Project Blackjack

Phillip Pascual

CSC-5-40107

M-Th 9:50

Introduction

Blackjack, also known as ‘21’, is a popular game of chance played in casinos worldwide. The object of the game is simple: to build a hand worth as much as 21 points without going over 21. The dealer, or ‘house’, does the same up, within a certain set of guidelines put forth by the house, and both hands are compared. If the player’s hand is greater than the dealer’s hand, or if the dealer goes over 21, the player wins. Alternatively, if the opposite is true, the dealer wins. Cards are worth their face value, except for face cards (Jack through King), which are worth 10, and Aces, which are worth 1 or 11. There are additional rules for more advanced players, but these are the basic rules of the game.

The house’s advantage comes from the setup of the game. While both of the player’s initial 2 cards are known, one card in the dealer’s hand is kept face-down, forcing the player to make a decision based on incomplete information. Additionally, any time the player exceeds 21 he or she automatically loses, in contrast to the dealer, who has a chance to ‘bust’ only after the player has been given an opportunity to do so. However, in spite of these advantages, the house advantage is still approximately 0.5%.

This program makes an initial effort in helping a blackjack player employ proper strategy in his or her play. By simulating a dealer’s base behavior, a player has the opportunity to play games of blackjack without any type of monetary risk.

Project Summary

Size: Approximately 340 lines

This project encompasses concepts of C++ covered and practiced in Chapter’s 1-5.

From initial flowcharting through final coding, this project has taken approximately 15 hours to complete over the course of a 7 days. I chose a blackjack simulation because I was very familiar with the game and felt it would be easier to code for a game that I was very knowledgeable about than one that I had learned recently.

This project makes extensive use of loops to manage the flow of the game. Several loops are used to deal the cards, obtain decisions from the player on whether to hit or stand, execute decisions made by the rudimentary AI on whether to hit or stand, and to check whether an Ace dealt should be treated as a 1 or 11. Personally I found it a very challenging experience. In a future version I would like to add the more advanced rules such as doubling-down and splitting cards, as well as integrate a betting system into the game, and will be doing so once additional programming concepts are learned and integrated into my coding.

Flowchart

Variables

|  |  |  |
| --- | --- | --- |
| **Constant Integer** | PERCENT | Used for percent conversion |
| **Integer** | p1 | Player card 1 |
|  | p2 | Player card 2 |
|  | pp1 | Retains original p1 value for ace check |
|  | pp2 | Retains original p2 value for ace check |
|  | pAdd | Additional player cards |
|  | d1 | Dealer card 1 |
|  | d2 | Dealer card 2 |
|  | dd1 | Retains original d1 value for ace check |
|  | dd2 | Retains original d2 value for ace check |
|  | dAdd | Additional dealer cards |
|  | choice | Input-designates whether to hit or stand |
|  | pScore | Player score |
|  | dScore | Dealer score |
|  | games | Input-number of games desired |
|  | wins | Number of wins, initialized=0 |
|  | lose | Number of losses, initialized=0 |
|  | push | Number of pushes, initialized =0 |
|  | tGames | wins+lose |
|  | num | Counter for for loop @ line 65 |
| **Char** | kpPlay | Input-designates whether or not to end program |
| **String** | name | Input-player name |
|  | pCards | Player’s card hand |
|  | dCards | Dealer’s card hand |
| **Bool** | deal | Initialized=1. Used in do-while loop @ line 148, sets initial true/false condition. |
|  | dStand | Initialized=false. Used in do-while loop @ line 194, sets initial false condition. |

Pseudocode

//Set RNG seed

//Instantiate and open file

//Declare Variables

//Ints for first/second player card, first/second dealer card, additional

//player cards, additional dealer cards, player/dealer score, number of games to play,

