**Project 1**

Title

**Slots Game**

Course

**CSC-17A**

Section

**48130**

Due Date

**24 October 2014**

Author

**Najera, Enrique**

**Introduction**

Title: Slots Game

The slots game is a game in which a player inserts a monetary bet, “rolls” the machine, and depending on the numbers after the roll, the player either wins three, five or ten times their bet; however, there is a chance the player loses their bet.

The player starts off with $100 and goes on from there. The machine has three choices for the player: roll, buy cheats, see a reference guide, or simply leave the game.

Slot rolls cost $2 by default, but the administrator can change it. This option depends whether the player gains riches or lose everything they have.

The player has an option to buy cheats to increase their winnings, or losses. The player can choose to double or triple their wins, get a free chance on the house (value of $5), or increase their winning chance.

The reference guide shows how much a player can win or lose depending on the roll, if no cheats are activated.

The administrator can review or clear the player stats and/or the machine rolls, change the administrator password, use the revenue calculator, and change the price to roll. The administrator menu can be accessed from the player menu by entering the password (default is 12345).

Gamestate Notes (for admin):

0 = Error

10 = Loss

11 = Win

21 = Double wins activated

22 = Triple wins activated

23 = Free chance activated

24 = Increase chance of winning activated

Note about ‘Increase chance of winning’: Purposely doesn’t do anything so the admin gets a free 100 and the user feels happy about adding more money since their chance of winning “increases.”

Extra: if the user enters a bad bet amount, the admin still gets the roll money; in other words, no refunds.

The game is over once the player runs out of money, not enough money to roll, or simply exits the game.

**Summary**

Project size: 739 lines

Variables: 26 (excluding for-loop variables)

Constructs utilized: 27 (from all chapters)

My project meets the criteria for a first project because it uses all the constructs we have learned thus far and went a bit further, but not too much. I say it was a bit challenging since I had no idea how I could’ve implemented all the constructs learned into a slot game. I had to then add as many features as possible to use the constructs. Figuring that out was the biggest challenge for me. I also had a problem with reading files. I could only read the files once and then I would get “error opening file” if I ever want to open it again. I solved this by making two separate functions for the separate files and I could open them again and again. This project took me about 5 days to fully complete (excluding the write up and flowchart). I would’ve loved to use separate hpp and cpp files for a cleaner looking project, but definitely next time, since it will be a requirement.

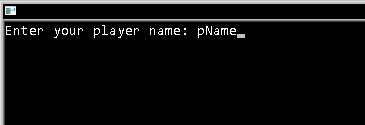
**Description**

I basically used random numbers from 3 to 7 and if statements to get the solution I wanted. Then I based every construct off those numbers.

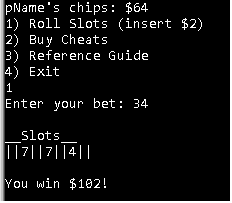
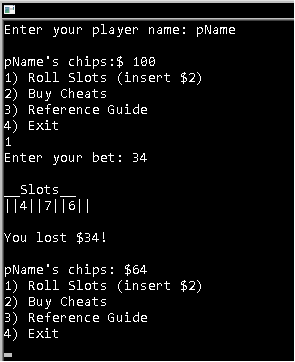
Some basic inputs that have outputs would be the player’s name, bet (if won, without cheats, multiply by 3, 5 or 7. If lost, without cheats, output chips lost), menus, player input errors, statistics of the machine and/or player, and the revenue calculator.

The player’s name would be displayed alongside his/her chip amount; updated after every turn.

