Blackjack 21

Venkatesh Tavva

Project 2

CIS5 – 45548

July 28, 2017

**Introduction:**

Title: Blackjack 21

Blackjack 21 is a card game where the goal is to get a hand with point total to or very near 21, without exceeding 21. It is played usually against the dealer. At the beginning, both the player and the dealer are dealt 2 cards. However only one of the dealer’s cards is actually shown, the other is hidden. After this, the player gets the option to either hit or stand. If the player chooses to hit, they will be dealt another card and if they choose to stand, they will keep their cards and will go against the dealer’s two cards. If they keep choosing to hit, and eventually go over 21, they immediately lose, which is known as “bust”. If the dealer’s card total is less than 17, they must hit, and if it is above 17, they will stand. If the dealer busts, meaning card total goes over 21, the player wins. Also, if the player’s total is higher than the dealer’s, then the player wins and if the card total between player and dealer is equal then it is a tie. If the player loses, then they have to forfeit their bet, but however if they win, they get keep their original and earn an additional bet. So, if you bet $15, and you win against dealer then you end up earning $30 total. If it is a tie, then player keeps the original bet.

**Summary:**

Project size: about 290 lines

Number of variables: 24

This project took me a little over a week and a half to do and most time was spent on creating the functions and figuring out how to implement them. The game uses many of the concepts learned in the class though not all. The game can be modified and improved several ways, one which involves the implementation of some blackjack side rules such as insurance, surrender, splitting, and doubling down. One of major issues I dealt with was the game not displaying the correct score for the player after each hit.

**List of Concepts:**

|  |  |  |  |
| --- | --- | --- | --- |
| Chapter | Section | Topic | Line Number |
| 2 | 2 | cout | 49, 56, 60, 62, 64, 66, 75, 76, 82, 83, 85, 86, 89, 103, 104, 106, 113, 114, 130, 131, 138, 139, 144, 145, 151, 152, 161, 163, 164, 167, 171, 191, 195, 199, 203. 207, 211, 216, 220, 224, 228, 237, 239, 286, 287, 289, 291 |
|  | 3 | Libraries | 9, 10, 11, 12, 13 |
|  | 4 | Variables/literals | 21, 22, 23, 24, 25, 26, 31, 32, 33, 34, 35, 36, 37, 38, 39, 53, 77, 78, 79, 120, 188, 213, 235, 243, 244, 257, 258, 261, 262 |
|  | 5 | identifiers | 21, 22, 23, 24, 25, 26, 31, 32, 33, 34, 35, 36, 37, 38, 39, 53, 77, 78, 79, 120, 188, 213, 235, 243, 244, 257, 258, 261, 262 |
|  | 6 | integers | 22, 23, 25, 26, 32, 33, 34, 35, 38, 39, 79, 120, 181, 188, 213, 231, 233, 235, 244, 257, 258, 261, 262, 284 |
|  | 7 | characters | 36, 37, 77 |
|  | 8 | strings | 53 |
|  | 9 | floats |  |
|  | 10 | bools | 24, 31, 78, 127, 243 |
|  | 11 | Size of |  |
|  | 12 | Variables 7 characters or less | 21, 22, 23, 24, 25, 26, 31, 32, 33, 34, 35, 36, 37, 38, 39, 53, 77, 78, 79, 120, 188, 213, 235, 243, 244, 257, 258, 261, 262 |
|  | 13 | Scope\*\*\*\*\* |  |
|  | 14 | Arithmetic operators | 41, 188, 195, 213, 262, 270, 274, 280 |
|  | 15 | comments | 21, 22, 23, 24, 25, 26, 31, 32, 33, 34, 35, 36, 37, 38, 39, 41, 69, 70, 71, 72, 73, 74, 77, 78, 79, 180, 188, 213, 243, 244, 261, 262 |
|  | 16 | Named constants |  |
|  | 17 | Programming style |  |
|  |  |  |  |
| 3 | 1 | Cin | 61, 65, 90, 166 |
|  | 2 | Math expression | 41, 188, 213, 262, 270, 274, 280 |
|  | 3 | Mixing data types |  |
|  | 4 | Overflow/underflow |  |
|  | 5 | Type casting | 30 |
|  | 6 | Multiple assignment |  |
|  | 7 | Formatting output |  |
|  | 8 | strings | 53 |
|  | 9 | Math library |  |
|  | 10 | Hand tracing |  |
|  |  |  |  |
| 4 | 1 | Relational operators | 63, 91, 97, 109, 111, 121, 136, 142, 160, 168, 169, 181, 189, 193, 197, 201, 205, 214, 218, 222, 233, 277 |
|  | 2 | if | 47, 63, 91, 111, 128, 136, 160, 169, 189, 214, 248, 263, 277 |
|  | 4 | If-else | 47, 51, 91, 101, 111, 118, 134, 136, 149, 189, 209, 214, 226, 263, 272 |
|  | 5 | nesting |  |
|  | 6 | If-else-if | 91, 97, 101, 136, 142, 149, 189, 193, 197, 201, 205, 209, 214, 218, 222, 226, 263, 268, 272 |
|  | 7 | flags |  |
|  | 8 | Logical operators | 109, 248, 277 |
|  | 11 | Validating user input | 103 |
|  | 13 | Conditional operator |  |
|  | 14 | switch |  |
|  |  |  |  |
| 5 | 1 | Increment/decrement | 94, 124, 181, 233, 265, 266, 279 |
|  | 2 | While | 54, 109, 168, |
|  | 5 | Do-while | 45, 80, 109, 168 |
|  | 6 | For loop | 181, 233, 259 |
|  | 11 | Files input/output both | 46, 58 |
|  | 12 | No breaks in loops |  |
|  |  |  |  |
| 6 | 3 | Function Prototypes | 21, 22, 23, 24, 25, 26 |
|  | 5 | Passing by value | 84 |
|  | 8 | Returning value from functions | 69, 70, 71, 72, 79, 86, 93, 123, 107, 120, 125 |
|  | 9 | Returning a boolean |  |
|  | 10 | No Global Variables allowed | 188, 235, 243, 244, 257, 258 |
|  |  | Only Global Constants |  |
|  |  | Meaning Conversions, Physical Constants only |  |
|  | 11 | Static local |  |
|  | 12 | Default arguments | 179, 186, 231, 241, 255, 284 |
|  | 13 | Reference parameters |  |
|  | 14 | Overloading functions |  |
|  | 15 | Exit function |  |
|  |  |  |  |
| 7 | 4 | Array initialization | 31, 32, 34 |
|  | 6 | Processing arrays | 183, 235, 261 |
|  | 7 | Parallel arrays | 84, 123, 248 |
|  | 8 | Arrays as function arguments | 179, 231, 241, 255, 284 |
|  | 9 | 2-D arrays |  |
|  | 12 | STL vector |  |
|  |  |  |  |
| 8 | 1 | Linear and binary search |  |
|  | 3 | Bubble and selection sort |  |
|  | 5 | Search/sorting vectors |  |
|  |  |  |  |

