Chris Stark

CSC/CIS - 17A – 48290

Dr.Lehr

10/28/2022

Project 1

Project 1

*\*\*Video Poker – Jacks or Better\*\**

**\*\*THINGS NOT WORKING\*\***

My Random Access Functions and Switch Menu option 2(Show Stats) which uses the random access function are not working correctly you can however look at the non binary file and see that records are being stored correctly, I created a non binary file as well as a binary file to compare the information grabbed from the binary file. The function for Random Access is not grabbing the information it should be grabbing from the binary file. I am writing the struct array (records[i]) the records array consists of a string name with a string size of 4 (I only allowed each user name to be a size of 4 for easier file searching) and 4 different int variables from the structure records array to the binary file, so one whole records[index] should hold 20 bits of information. I tried creating separate variables and grabbing 4 bits at a time to make sure it was grabbing what was expected but I could not get it working. \*\*\*I sat with Elizabeth the lab aid for a couple hours and we tried to troubleshoot it, but we could not get it working\*\*\* I left all the code that was not working and commented it within the checklist and program so you could see what I was working with. I commented out all the function calls that were not working properly so it would not throw the program off for everything else. I want to sit down with you during class hours at some point and discuss my errors with my binary file code.

**Introduction:**

Game Type: Video Poker – Jacks or Better

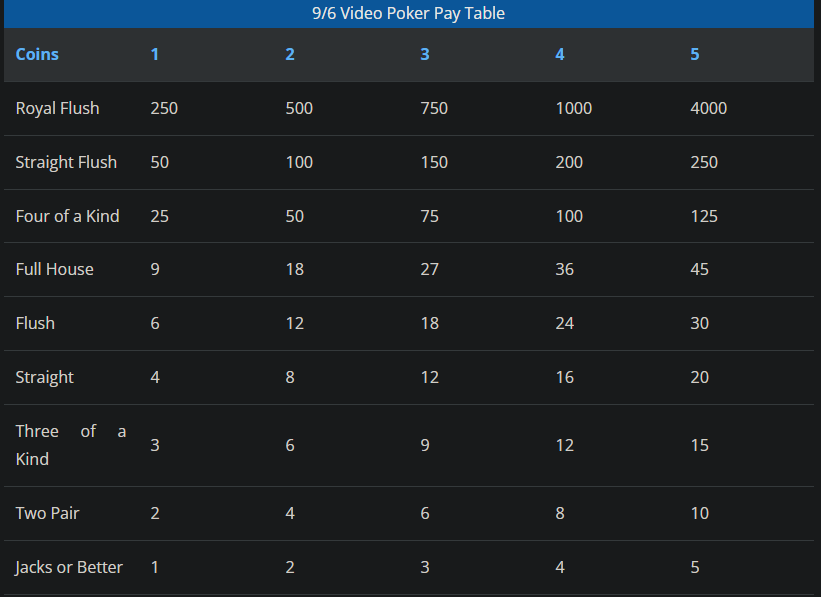
This game is based around the popular casino video poker games. There are multiple versions of video poker, this game is based around the rules for Jacks or Better. Meaning the lowest winning hand is a pair of Jacks, Queens, Kings or Aces. Each game starts with 200 credits, before forcing you to start over. You will be dealt a hand of 5 face/suit cards. You will be asked to enter a bet amount between 1-5 credits. The amount you bet determines the payout amount for each hand. Each hand is allowed to discard once up to all 5 cards. If you choose to discard cards, your choice of cards to discard will be replaced and a new hand will be shown. Once the discard phase is over, the game determines if you have a winning hand. If you are not familiar with poker hands, below is a chart of winnable hands – minus the Jacks or Better pair rules. So, make sure you remember Jacks or Better as a winnable hand as well.

**Winning Hands:**



**Payouts:**

Payouts are determined based off rarity of hand and adjusted based off your initial bet amount. The payouts for each hand are listed from highest to lowest below. (Both images supplied by <https://www.casinoreports.ca/video-poker/rules/>)



**Stats records:**

When you start your firsthand you will be asked for a 4-char sized name to enter. If it is your first time playing your name will be freshly recorded within the game's stat records. If the name you have entered is a previous name that has played your records will be updated as you play. The stats that are recorded will be highest credits won, highest credits lost, highest hands won, highest hands lost. These stats are based off a 200 credit game or until the user decides to quit. In the main menu you are allowed to view your stats.