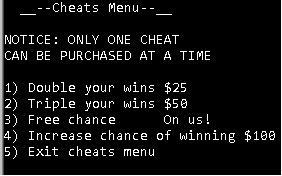
Input player’s name Output player’s name



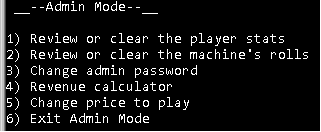
Losing bet Winning bet



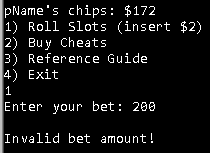
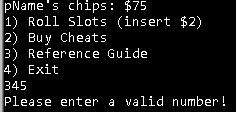
Player Menu Cheats Menu (key 2 on player menu)

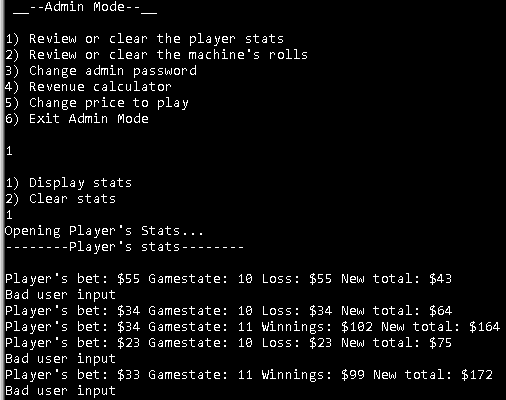


Administrator Menu (correct password on player menu)

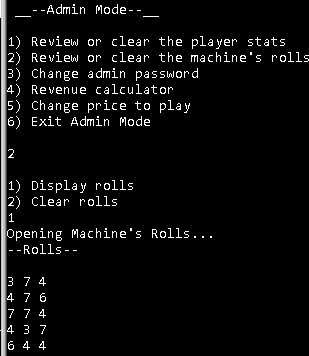


Player input error (menu) Player input error (bet more than actual chips)

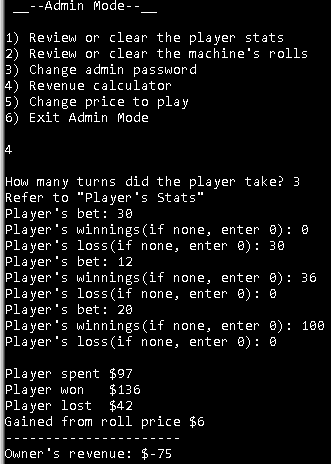


Player Stats (admin mode)

Machine Roll Stats (admin mode)



Revenue Calculator (admin mode)



Basic Flowchart

B

A

No global variables and constants

Libraries

iostream, iomanip, cstdlib, ctime, fstream, string

Global Comments

Name

Date

Purpose

FALSE

End Game

Start choice 1, 2, or 3

TRUE

Output decisions and prompt for p.choice

p.chips != 0

p.choice != 4

B

Output pName and p.chips

Prompt for pName

Declare Variables

aPass, min[], max[], aPWarry[],

inC, insert, adminPW, \*calc, numTrn, disOclr, winTbl[][]

Start method main

A

Function Prototypes

randnum, cheatMenu, adminMenu, newPW, getC, getStats, dispPStats, dispMStats, getMacrll

Pseudo Code

*Prompt for player’s name*

*Display player’s name alongside chips*

*Get and decrypt admin password*

*While player chips > 0 or choice != 4*

*Get user choice*

*If choice = 1*

*Get player’s bet*

*Start randnum and return slot value*

*If 2 values are equal*

*Multiply bet by 3*

*Else if all values are equal*

*Multiply bet by 5*

*Else if values all 7*

*Multiply bet by 10*

*Else no values are equal*

*Subtract bet from total chips*

*Else if choice = 2*

*Activate cheats menu*

*Get player’s choice*

*Case 1*

*Double wins*

*Subtract 25 from total chips*

*Case 2*

*Triple wins*

*Subtract 50 from total chips*

*Case 3*

*Free chance*

*Case 4*

*Subtract 100 from total chips*

*Else if choice = 3*

*Display the reference guide*

*Else if choice = 4*

*Exit the program*

*Else if choice = admin password*

*Activate admin menu*

*Case 1*

*Prompt for display or clear player stats*

*If display*

*Open stats file and display stats*

*If clear*

*Clear the stats file*

*Case 2*

*Display or clear machine’s rolls*

*If display*

*Open rolls file and display rolls*

*If clear*

*Clear the rolls file*

*Case 3*

*Prompt for new password*

*Case 4*

*Prompt for turns*

*Prompt for bet, winnings, and loss*

*Add up the values separately*

*Display player spent, won, lost, gain from roll price*

*Display revenue ((bet + roll price) – loss) – win)*

*Case 5*

*Prompt for new price to roll*

*Default*

*Exit*

*Else*

*Output error*

No classes therefore no UML available.

