**----------------------------------------------------------------------------**

**Author: Zhong Zhenghang (Klaus)**

**Date: Oct, 2016 - Dec, 2016**

**----------------------------------------------------------------------------**

**EEE101 C Programming and Software Engineering 1 – Roulette**

1. **Problem statement.**

Design a gamble game called Roulette. Firstly, players could create their account with given name, surname and password, then they could credit their account with money which should be more than 10. Secondly, players could log in their accounts and play the Roulette game, in the game they could choose the amount of bets they would like to place which including 10, 100, 1000 and 10000, then they are asked to input the number of bets they want to place (as many as they like if with enough money), system should be able to judge if players have enough money for their bets and then to simulate the process of placing a ball into the spinning wheel and get a final number. Comparing the number with the bets that players place and change the money in the accounts if players win some money. Then players could choose to have a second try or log out the account or exit the program.

To make the game more interesting. Players could choose many modes of bets, for example, they could choose the 1st region (1 - 12) or all even numbers or the right region numbers (19 - 36), or just choose a single number. And the winning rate is different from 2:1, 3:1 to 37:1.

1. **Analysis**

Input:

Create account:

* A string to represent the surname of account.
* A string to represent the given name of account.
* A string to represent the password of account.
* A number of type **long long** representing the credited money.

Log in:

* A string to represent the surname of account.
* A string to represent the given name of account.
* A string to represent the password of account.
* (optional) A number of type **long long** representing the credited money.
* A number of type of **long long** representing the money of each bet being placed.
* A number of type of **int** representing the number of bets will be placed.
* A serial of numbers of type of **int** representing the numbers players want to bet.
* A character representing the instructions to start again or log out.

End the game:

* An **int** number to quit the program.

1. **Design**

Algorithm:

— 1. Declare a function called **\_create.** This function has a pointer input of type **char** which points to the address of a file and return an integer. This function could create an account and store the relevant information to a binary file.

* 1. Declare two variables namely account and person both in the type of structure entry. The **struct entry** person stores the information entered by users. Then the **struct entry** account stores the account’s information in file one by one to check if accounts in the binary file have the same name to this user, if there exists an account with same name, tell user to change the name; if not, this new account is successfully built up.
  2. Declare a FILE pointer called **fptr** which stores the address of the file opened.
  3. Use the function **scanf()** to store the surname, given name, password and money of a new account a string variable or a variable of type **long long**. Program need to check the return value of **scanf()** to see if players input in a correct way, and the length of string need to be check if accord with requirements, and use the function namely **fflush(stdin)** to clear the standard input buffer.
  4. After store all information in the **struct** namely person, open the file and write person in the final space. Open the file with mode “**ab+**” (users could read and write the binary file optionally) and use the file pointer namely **fptr** to store the address. Importantly, check the value of **fptr** to see if program open the file successfully, if not, exit the program in case cause more damages to the file. If all goes well, use a function namely **fread**() to read a space of memory of the size of **struct entry** in the file pointed by **fptr** and store the data in the account of type of **struct entry**. Then do the reading loop until to the end or find an account with the same name to users’. If existing same account, close the file using function **fclose()** and return number 0 to end the whole function.
  5. In fact, the steps of store password and money are placed after checking the name is unique.
  6. Use the function **fseek**() to set the cursor to the end of the file and wright the whole **struct entry** of person in the file. Finally, close the file using **fclose**().

—2. Declare a function namely **logIn**. This function has a pointer input of type char which points to the address of a file and return an integer **n** which indicates the location (**n** \* **sizeof(struct entry)** ) of user’s information. This function enables users to log in with correct information.

1. Declare two variables of type **int** namely **flag** and **n**. The flag represents the state that names input by users could be find in file, and **n** indicates the information is in the location of after **n** **struct entry**.
2. Declare a file pointer called **fptr** to store the address of a file. A variable of type **long long** to store the money users want to credit, and three arrays of type char to store the surname, given name and password of input information.
3. Like did in function **\_create**. Use function **scanf()** to store the name and password in the three arrays. Then open the file and search through the whole file to see if the information matching the account in file. If they match, log in successfully and users could choose to recharge more money or not and then the **logIn** function closes and return the value of **n+1**. If the name could not match, tell the user no such an account exists and close the file. If the password is not correct, tell the users to have another try and close the file.
4. It’s important that close the file after open it as soon as possible.

—3. Declare a function namely **\_gamble**. This function has two variables input of type **int** and char pointer namely **number** and **filename** separately, and it returns nothing. Players could see the interface only when they log in successfully.

1. Declare a variable called **Bets** of type **int**, which represents the times of bets players want to place.
2. Declare a variable called **flag3** of type **int** to represents the state of playing the game or not.
3. Declare a pointer called **Num** of type of **char**, then create a block of memory space with a size of **Bets** \* the size of **char**.
4. Declare two variables namely **bMoney** and **cMoney** of type of **long long**. And first one represents the money of every bet players want to place and the second one represents the money users would like to credit.
5. Open the binary file with mode **“ab+”** and store the address of file in a pointer. Check if open the file successfully by checking if the pointer equals to **NULL.**
6. To find the location of the user’s information using function of **fseek**() and set the cursor to the location of number (the location) times the size of the structure called entry.
7. To read the information this user who has logged in and store it in a structure declared in this \_**gamble** function. Then close the file.
8. Then show the rules of this game and give users 3 choices of starting the game, recharge more money or log out.
9. If players choose to start the game. Ask them to input the betting money for every bet and how many bets they want to place. Then input the numbers they want to bet with total amounts according to previous input. Always check if players have enough money to bet. Then use the function **random**() to create a random number with time as the seed. Finally get a number and check it with the numbers players bet and change the money of users.
10. About the numbers users could choose. Normally, from 0 to 36 with a winning rate of 37:1. Moreover, number 37, 38 and 39 representing the range of numbers 1 to12, 13 to 24 and 25 to 36 with winning rate of 3:1. Number 40 and 41 representing the range of numbers 1 to 18 and 19 to 36. Number 42 represents the even numbers without 0, number 43 represents the odd numbers with a winning rate of 2:1.
11. If users want to recharge more money. Change the money in according account and return to before interface.
12. With game over and players wanting to end the game, find the according location and refresh the information in file using function **fseek**() and **fwrite**(), then close the file.

