Roulette Game

Kevin Rivera

CSC 5

Fall 2017

47472

**Introduction**

Title: Roulette

Roulette is a casino game where players choose to place bets on a single number, various grouping of numbers, the colors red or black, whether the number is odd or even, or if the numbers are high (19-36) or low (1-18). To determine a winning number and color, the wheel is spun on a spinning platform with wedges for the roulette ball to land in. The ball is to eventually land on a number of a certain color (black or red) to determine the outcome.

The betting system on this game is simple. If a player bets on a single number—also known as straight bet—the player, then has a chance for a 35/1 payoff. The other bets only pay a 1/1 payoff. These bets include betting on red numbers, black numbers, even numbers, or odd numbers. In a traditional game of Roulette, the player can choose on a group of numbers, columns, and so forth. However, that feature is not present in this program.

**How to Play Roulette**

**Object of the Game**

Bet on a category and hope the ball lands on that number.

**Betting**

Roulette is a game based on solely betting. It’s easy to pick up, and it can be easy to make money and lose money.

1. *Straight Bet* – In this type of bet, presented as Single Number in the program, the player bets on a single number. If the ball lands on that one number, the player receives a 35/1 payoff.
2. *Odd/Even Bet* – This bet offers a 50/50 chance for the player to land the ball on their desired number. The player can choose from even or odd numbers. If the ball lands on the correct category of numbers, the player receives a 1/1 payoff.
3. *Red/Black Bet* – Like the Odd/Even Bet, this plays the same role. However, with this bet, the player bets on either red or black numbers. If the ball lands on the correct category of numbers, the player receives a 1/1 payoff.

**Summary**

Project size: 312 lines

Number of variables: 17

The program consists mainly off *if* statements to check if the player has won a game. The program’s fundamental key is to generate a random number within the number limit to see if the player has won the game. In this version of the game, the game now has two new betting options, black and red numbers. Along with that, a balance system was added to the program. With the latest addition, I was able to program for the player to keep playing if they have sufficient funds to keep playing.

The previous iteration of the game was written mostly without functions; instead, this version was written with plenty of functions to keep the main function shorter than before. When programming the program, I had to look up multiple concepts we did not cover in class; or, at least I don’t remember.

Algorithm library – This library is used for the *find* function which finds a position inside of an array.

Iterator library – This library is used for the *begin* and *end* functions also used to find a position inside of an array. However, this tells the program where to stop within the array.

Goto function – This function is probably not the best technique, but it does the job. It simply jumps back into the program wherever I told the program to go to. As someone said, it’s an evil way of using this function, as it is not it’s purpose.

**Pseudocode and Flowchart**

|  |  |
| --- | --- |
| *Opening comments*  *Bring in system libraries*  *Declare the function prototypes*  *Enter main function*  *Declare variables*  *User inputs beginning balance*  *Checks if input is valid*  *Sets beginning balance to current balance*  *User inputs bet*  *Checks if input is valid*  *User inputs game type/ bet type*  *If input = 1, call snPlay function*  *If input = 2, call oddPlay function*  *If input = 3, call evnPlay function*  *If input = 4, call blkPlay function*  *If input = 5, call redPlay function*  *Checks current value of current balance*  *If greater than 0, prompt to play again*  *If yes, go to certain point in program*  *If less than or equal to 0, error message*  *Call res function to display results*  *Open file*  *Out the results*  *Close file*  *Return 0* |  |

**Program Code**

#include <iostream>

#include <cstring>

#include <iomanip>

#include <cstdlib>

#include <ctime>

#include <cmath>

#include <fstream>

#include <algorithm> // for find

#include <iterator> // for begin, end

using namespace std;

// FUNCTIONS ///////////////////////////////////////////////////////////////////

// This function is used when the player decides to play on one single number.

int snPlay(int, int, float, int);

// This function is used to check if number is even.

bool isEven(int);

// This function shows current balance after a play.

int showBal(int);

// This function is used for when player plays on even.

int evnPlay(int, float, int);

// This function is used for when player plays on odd.

int oddPlay(int, float, int);

// This function displays the final results

int res(int, int);

// This function is used when the player plays on all black numbers.

int blkPlay(int, float);

// This function is used when the player plays on all red numbers.

int redPlay(int, float);

int main()

{

int num; // number for input for number gametype

int rng; // random number to be generated

int bBal; // beginning balance of player

int cBal; // current balance

char agn; // play again?

float bet; // bet inputted

// array for player decisions (number, even, odd)

char gamet[4];

// display main menu

cout << "\t\tWelcome to Roulette!\n\n";

cout << "=================================================================\n";

// player inputs balance to be played

cout << "Enter amount of money to be played: \n$";

cin >> bBal;

while (bBal <= 0)

{

cout << "\nInsufficient funds to play.\n\n";

cout << "Enter amount of money to be played: \n$";

cin >> bBal;

}

// set balance input set to current balance

cBal = bBal;

begin: // links the program to come back here to play again

// player inputs their money for bet

cout << "\nHow much would you like to bet?\n$";

cin >> bet;

// checks if player is inputting valid bet

while(bet > cBal)

{

cout << "\nInsufficient funds to place bet.\n\n";

cout << "How much would you like to bet?\n$";

cin >> bet;

}

errBet:

// select bet type

cout << "\nChoose your type of bet:\n";

cout << "1. Specific Number\n";

cout << "2. Odd Numbers\n";

cout << "3. Even Numbers\n";

cout << "4. Black Numbers\n";

cout << "5. Red Numbers\n";

cin >> gamet;

cout << endl;

// specific number bet

if(!strcmp(gamet,"1")||(!strcmp(gamet,"1.")))