Major Variables

|  |  |  |  |
| --- | --- | --- | --- |
| **Type** | **Variable Name** | **Description** | **Location (lines based on importance)** |
| Short | p.choice | Gets the player’s main menu choice | Struct Player and main(), line 140 |
| Integer | p.chips | Holds the player’s chip amount | Struct Player and main(), line 125 |
|  | p.bet | Holds the player’s bet amount | Struct Player and main(), lines 38, 271 |
|  | c.mult2 | Multiplies winnings/losses by 2 | Struct Cheats and main(), lines 53, 206, 219, 309, 326, 343, 361 |
|  | c.mult3 | Multiplies winnings/losses by 3 | Struct Cheats and main(), lines 54, 207, 218, 309, 326, 343, 361 |
|  | m.s1, m.s2, m.s3 | Holds values for the slot roll | Struct Machine and main(), lines 44-46, 296-298, 303, 306, 323, 581 |
| String | pName | Holds the player’s name | Main(), lines 121, 125, 210, 222, 234, 246, 386 |

Concepts, Syntax, and Keywords

|  |  |
| --- | --- |
| **Type** | **Location** |
| Pointer | Lines 30, 84, 308, 589, 595-597 |
| Structure | Lines 34, 42, 51, 61, 97-118 |
| Array in a structure | Line 66 |
| fstream | Lines 73, 592, 607, 671, 702, 732 |
| ifstream | Line 74 |
| getline | Line 129 |
| Read from file | Lines 136-141, 678-691, 709-722 |
| Write to file | Lines 416-418, 436-438, 594-599, 612-621, 626-635, 640-642, 647-656, 661-663, 735-737 |
| 2-dimensional array | Lines 89, 170, 182, 194 |
| Dynamic arrays | Line 465, 530 |
| Array of a structure | Line 121-125 |

References

I used the book and previous homework to help me understand how to properly use the concepts and their syntax. I did search some things online like what is functions with structures and arrays of structures but I couldn’t find what I was actually expecting; then I found them in the book and I only used the book and homework ever since.

Program

//System Libraries

#include <iostream>

#include <iomanip>

#include <cstdlib>

#include <ctime>

#include <fstream>

#include <string>

//Namespaces

using namespace std; // iostream

//Function Prototypes

int randnum(int, int); // Generate a random number for slot values

void cheatMenu(); // Display the cheat menu

void adminMenu(); // Display 'admin mode' menu

int newPW(int []); // New password config.

int getC(); // Get input for the cheat menu

void getStats(int, int, int, int); // Get user stats(current $, gamestate, bet, winnings)

void dispPStats(); // Display user stats

void dispMStats(); // Display machine stats

void getMacrll(int \*, int \*, int \*); // Get machine roll values

//Structures

//Hold values for the player

struct Player

{

int chips; // Hold player's current balance

short choice; // Hold player's choice

int bet; // Hold player's bet amount

};

//Hold values for the slot machine

struct Machine

{

int s1; // Machine Slot 1

int s2; // Machine Slot 2

int s3; // Machine Slot 3

int price; // Price to roll

};

//Hold values for cheats

struct Cheats

{

int mult2; // Multiply winnings by 2

int mult3; // Multiply winnings by 3

int free; // Free spin

int chance; // Increase chance of winning

bool isFree; // disable or enable 'free'

};

