocated memory and move on. Finally we set the parameter to the value of the uj variable.

View: Open file in reading mode. If NULL then there is an error, otherwise we read every line using fgets and print it on the screen with its row number (except for the first 2 as thats the header).

Add: Open file in mode 'a' (writing while perserving old data). (If NULL then there is an error). The exact same method is used here as in delete, where 2 files are used, one being the file storing the information (opened in read mode) and the other being the one modified with the new information. Once we enter the loop, we ask the user for a title, dynamically allocate memory for it using the dinamkius_tomb function print into the file and finally free. Then repeat the same process for the author. For the genre(s), since a book can be of multiple genres, we ask the user for the number of genres they want to add and use the foglal function to allocate memory for a 2D array. Finally for the date we use the a datum type variable, read info from the user and print into file.

Search: In this function we enter a key word, then while its not 9 (which returns back to menu, we check if we havent reached the end of the file, if not, compare the key word with whats being read in the file using strstr. If it matches, then we get it using fgets and print it onto the screen.

Felhasználói dokumentáció

Főmenu: A felhasználó számok beadásával tudja választani a program különböző funkcionalításait. (pl: Az 1-es szám inputra meglehet tekinteni a fájl összes tartalmát). A 9-es szám beadásával kilehet lépni a programból.

Függvények:

2 (to add a new book): Amikor a felhasználó belép az függvénybe, akkor tőle kérdezzük egy könyvnek az információját, ahol a cím és a feladó bármi lehet (szám, betű vagy karakter), aztán megkérdezzük hogy hány zsánert szertne hozzádni. Ezt számnak kell hogy lennie. A zsánerek beadása után, a kiadási dátumora van szükség, amit éééé.hh.nn formátumba kéri a program.

3 (To find a specific author/book): A felhasználó egy kulcsszó beadásával tud a fájl tartalmából keresni.

4 (Delete a book): A törölni kívánt sorszám adásával, törlésre kerül.

5 (Edit/Replace): A delete függvényhez hasonlóan választása kerül a módosítandó sor, aztán a felhasznló megadja az új könyv adatait, pontosan úgy mint az add-nél.

6-os szám beadásával az olvasók menüt lehet elérni.

Abban 4 különböző dolgot tudunk csinálni:

1 (Display all reader): Megmutatja az összes olvasót.

2 (Rented books): megmutatja hogy melyik olvasó, melyik könyvet bérelt.

3 (Add new reader): Egy új olvasó adatait lehet tárolni. Először egy azonosítót ami bármilyen szám lehet 1 és 1000 között. Ez után egy nevet ami bármilyen karakterű lehet, és végül egy kort amit természetesen számnak kell lennie.

4 (Rent a book): Ebbe a függvénybe csak egy könyvet kell választani (annak a sorszámát várja inputnak) és egy tanulói azonosítót.