{

cout << "You are betting on a specific number.\n";

cBal=snPlay(num, rng, bet, cBal);

}

// odd number bet

if(!strcmp(gamet,"2")||(!strcmp(gamet,"2.")))

{

cout << "You are betting on odd numbers.\n\n";

cBal=oddPlay(rng, bet, cBal);

}

// even number bet

if(!strcmp(gamet,"3")||(!strcmp(gamet,"3.")))

{

cout << "You are betting on even numbers.\n\n";

cBal=evnPlay(rng, bet, cBal);

}

// black number bet

if(!strcmp(gamet,"4")||(!strcmp(gamet,"4.")))

{

cout << "You are betting on black numbers.\n\n";

cBal=blkPlay(cBal, bet);

}

// red number bet

if(!strcmp(gamet,"5")||(!strcmp(gamet,"5.")))

{

cout << "You are betting on red numbers.\n\n";

cBal=redPlay(cBal, bet);

}

// checks if player has enough funds to play again

if(cBal > 0)

{

cout << "Keep playing? (Y/N)\n";

cin >> agn;

cout << endl;

// prompts player if wants to play again

if ((agn == 'Y')||(agn == 'y'))

{

goto begin; // runs the program again from the begin:

}

}

if (cBal <= 0)

{

cout << "You are unable to play due to your funds.\n\n";

}

res(bBal, cBal);

ofstream results ("Past Results.txt");

if(results.is\_open())

{

results << "Your results:" << endl;

results << "You entered the game with: $" << setw(4) << bBal << endl;

results << "Your ending balance: $" << setw(4) << cBal << endl;

results << "Total earnings: $" << setw(4) << cBal - bBal << endl;

results.close();

}

return 0;

}

// bool to check if number is even

bool isEven(int n)

{

if (n%2==0)

return true;

else

return false;

}

// displays current balance remaining

int showBal(int n)

{

cout << "You are currently sitting on $" << n << endl << endl;

}

// displays final results

int res(int bb, int c)

{

cout << "Your results:\n";

cout << "You entered the game with: $" << bb << endl;

cout << "Your ending balance: $" << c << endl;

cout << "Total earnings: $" << c - bb << endl;

}

// play on one number

int snPlay(int n, int r, float b, int c)

{

cout << "\nWhat number would you like to bet on? (1-36)\n"; // what number to bet on

cin >> n;

while((n > 36)||(n < 1))

{

cout << "\nInvalid number.\n\n";

cout << "What number would you like to bet on? (1-36)\n";

cin >> n;

}

// random number generator between 0-36

srand(time(NULL));

r = rand() % (36 - 1 + 1) + 1;

cout << "The ball landed on " << r << "\n";

// loss

if(n != r)

{

cout << "You lose $" << b << "\n";

c -= b; // add -bet to winnings

}

// win

else

{

cout << "You win $" << 35\*b << endl;

c += 35\*b; // multiply bet times 35 and add to winnings

}

showBal(c);

return c;

}

// play on even numbers

int evnPlay(int r, float b, int c)

{

// random number generator between 1-36

srand(time(NULL));

r = rand() % (36 - 1 + 1) + 1;

cout << "The ball landed on " << r << endl;

// win

if(isEven(r))

{

cout << "You win $" << b << endl;

c += b; // add bet to winnings

}

// loss

else

{

cout << "You lose $" << b << endl;

c -= b; // add -bet to winnings

}

showBal(c);

return c;

}

// play on odd numbers

int oddPlay(int r, float b, int c)

{

// random number generator between 1-36

srand(time(NULL));

r = rand() % (36 - 1 + 1) + 1;

cout << "The ball landed on " << r << endl;

// loss

if(isEven(r)) // check if random number is even

{

// if even, then lose because betted on odd

cout << "You lost $" << b << endl;

c -= b; // add bet to winnings

}

// win

else

{

cout << "You win $" << b << endl;

c += b; // add -bet to winnings

}

showBal(c);

return c;

}

// play on black numbers

int blkPlay(int c, float b)

{

// initialize the random seed

srand(time(NULL));

// array for ALL numbers

int n[36] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12,

13, 14, 15, 16, 17, 18, 19, 20, 21, 22,

23, 24, 25, 26, 27, 28, 29, 30, 31, 33,

34, 35, 36 };

// array for black numbers ONLY

int bn[18] = { 2, 4, 6, 8, 10, 11, 13, 15, 17, 20, 22,

24, 26, 28, 29, 31, 33, 35 };

int r = n[rand()%36]; // generates a random number between 0 and 35

cout << "The ball landed on " << r << endl;

// checks if r is found in second array

if(find(begin(bn), end(bn), r) != end(bn))

{

cout << "You win $" << b << endl;

c += b; // add bet to winnings

}

else

{

cout << "You lost $" << b << endl;

c -= b; // add -bet to winnings

}

showBal(c);

return c;

}

// play on red numbers

int redPlay(int c, float b)

{

// initialize the random seed

srand(time(NULL));

// array for ALL numbers

int n[36] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12,

13, 14, 15, 16, 17, 18, 19, 20, 21, 22,

23, 24, 25, 26, 27, 28, 29, 30, 31, 33,

34, 35, 36 };

// array for red numbers ONLY

int rn[18] = { 1, 3, 5, 7, 9, 12, 14, 16, 18, 19, 21,

23, 25, 27, 30, 32, 34, 36 };

int r = n[rand()%36]; // generates a random number between 0 and 35

cout << "The ball landed on " << r << endl;

// checks if r is found in second array

if(find(begin(rn), end(rn), r) != end(rn))

{

cout << "You win $" << b << endl;

c += b; // add bet to winnings

}

else

{

cout << "You lost $" << b << endl;

c -= b; // add -bet to winnings

}

showBal(c);

return c;

}