//Holds values for revenue calculation

struct CRev

{

int plBet; // Player's bet

int plWin; // Player's win

int plLoss; // Player's loss

int mult[10]; // Calculate multiple things

};

//Start method main

int main(int argc, char \*argv[])

{

//Declare Variables

fstream clStat; // Clear stat contents from files

ifstream passF; // Open, Close, and Read password from file

string pName; // Holds the player's name

int aPass; // Get chars from admin.sec

int min[3] = {3,3,3}; // Minimum of slots is 3

int max[3] = {7,7,7}; // Maximum of slots is 7

int aPWarry[1] = {0}; // Holds admin password

int inC = 0; // Choose the cheat

int insert = 0; // Calculate insert for revenue

int adminPW; // Holds new admin password

int \*calc = NULL; // Dynamically allocate an array for a calculator

short numTrn = 0; // Hold number of turns of player for calc

short disOclr = 0; // Display or clear stats in Admin Mode

//Show winning examples

int winTbl[7][3] = {{3,3,4},

{7,7,5},

{6,3,3},

{3,3,3},

{4,4,4},

{5,5,5},

{7,7,7}};

Player p;

p.chips = 100; // Start player with $100

p.choice = 0; // Init choice to 0

p.bet = 0; // Init bet to 0

Machine m;

m.s1 = 0; // Init slot1 to 0

m.s2 = 0; // Init slot2 to 0

m.s3 = 0; // Init slot3 to 0

m.price = 2; // Default price to roll $2

Cheats c;

c.mult2 = 1; // Multiply winnings by 2. Value '1' = cheat off

c.mult3 = 1; // Multiply winnings by 3. Value '1' = cheat off

c.free = 0; // Free chance. Value '0' = cheat off

c.chance = 0; // Increase winning chance. Value '0' = cheat off

c.isFree = false; // Disable or enable 'free'.

CRev cr;

cr.plBet = 0; // Init bet to 0

cr.plWin = 0; // Init win to 0

cr.plLoss = 0; // Init loss to 0

//Calculate Revenue more than once

CRev multCal[3] = {

{0,0,0}, // INIT vars

{0,0,0}, // INIT vars

{0,0,0} // INIT vars

};

//Get player's name

cout<<"Enter your player name: ";

getline(cin, pName);

cout<<endl;

//Display players chips. Start $100

cout<<pName<<"'s chips:$ "<<p.chips<<endl;

//Get and decrypt admin password

passF.open("admin.sec");

passF.read(reinterpret\_cast<char \*>(aPWarry), sizeof(aPWarry)) >> aPass;

passF >> aPass;

adminPW = aPass;

passF.close();

//While p.choice != 3 || p.chips != 0

while(true)

{

//Get user choice

cout<<"1) Roll Slots (insert $"<<m.price<<")\n2) Buy Cheats\n3) Reference Guide\n4) Exit"<<endl;

cin>>p.choice;

//If player chose to exit (4)

if(p.choice == 4)

{

//Display final $ amount

cout<<"\nExiting Slots with $"<<p.chips<<endl;

break;

}

//Show ref. guide

else if(p.choice == 3)

{

cout<<" \_\_--Reference Guide--\_\_"<<endl;

cout<<"\n3x your bet by getting two of the same values"<<endl;

cout<<" Examples"<<endl;

//Display contents of 2d array winTbl

for(int x = 0; x <= 2; x++)

{

for(int y = 0; y <= 2; y++)

{

cout<<setw(4)<<winTbl[x][y]<<" ";

}

cout<<endl;

}

cout<<"\n5x your bet by getting three of the same values (not 7)"<<endl;

cout<<" Examples"<<endl;

//Display contents of 2d array winTbl

for(int x = 3; x <= 5; x++)

{

for(int y = 0; y <= 2; y++)

{

cout<<setw(4)<<winTbl[x][y]<<" ";

}

cout<<endl;

}

cout<<"\n10x your bet by getting Jackpot (7 7 7)"<<endl;

cout<<" Example"<<endl;