//wins/losses/pushes

//String to display cards

//Program description and input of number of games

//Loop that will play blackjack as long as player would like to

//Create loop that plays blackjack for number of games player inputs

//Deal player cards

//Generates random number between 2-14 for card 1

//Corresponds random number 2-14 to player card 2-A

//If RNG generates 10-K, score automatically set to add 10

// If RNG generates Ace, score automatically set to add Ace=11

//Generates random number between 2-14 for card 2

//Corresponds random number 2-14 to player card 2-A

//If RNG generates 10-K, score automatically set to add 10

// If RNG generates Ace, score automatically set to add Ace=11

//If two aces dealt, first ace switched from worth 11 to

//worth 1

//Add first and second card to find initial score

//Display player hand

//Deal first dealer card

//RNG generates random number 2-14

//Corresponds random number 2-14 to dealer card 2-A

//If RNG generates 10-K, score automatically set to add 10

// If RNG generates Ace, score automatically set to add Ace=11

//Display dealer hole card

//Obtain decision from player on whether to hit or stand

//If player chooses to hit

//Deal additional player cards

//Create boolean allowing loop to deal additional cards

//Generates random number 2-14

//Corresponds random number 2-14 to player card 2-A

//If total score busts and Ace dealt, subtract 10 to make

Ace=1

//Display new card

//Bust automatic loss, stop dealing

//Give choice to stand or hit

//If stand, stop dealing to player

//Else, continue dealing to player

//When player chooses to stand start dealer's cards

//Create loop that deals when dealer has under 17

//Deal dealer's hole card

//Generates random number 2-14

//Corresponds random number 2-14 to dealer

card 2-A

//Dealer reveals hole card and determines if needs

to hit or stand

//Dealer under 17, hits

//If over 17, dealer stands and program

bypasses next loop

//If dealer needs to deal additional cards to

himself if less than or equal to 17

//Deal additional dealer cards

//Add dealer's new score

//Dealer's hand displayed and decides if

needs to hit again

//Compare player and dealer scores and determine the winner

//Player score over 21 means automatic loss

//Loss increment by 1

//Total games increment by 1

//Dealer score over 21 means automatic win

//Wins increment by 1

//Total games increment by 1

//Dealer score greater than player score means loss

//Losses increment by 1

//Total games increment by 1

//Player score greater than dealer score means win

//Wins increment by 1

//Total games increment by 1

//Ties mean push, nobody wins

//Pushes increment by 1

//Total games increment by 1

//When all games are completed, statistics will be displayed

//Give player choice on if to play again.

//Go back to top of loop if player wants to play again

//If player chooses not to play again, output most recent stats to file and return

Final Code

/\*

File: main.cpp

Author: Phillip Pascual

Created on January 27, 2017, 5:00 PM

Purpose: Project 1 - Simple Blackjack Program

\*/

//System Libraries

#include <iostream>

#include <cstdlib>

#include <ctime>

#include <iomanip>

#include <fstream>

using namespace std;

//User Libraries

//Global Constants

//Such as PI, Vc, -> Math/Science values

//as well as conversions from system of units to

//another

const float PERCENT=100.0; //Constant float for percentage conversion

//Function Prototypes

//Executable code begins here!!!

int main(int argc, char\*\* argv) {

//Set RNG seed

srand(static\_cast<unsigned int>(time(0))); //Sets RNG seed based on system time

//Instantiate and open file

ifstream in;

ofstream out;

in.open("blackjackstats.txt");

out.open("blackjackstats.txt");

//Declare Variables

//Ints for first/second player card, first/second dealer card, additional

//player cards, additional dealer cards, player/dealer score, number of games to play,

//wins/losses/pushes

unsigned int p1,pp1,p2,pp2,pAdd,d1,dd1,d2,dd2,dAdd,choice,pScore,dScore,games,

wins=0,lose=0,push=0,tGames=0;

string name;

//String to display cards

string pCards,dCards;

char kPlay;