**Credits:**

Each game starts with 200 credits. You are allowed to play until your credits reach 0, once your credits reach 0 the game is terminated.

**Summary:**

The code is based off chapters 9-12 in the Gaddis Book “C++ From Control Structures through Objects.” My code utilizes structures filled with variables ranging from strings, string arrays, \*string arrays, char, char array, \*char array, int’s and Enumerator’s. Lots of functions were utilized to pass around different tasks, making the main code small and neat. The main \*arrays being passed around are the players poker hand consisting of an array of face cards and an array of suits for the cards. Dynamically allocated arrays are produced when sorting the original poker hand to run easier calculations on winning hands. These arrays are destroyed and never displayed to the user. So the ending hand will always be visible without sorting. I chose to use strings and char arrays for project requirements. The record files that are being read/written consist of line-by-line name, highest credits won, highest credits lost, highest hands won, highest hands lost. I have both non binary and binary files for comparison reasons.

Project lines of code = 1312

**Important Variables:**

Player player ---- Player struct object that will hold all our struct information

player.records ---- holds player name,credits won, credits lost, hands won, hands lost

player.name ---- holds player name

player.credits ---- holds current credits

txtFile ---- text file name for non binary file

binFile ---- binary file name

menu ---- switch input for menu control for user

betAmt ---- users bet amount passed around in functions

index ---- users specified index for binary random access cursor calculation

**Important Function Specific Variables:**

player.hand//a.hand ---- Dynamic allocation for hand size 5 array

player.suit//a.suit ---- Dynamic allocation for hand/suit size 5 array

tempCrd ---- Pointer Dynamic variable with players hand cards that get sorted

tempSut ---- Pointer Dynamic variable that gets sorted with variable(tempCrd)

num1, num2, num3, num4, num5 ---- These variables hold integer values based on each card in hand

**PSEUDO CODE:**

Project 1 - Video Poker \*Jacks or Better\*

Pseudo Code

Variables

---------

FaceCards

Suits

PlayerName

RecordArray

CreditsWon

CreditsLost

HandsWon

HandsLost

CurrentCredits

WinAmount

MenuInput

BetAmount

NonBinFileName

BinFileName

GetLineVariable

Check for files

While(!File.open)

Create NonBinFile

Create BinFile

Read records file for stats = Will need struct array of records & struct stats variables

for(i = 0; i < amount of records; i++)

While(!File.eof)

getLine(Name)

records[i]name = GetLineVariable

getLine(GamesWon)

records[i]GamesWon = GetLineVariable

getLine(GamesLost)

records[i]GamesLost = GetLineVariable

getLine(CreditsWon)

records[i]CreditsWon = GetLineVariable

getLine(CreditsLost)

records[i]CreditsLost = GetLineVariable

do

Switch(menu)

COUT menu with some game info - Video poker type/starting credits

1 Deal hand

2 Show stats

3 Show current credits

4 Quit

Default

COUT Please enter correct menu option

while(!4 && CurrentCredits !=0)

if(CurrentCredits == 0)

COUT user is out of credits

else

COUT user Quit the program

Menu 1

COUT Game info/type Jacks or Better

COUT bet amount procedure

COUT Ask user to place bet amount

while(!1-5)

COUT ask user to place bet amount

CurrentCredits -= betamount

COUT face/suit cards

COUT ask user if they want to discard cards

if(yes)

COUT ask which cards to replace

while(!1-5 && duplicate)

COUT ask correct cards to replace

COUT deal new face/suit cards

else

Check cards for win conditions

if(no win)

0 credits earned

hand lost++

credits lost += bet amount

else

Determine winning hand

WinAmount determined off winningHand + betAmount

CurrentCredits += WinAmount

Credits won += WinAmount

Handswon++

COUT win/lost hand

COUT win/lost amount

Write Records to file = Will need struct array of records & struct stats variables

for(i = 0; i < amount of records; i++)

records[i]name

records[i]CreditsWon

records[i]CreditsLost

records[i]HandsWon

records[i]HandsLost

Kick back to main menu/ClearScreen

Menu 2

Read records from file = Will need struct array of records & struct stats variables

for(i = 0; i < amount of records; i++)