//Display contents of 2d array winTbl

for(int x = 6; x <= 6; x++)

{

for(int y = 0; y <= 2; y++)

{

cout<<setw(4)<<winTbl[x][y]<<" ";

}

cout<<endl;

}

cout<<"\nLose your bet if no numbers are equal\n"<<endl;

}

//If player wants to buy a cheat

else if(p.choice == 2)

{

cout<<"\nCheats\n"<<endl;

cheatMenu(); // Call cheatMenu to display cheats

inC = getC(); // Call getC to get user input

switch(inC)

{

case 1:

//Double wins chosen

cout<<"Double your wins purchased\n"<<endl;

p.chips -= 25; // Take $25

c.mult2 = 2; // Start 2x multiplier

c.mult3 = 1; // Reset mult3 -cheats only used once

getStats(p.chips, 21, 0, 25); // Call getStats(current $, gamestate, bet, cost)

cout<<pName<<"'s chips:$ "<<p.chips<<endl;

break;

case 2:

//Triple wins chosen

cout<<"Triple your wins purchased\n"<<endl;

p.chips -= 50; // Take $50

c.mult3 = 3; // Start 3x mulitplier

c.mult2 = 1; // Reset mult2 -cheats only used once

getStats(p.chips, 22, 0, 50); // Call getStats(current $, gamestate, bet, cost)

cout<<pName<<"'s chips:$ "<<p.chips<<endl;

break;

case 3:

//Free chance chosen

cout<<"Free chance!\n"<<endl;

if(c.free == 1){cout<<"Free chance is used up!\n"; break;} // If free chance already

// used up, display message

c.free = 1; // 'free' value = 1 to enable cheat

c.isFree = true; // If free chosen, true

getStats(p.chips, 23, 5, 0); // Call getStats(current $, gamestate, bet, cost)

cout<<pName<<"'s chips:$ "<<p.chips<<endl;

break;

case 4:

//Better winning chance chosen

cout<<"Increase chance of winning purchased\n"<<endl;

p.chips -= 100; // Take $100

c.mult2 = 1; // Reset mult2 -cheats only used once

c.mult3 = 1; // Reset mult3 -cheats only used once

getStats(p.chips, 24, 0, 100); // Call getStats(current $, gamestate, bet, cost)

cout<<pName<<"'s chips:$ "<<p.chips<<endl;

break;

//If invalid input, break

default: break;

}//End switch for cheats

}//End choice 2 (cheats)

//If player chose to roll (1)

else if(p.choice == 1)

{

//If not enough money for a roll

if(p.chips < m.price)

{

cout<<"You do not have enough money to roll!\n"

<<"Exiting with $"<<p.chips<<endl;

break;

}

p.chips -= m.price; // Delete price of 1 roll

//Get player's bet if !'free'

if(c.isFree == false)

{

cout<<"Enter your bet: ";

cin>>p.bet;

}

//If free chance

else if(c.free == 1)

{

//bet a free $5

p.bet = 5;

c.isFree = false; //Make sure cheat only used once

}

//If bet value more than player's $, bet < 0, or bet is 0

if(p.bet > p.chips || p.bet < 0 || p.bet == 0 )

{

//Display invalid amount detected

cout<<"\nInvalid bet amount!\n"<<endl;

getStats(0, 0, 0, 0); // Call getStats(current $, gamestate, bet, winnings)

}

//If good input and still has $

else

{

//Generate random number seed @ 0 seconds

srand(time(0));

//Set slot value amounts

m.s1 = randnum(min[0], max[0]); // Slot 1

m.s2 = randnum(min[1], max[1]); // Slot 2

m.s3 = randnum(min[2], max[2]); // Slot 3

getMacrll(&m.s1, &m.s2, &m.s3); // Get slot values for stats

//Output slot values

cout<<"\n\_\_Slots\_\_\n||"<<m.s1<<"||"<<m.s2<<"||"<<m.s3<<"||"<<endl;