//Program description and input of number of games

cout<<"Welcome to Blackjack!"<<endl;

cout<<"You will play a game of blackjack against a computer dealer. The object"<<endl;

cout<<"of the game is to make a hand that beats the dealer's hand without going"<<endl;

cout<<"over 21. The dealer will stand on any hand worth 17 or above."<<endl;

cout<<endl;

cout<<"Please enter your name: "<<endl;

cin>>name;

cout<<endl;

do{

cout<<"How many games would you like to play? ";

cin>>games;

cout<<endl;

//Create loop that plays blackjack for number of games player inputs

for(int num=1;num<=games;num++){

//Deal player cards

p1=pp1=rand()%13+2; //Generates random number between 2-14 for card 1

pCards=' ';

dCards=' ';

//Corresponds random number 2-14 to player card 2-A

switch(p1){

case 2:pCards+="2 ";break;

case 3:pCards+="3 ";break;

case 4:pCards+="4 ";break;

case 5:pCards+="5 ";break;

case 6:pCards+="6 ";break;

case 7:pCards+="7 ";break;

case 8:pCards+="8 ";break;

case 9:pCards+="9 ";break;

case 10:{pCards+="10 ";p1=10;}break; //If RNG generates 10-K, score

case 11:{pCards+="J ";p1=10;}break; //automatically set to add 10

case 12:{pCards+="Q ";p1=10;}break;

case 13:{pCards+="K ";p1=10;}break;

case 14:{pCards+="A ";p1=11;}break; //If RNG generates A, score

} //automatically set to add A

p2=pp2=rand()%13+2; //Generates random number between 2-14 for card 2

//Corresponds random number 2-14 to player card 2-A

switch(p2){

case 2:pCards+="2 ";break;

case 3:pCards+="3 ";break;

case 4:pCards+="4 ";break;

case 5:pCards+="5 ";break;

case 6:pCards+="6 ";break;

case 7:pCards+="7 ";break;

case 8:pCards+="8 ";break;

case 9:pCards+="9 ";break;

case 10:{pCards+="10 ";p2=10;}break;

case 11:{pCards+="J ";p2=10;}break;

case 12:{pCards+="Q ";p2=10;}break;

case 13:{pCards+="K ";p2=10;}break;

case 14:{pCards+="A ";p2=11;}break;

}

if(pp1==14&&pp2==14){ //If two aces dealt, first ace switched from

p1=11; //worth 11 to worth 1

}

pScore=p1+p2; //Adds first and second card to find initial score

cout<<"Your first two cards are:"<<pCards<<"= "<<pScore;

cout<<endl;

//Deal first dealer card

d1=dd1=rand()%13+2; //RNG generates random number 2-14

//Corresponds random number 2-14 to dealer card 2-A

switch(d1){

case 2:dCards+="2 ";break;

case 3:dCards+="3 ";break;

case 4:dCards+="4 ";break;

case 5:dCards+="5 ";break;

case 6:dCards+="6 ";break;

case 7:dCards+="7 ";break;

case 8:dCards+="8 ";break;

case 9:dCards+="9 ";break;

case 10:{dCards+="10 ";d1=10;}break;

case 11:{dCards+="J ";d1=10;}break;

case 12:{dCards+="Q ";d1=10;}break;

case 13:{dCards+="K ";d1=10;}break;

case 14:{dCards+="A ";d1=11;}break;

}

if(d1<=13&&d1>=10){

p1=10;

}

else if(d1==14){

d1=11;

}

cout<<"The dealer's hand shows a"<<dCards<<endl;

cout<<endl;

//Obtain decision from player on whether to hit or stand

cout<<"Please enter 1 to hit, 2 to stand: ";

cin>>choice;

while(choice!=1&&choice!=2){

cout<<"Please enter a valid number (1 or 2): ";

cin>>choice;