While(!File.eof)

getLine(Name)

records[i]name = GetLineVariable

getLine(GamesWon)

records[i]GamesWon = GetLineVariable

getLine(GamesLost)

records[i]GamesLost = GetLineVariable

getLine(CreditsWon)

records[i]CreditsWon = GetLineVariable

getLine(CreditsLost)

records[i]CreditsLost = GetLineVariable

COUT ask for PlayerName

for(i = 0; i < amount of records; i++)

check records[i]name

if(records[i]name == PlayerName)

COUT records[i]name

COUT records[i]CreditsWon

COUT records[i]CreditsLost

COUT records[i]HandsWon

COUT records[i]HandsLost

else

COUT PlayerName does not exist

Kick back to main menu/ClearScreen

Menu 3

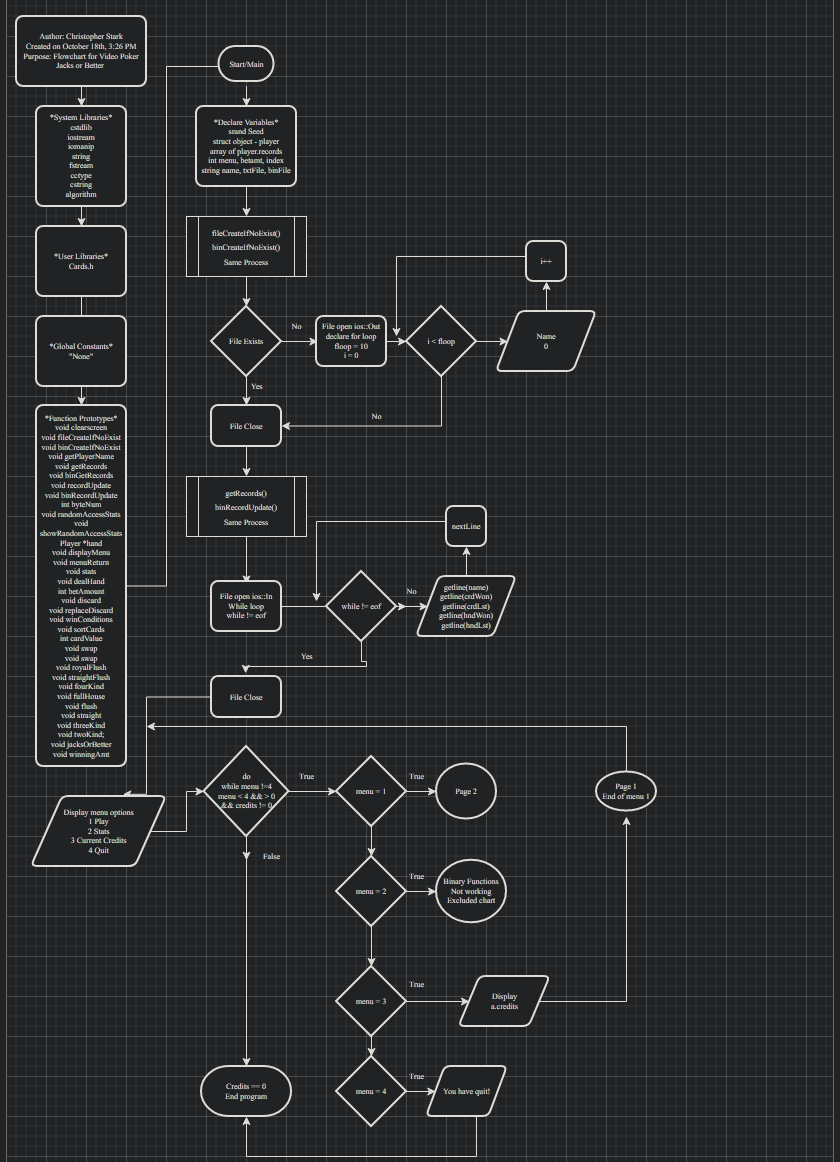
COUT display CurrentCredits

Menu 4

Quit program

**Flow Chart:**

Page 1:



Page 2: A picture containing text, blackboard

Description automatically generated

Page 3: Diagram

Description automatically generated