//If 2 slots are equal

if(m.s1 == m.s2 || m.s2 == m.s3 || m.s1 == m.s3)

{

//Add 3x the bet to player's balance and add any cheats

p.chips += ((p.bet \* 3) \* c.mult2 \* c.mult3);

//Tell player they won

cout<<"\nYou win $"<<((p.bet \* 3) \* c.mult2 \* c.mult3)<<"!"<<endl;

getStats(p.chips, 11, p.bet, ((p.bet \* 3) \* c.mult2 \* c.mult3)); // Call getStats(current $, gamestate, bet, winnings)

//Reset cheats

c.mult2 = 1;

c.mult3 = 1;

cout<<endl; //For console readability

}

//If all slots equal

else if(m.s1 == m.s2 == m.s3)

{

//Add 5x the bet to player's balance and add any cheats

p.chips += ((p.bet \* 5) \* c.mult2 \* c.mult3);

//Tell player they won

cout<<"\nYou win $"<<((p.bet \* 5) \* c.mult2 \* c.mult3)<<"!"<<endl;

getStats(p.chips, 11, p.bet, ((p.bet \* 5) \* c.mult2 \* c.mult3)); // Call getStats(current $, gamestate, bet, winnings)

//Reset cheats

c.mult2 = 1;

c.mult3 = 1;

cout<<endl; //For console readability

}

//If 777

else if (m.s1 == 7 && m.s2 == 7 && m.s3 == 7)

{

//Add 10x the bet to player's balance and add any cheats

p.chips += ((p.bet \* 10) \* c.mult2 \* c.mult3);

//Tell player they won

cout<<"\n\_\_--!Jackpot!--\_\_"<<endl;

cout<<"You win $"<<((p.bet \* 10) \* c.mult2 \* c.mult3)<<"!"<<endl;

getStats(p.chips, 11, p.bet, ((p.bet \* 10) \* c.mult2 \* c.mult3)); // Call getStats(current $, gamestate, bet, winnings)

//Reset cheats

c.mult2 = 1;

c.mult3 = 1;

cout<<endl; //For console readability

}

//If didn't win, lost

else

{

//Subtract bet and any cheats from balance

p.chips -= (p.bet \* c.mult2 \* c.mult3);

//Tell player they lost

cout<<"\nYou lost $"<<(p.bet \* c.mult2 \* c.mult3)<<"!"<<endl;

getStats(p.chips, 10, p.bet, (p.bet \* c.mult2 \* c.mult3)); // Call getStats(current $, gamestate, bet, winnings)

//Reset cheats

c.mult2 = 1;

c.mult3 = 1;

cout<<endl; //For console readability

}

//If no more money in balance

if(p.chips <= 0)

{

//Tell player they have no more money

cout<<"You have no more chips in your balance!"<<endl;

getStats(p.chips, 10, 0, 0); // Call getStats(current $, gamestate, bet, winnings)

//Exit loop, end game

break;

}

//Output player's current balance

cout<<pName<<"'s chips: $"<<p.chips<<endl;

}

}//End of choice = 1 (roll slot)

//If admin mode

else if (p.choice == aPWarry[0])

{

adminMenu(); // Display admin menu

inC = getC(); // Get admin's choice

switch(inC)

{

case 1:

//Display or clear player stats chosen

cout<<"1) Display stats\n2) Clear stats"<<endl;

cin>>disOclr;

//If want to display stats

if(disOclr == 1)dispPStats();

//If want to clear stats

else if(disOclr == 2)

{

//Delete contents of player.stats file

clStat.open("player.stats", ios::out);

clStat << "--------Player's stats--------\n"<<endl;

clStat.close();

cout<<"Stats successfully cleared!\n"<<endl;

}

disOclr = 0; //Reset

break;

case 2:

//Review or clear the machine's rolls chosen

cout<<"1) Display rolls\n2) Clear rolls"<<endl;

cin>>disOclr;