}

//If player chooses to hit

if(choice==1){

//Deal additional player cards

bool deal=1; //Boolean allowing loop to deal additional cards

do{

pAdd=rand()%13+2; //Generates random number 2-14

switch(pAdd){

case 2:pCards+="2 ";break;

case 3:pCards+="3 ";break;

case 4:pCards+="4 ";break;

case 5:pCards+="5 ";break;

case 6:pCards+="6 ";break;

case 7:pCards+="7 ";break;

case 8:pCards+="8 ";break;

case 9:pCards+="9 ";break;

case 10:{pCards+="10 ";pAdd=10;}break;

case 11:{pCards+="J ";pAdd=10;}break;

case 12:{pCards+="Q ";pAdd=10;}break;

case 13:{pCards+="K ";pAdd=10;}break;

case 14:{pCards+="A ";pAdd=11;}break;

}

pScore+=pAdd;

if (pAdd==11&&pScore>=22){

pScore-=10;

}

cout<<"The dealer gives you a card. Your hand is now"<<pCards<<"= "<<pScore<<endl;

if(pScore>21){

cout<<"Your score is over 21, you bust!"<<endl;

deal=0;

}

else{

cout<<"Please enter 1 to hit, 2 to stand: ";

cin>>choice;

while(choice!=1&&choice!=2){

cout<<"Please enter a valid number (1 or 2): ";

cin>>choice;

}

if (choice==2){

deal=0;

}

else {

deal=1;

}

}

}while (deal);

}

//When player chooses to stand

if (choice==2) {

bool dStand=false;

do{

//Deal dealer's hole card

d2=dd2=rand()%13+2; //Generates random number 2-14

switch(d2){

case 2:dCards+="2 ";break;

case 3:dCards+="3 ";break;

case 4:dCards+="4 ";break;

case 5:dCards+="5 ";break;

case 6:dCards+="6 ";break;

case 7:dCards+="7 ";break;

case 8:dCards+="8 ";break;

case 9:dCards+="9 ";break;

case 10:{dCards+="10 ";d2=10;}break;

case 11:{dCards+="J ";d2=10;}break;

case 12:{dCards+="Q ";d2=10;}break;

case 13:{dCards+="K ";d2=10;}break;

case 14:{dCards+="A ";d2=11;}break;

}

if(dd2<=13&&dd2>=10){

d2=10;

}

else if(dd2==14){

p2=11;

}

if(dd1==14&&dd2==14){

d1=11;

}

}while (dStand);

dScore=d1+d2;

//Dealer reveals hole card and determines if needs to hit or stand

cout<<"The dealer turns over his hole card. His hand is now"<<dCards<<"= "<<dScore<<endl;

if(dScore<17){

cout<<"Dealer has under 17 and needs to hit again..."<<endl;

}

else if(dScore>=17){

cout<<"Dealer has 17 or over and stands."; //If over 17, dealer stands and program bypasses next loop

}

//Dealer needs to deal additional cards to himself if less than or equal to 17

while(dScore<17){

//Deal additional dealer cards

dAdd=rand()%13+2; //RNG generates numbers 2-14

switch(dAdd){ //Converts 2-14 into playing cards

case 2:dCards+="2 ";break;

case 3:dCards+="3 ";break;

case 4:dCards+="4 ";break;

case 5:dCards+="5 ";break;

case 6:dCards+="6 ";break;

case 7:dCards+="7 ";break;

case 8:dCards+="8 ";break;

case 9:dCards+="9 ";break;

case 10:{dCards+="10 ";dAdd=10;}break;

case 11:{dCards+="J ";dAdd=10;}break;

case 12:{dCards+="Q ";dAdd=10;}break;

case 13:{dCards+="K ";dAdd=10;}break;

case 14:{dCards+="A ";dAdd=11;}break;

}

if(dAdd==11&&dScore>=12){//If ace is dealt and score is over 12,

dAdd=1; //one ace treated as 1 instead of 11

}

dScore+=dAdd;//Add dealer's new score

//Dealer's hand displayed and decides if needs to hit again

cout<<"The dealer deals himself another card. His hand is now"<<dCards<<"= "<<dScore<<endl;

if(dScore<17){//Over 17, hit

cout<<"The dealer is still under 17 and needs to hit again..."<<endl;