—4. Declare a function namely **\_chessboard**. This function has no input variable or return value. It just print the chessboard on the screen and show the betting rules to make it easier for players to understand.

—5. In the **main** function. Two variables of type **int** and char are sent to it. And **argc** represents the number of files transferred to **main(), argv[]** represents the name of file.

1. Declare an array namely filename to store the name of the file which will be built up.
2. Check if an additional argument (filename) has been added to the execution list. If there is a filename copy it to the filename variable contining the default name.
3. Then open the file called filename with mode “**ab**”. Check if open the file successfully.
4. Let users to choose between creating an account and logging in or just quit the game.
5. If log in successfully, start the game. If choose to quit the game, exit the program.
6. **Implementation**

See the C code in file 1508686\_5.c with comments.

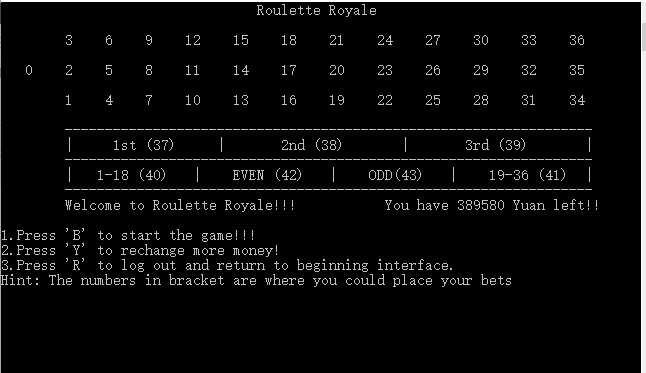
1. **Testing**

During the time I code, I had actually encountered with several problems and find some deficiency aspects with some of them successfully solved others not solves well due to the limitation of time. I will list them below:

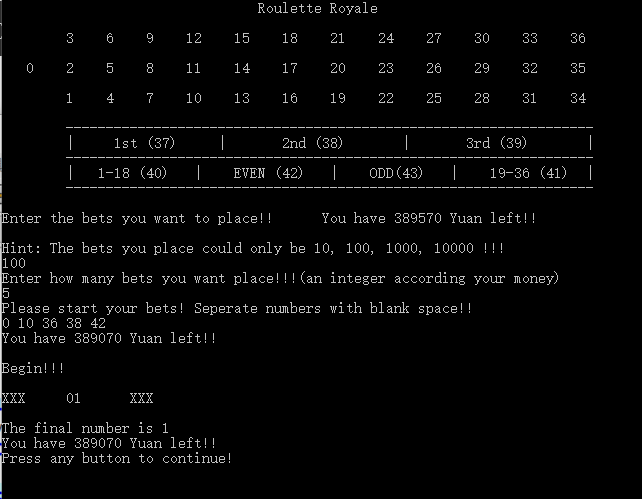
1. One user-unfriendly aspect is that the program require users to input surname and given name to create an account or log in, rather than just only a string of username. However, I have come up a more user-friendly way, more specifically, to store all the input characters in a string and use function **if()** to detect if there is a blank space in the string between two characters, which could tell the program either user just input username or two names. I think it a more user-friendly way but have not much time.
2. A problem that I have solved is about to trace the location of user’ information who has logged in. I initially use function **fseek**(), **fread**() and **strcmp**() to trace the information one by one every time program need it, which wastes much time and make code not tidy enough, therefore, later on I found a better way by return a number n in the function **logIn** which tell the user the **n th** structure in the file stores the information needed, which is a very efficient way.
3. In the game part. The program was designed to have additional functions. The money are fixed with 10, 100, 1000, 10000 RMB of every bet. The total number of bets need to be preset, which is not very user-friendly in my view but is not solved very well. User could bet on numbers from 0 to 43 and numbers after 36 mean an area of numbers, for example 42 stands all the even numbers and 43 stands all the odd numbers. Players are free to place as many bets as they want if with enough money.

The C program was tested by carrying out a set of experiments; and the C program output was verified successfully. For instance,

* 1. When I create an account with name already exists, program would say “Sorry, the same account exists”, and return to the original interface.
  2. When I create an account with unique name but credit a minus money or English character (-1000 or ABC), program will warm “Please try again and make sure the number or correct”.
  3. When I choose the log in mode, program will judge if the name and password I input could match an account in the file, it will tell users to have another try if not match, if log in successfully the game part will start and tell users the balance.
  4. The interface of game is like below:



1. Due to the rules, user can’t start gamble if not press ‘R’ or money less than 10 Yuan. If users press buttons except ‘B’, ‘Y’ and ‘R’, program will warm “Check if your money is less than 10 or gave a wrong instruction!!”
2. When starting the game, players are asked to choose the size of bet, number of total bets and numbers they want choose, just like below:



1. Every time users do not follow the rules and input a number in an incorrect form, the program would give warms and tell them to try again.
2. The random number is produced by simulating the process of place the ball into the spinning wheel and speed of change of numbers are from fast to slow. And finally get a number. Judge if user wins and change the balance.
3. When want to quit, just press ‘R’ to return beginning interface and press ‘3’ to quit the game.