//If want to display rolls

if(disOclr == 1)dispMStats();

//If want to clear rolls

else if(disOclr == 2)

{

//Delete contents of mach.rol file

clStat.open("mach.rol", ios::out);

clStat << "--Rolls--\n"<<endl;

clStat.close();

cout<<"Rolls successfully cleared!\n"<<endl;

}

disOclr = 0; //Reset

break;

case 3:

//Change admin password chosen

cout<<"Enter the new password(5 numbers): ";

cin>>adminPW;

aPWarry[0] = adminPW; // New password insert to array

newPW(aPWarry); // Encrypt and send password to file

break;

case 4:

//Revenue calculator chosen

cout<<"How many turns did the player take? ";

cin>>numTrn;

insert = 0; // INIT insert value for calculation

insert = numTrn \* m.price; // Insert = number of rolls \* roll price

numTrn \*= 3; // 3x numTrn to include bet, wins, losses

//Dynamically allocate array

calc = new int[numTrn];

cout<<"Refer to \"Player's Stats\""<<endl;

//Start input of values

for(int i = 0; i < numTrn ;i++)

{

cout<<"Player's bet: ";

cin>>calc[i];

i++;

cout<<"Player's winnings(if none, enter 0): ";

cin>>calc[i];

i++;

cout<<"Player's loss(if none, enter 0): ";

cin>>calc[i];

}

//Add up all the values

for(int j = 0; j < numTrn; j++)

{

cr.plBet += calc[j];

j++;

cr.plWin += calc[j];

j++;

cr.plLoss += calc[j];

}

//Output values

cout<<"\nPlayer spent $"<<cr.plBet<<endl;

cout<<"Player won $"<<cr.plWin<<endl;

cout<<"Player lost $"<<cr.plLoss<<endl;

cout<<"Gained from roll price $"<<insert<<endl;

cout<<"----------------------"<<endl;

//Output $ gained (bets & insert - loss = gained) - $ given

cout<<"Owner's revenue: $"<<((cr.plBet + insert) - cr.plLoss) - cr.plWin<<"\n"<<endl;

//Delete contents of array for 2nd use

for(int k = 0; k < numTrn; k++)

{

calc[k] = 0;

}

//Free dynamic memory

delete [] calc;

calc = NULL;

break;

case 5:

//Change price to play chosen

cout<<"Current price to play: $"<<m.price<<endl;

cout<<"Enter the new price to play: $"<<endl;

cin>>m.price;

break;

default: cout<<"Goodbye!"<<endl;break;

}

}

//If choice not 1-3

else

{

//Display error message

cout<<"Please enter a valid number!"<<endl;

getStats(0, 0, 0, 0); // Call getStats(current $, gamestate, bet, winnings)

}

}//End of game loop

//Free dynamic memory in case of a fail

delete [] calc;

calc = NULL;

//Exit main program

system("PAUSE");

return 0;

}//End method main

//Start method randnum

int randnum(int min, int max)

{

//Declare function variables

//Keeps number between 3 & 7

int randN = min + rand() % ((max + 1) - min);

//Return the random number

return randN;

}//End method randnum

//Start cheatMenu display menu

void cheatMenu()

{

cout<<" \_\_--Cheats Menu--\_\_ "<<endl;

cout<<"\nNOTICE: ONLY ONE CHEAT\nCAN BE PURCHASED AT A TIME\n"<<endl;

cout<<"1) Double your wins $25"<<endl;

cout<<"2) Triple your wins $50"<<endl;

cout<<"3) Free chance On us!"<<endl;

cout<<"4) Increase chance of winning $100"<<endl;

cout<<"5) Exit cheats menu"<<endl;

}//End cheatMenu

//Start adminMenu display admin mode

void adminMenu()