}

else if (dScore>=17&&dScore<=21){//Between 17-21, stand

cout<<"The dealer stands on 17 or over..."<<endl;

}

else if (dScore>21){//Over 21, bust

cout<<"The dealer busts!"<<endl;

}

}

}

cout<<endl;

//Compare player and dealer scores and determine the winner

if(pScore>21){//Player score over 21 means automatic loss

cout<<"Sorry, you lose."<<endl;

lose++; //Loss increment by 1

tGames++; //Total games increment by 1

cout<<endl;

}

else if (dScore>21){//Dealer score over 21 means automatic win

cout<<"Congratulations! You win!"<<endl;

wins++; //Wins increment by 1

tGames++; //Total games increment by 1

cout<<endl;

}

else if (pScore<dScore){//Dealer score greater than player score means

//loss

cout<<"Sorry, you lose."<<endl;

lose++; //Losses increment by 1

tGames++; //Total games increment by 1

cout<<endl;

}

else if (pScore>dScore){//Player score greater than dealer score means

//win

cout<<"Congratulations! You win!"<<endl;

wins++; //Wins increment by 1

tGames++; //Total games increment by 1

cout<<endl;

}

else{

cout<<"A push, nobody wins."<<endl;//Ties mean push, nobody wins

push++; //Pushes increment by 1

tGames++; //Total games increment by 1

cout<<endl;

}

}

//When all games are completed, statistics will be displayed

cout<<"Thanks for playing! Here are your win/loss statistics for this session: "<<endl;

cout<<"Total Games: "<<tGames<<endl;

cout<<"Wins: "<<setw(5)<<right<<wins<<endl;

cout<<"Losses: "<<setw(5)<<right<<lose<<endl;

cout<<"Pushes: "<<setw(5)<<right<<push<<endl;

cout<<"Win percentage: "<<setw(7)<<right<<fixed<<setprecision(2)<<(static\_cast<float>(wins)/tGames)\*PERCENT<<"%"<<endl;

cout<<"Loss percentage: "<<setw(7)<<right<<fixed<<setprecision(2)<<(static\_cast<float>(lose)/tGames)\*PERCENT<<"%"<<endl;

cout<<"Push percentage: "<<setw(7)<<right<<fixed<<setprecision(2)<<(static\_cast<float>(push)/tGames)\*PERCENT<<"%"<<endl;

cout<<endl;

cout<<"Would you like to play again (Y or N)? ";

cin>>kPlay;

while (kPlay!='Y'&&kPlay!='y'&&kPlay!='N'&&kPlay!='n'){

cout<<"Please enter Y or N: ";

cin>>kPlay;

}

}while (kPlay=='Y'||kPlay=='y');

//Export stats to file

cout<<"Thanks for playing, your stats are available at blackstats.txt!"<<endl;

out<<name;

out<<endl;

out<<"Total Games: "<<tGames<<endl;

out<<endl;

out<<"Wins: "<<setw(5)<<right<<wins<<endl;

out<<endl;

out<<"Losses: "<<setw(5)<<right<<lose<<endl;

out<<endl;

out<<"Pushes: "<<setw(5)<<right<<push<<endl;

out<<endl;

out<<"Win percentage: "<<setw(7)<<right<<fixed<<setprecision(2)<<(static\_cast<float>(wins)/tGames)\*PERCENT<<"%"<<endl;

out<<endl;

out<<"Loss percentage: "<<setw(7)<<right<<fixed<<setprecision(2)<<(static\_cast<float>(lose)/tGames)\*PERCENT<<"%"<<endl;

out<<endl;

out<<"Push percentage: "<<setw(7)<<right<<fixed<<setprecision(2)<<(static\_cast<float>(push)/tGames)\*PERCENT<<"%"<<endl;

//Exit stage right!

return 0;

}