{

cout<<" \_\_--Admin Mode--\_\_ "<<endl;

cout<<endl;

cout<<"1) Review or clear the player stats"<<endl;

cout<<"2) Review or clear the machine's rolls"<<endl;

cout<<"3) Change admin password"<<endl;

cout<<"4) Revenue calculator"<<endl;

cout<<"5) Change price to play"<<endl;

cout<<"6) Exit Admin Mode"<<endl;

cout<<endl;

}

//Start getC user input

int getC()

{

//Declare Function Variables

int inC = 0;

//Prompt for user input

cin>>inC;

cout<<endl; //For console readability

return inC;

}//End getC

//Start getMacrll open file for output

void getMacrll(int \*s1, int \*s2, int \*s3)

{

//Declare function variables

fstream macrll; // Machine's rolls

//Open mach.rol file for output

macrll.open("mach.rol", ios::out | ios::app);

macrll << \*s1 << " "

<< \*s2 << " "

<< \*s3 << " "

<<"\n";

macrll.close();

}//End getMacrll

//Start getStats get values from gameplay

// current $,gamestate, bet , winnings

void getStats(int chip, int gameS, int bet, int win)

{

//Declare function variables

fstream pStats; // Player's stats

//If lost, display 'loss'

if(gameS == 10)

{

pStats.open("player.stats", ios::out | ios::app);

pStats << "Player's bet: $"

<< bet

<< " Gamestate: "

<< gameS

<< " Loss: $"

<< win

<<" New total: $"

<< chip <<"\n";

pStats.close();

}

//If cheating, display 'Cheat price'

else if(gameS > 20)

{

pStats.open("player.stats", ios::out | ios::app);

pStats << "Player's bet: $"

<< bet

<< " Gamestate: "

<< gameS

<< " Cheat price: $"

<< win

<<" New total: $"

<< chip <<"\n";

pStats.close();

}

//If bad bet amount detected

else if(chip == 0 && gameS == 0 && bet == 0 && win == 0)

{

pStats.open("player.stats", ios::out | ios::app);

pStats << "Bad user input\n";

pStats.close();

}

//If won, display 'winnings'

else if(gameS == 11)

{

pStats.open("player.stats", ios::out | ios::app);

pStats << "Player's bet: $"

<< bet

<< " Gamestate: "

<< gameS

<< " Winnings: $"

<< win

<<" New total: $"

<< chip <<"\n";

pStats.close();

}

//If didn't win, didn't lose, no bad input then machine error

else

{

pStats.open("player.stats", ios::out | ios::app);

pStats << "--!MACHINE\_ERROR\_DETECTED!--\n";

pStats.close();

}

}//End getStats

//Start dispPStats display values from getStats

void dispPStats()

{

//Declare function variables

fstream stats;

char ch; //Gets chars from a file

//Review the last player's stats chosen

cout<<"Opening Player's Stats..."<<endl;

//Open the file

stats.open("player.stats", ios::in);

//If file open success

if(stats)

{

stats.get(ch); // Get chars

while(stats)

{

cout<<ch; // Display chars

stats.get(ch); // Get next char

}

stats.close(); // Close file

}

//If file open fail, display error message

else

cout<<"Error opening stats!"<<endl;

cout<<endl;

}//End dispPStats

//Start dispMStats display values from getMacrll

void dispMStats()

{

//Declare function variables

fstream stats;

char ch;// Gets chars from a file

//Review the last player's stats chosen

cout<<"Opening Machine's Rolls..."<<endl;

//Open the file

stats.open("mach.rol", ios::in);

//If file open success

if(stats)

{

stats.get(ch); // Get chars

while(stats)

{

cout<<ch; // Display chars

stats.get(ch); // Get next char

}

stats.close(); // Close file

}

//If file open fail, display error message

else

cout<<"Error opening rolls!"<<endl;

cout<<endl;

}//End dispMStats

//Start newPW new admin password

int newPW(int pW[])

{

fstream pw;

//Encrypt password to file

pw.open("admin.sec", ios::out | ios::binary);

pw.write(reinterpret\_cast<char \*>(pW), sizeof(pW));

pw.close();

}//End newPW