**Pseudo Code:**

*Initialize*

*If player is ready to deal*

*Display input for bet from player*

*Else if player selects ‘n’*

*Display “Good-bye” and exit*

*If player inputs bet*

*Dealer and Player are dealt two random cards*

*One dealer card is hidden*

*If player decides to hit*

*Another random card is dealt*

*Score is added to player’s score until total reaches 21 or above*

*Else if player decides to stand*

*Program will determine whether won or loss*

*Display results*

*If dealer’s total is less than 17*

*Dealer is dealt another random card*

*If player’s score is greater than 21*

*Player loses and no payout*

*Else if dealer’s score is greater than 21*

*Player wins twice original bet*

*Else if dealer score and player score is equal*

*Declare tie and player gets original bet*

*Else if player score is greater than dealer score*

*Player wins twice original bet*

*Else*

*Player loses and bet is lost*

*If player chooses to play again*

*Start the program from beginning*

*Else if player selects ‘n’*

*Display “Good-bye” and exit*

**Actual Code:**

/\*

\* File: main.cpp

\* Author: Venkatesh Tavva

\* Created on July 24th, 2017, 7:40 PM

\* Purpose: a blackjack game

\*/

//System Libraries

#include <iostream> //Input - Output Library

#include <ctime>

#include <cstdlib>

#include <fstream>

#include <string>

using namespace std; //Name-space under which system libraries exist

//User Libraries

//Global Constants

//Function Prototypes

void shuCard(bool cdeck[]);//shuffles card from deck

void disCard(int cards );//displays the card

void curCard(int current[], int ccount);//shows current card

int newCard(bool cdeck[]);//getting the next card

int kpScore (int current[], int ccount);//keeps score

void disRes (int dcard[], int dccount, int pcard[], int pccount);//displays results

//Execution begins here

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

//Declare variables

srand(static\_cast<unsigned int>(time(0)));

bool cdeck[52];//deck of cards; holds 52 cards

int pcard[12];//player cards, holds 12 cards

int pccount1;// player card count

int dcard[12];//dealer cards, holds 12 cards

int dccount1;//dealer card count

char choice1;//input of choice

char choice3;//input of choice

int bet;//player's bet input in dollars

int betwon;//player's winnings in results

//Initialize variables

betwon = bet\*2;//winnings twice original bet in dollars

//Input data

//Map inputs to outputs or process the data

do{

ifstream infile("header.txt");

if (infile.fail())

{

cout<<"File couldn't open";

}

else

{

string header;

while(getline(infile,header))

{

cout<<header<<endl<<endl;

}

infile.close();

}

cout<<"Ready to Deal? (y or n) ";

cin>>choice1;

cout<<endl;

if (choice1 == 'y') {

cout<<"How much do you want to bet? $";

cin>>bet;

cout<<endl;

shuCard(cdeck);

pcard[0] = newCard(cdeck);//first initial player card dealt

dcard[0] = newCard(cdeck);//first initial dealer card hidden

pcard[1] = newCard(cdeck);//second initial player card dealt

dcard[1] = newCard(cdeck);//second initial dealer card shown

pccount1 = 2;//2 cards

dccount1 = 2;//2 cards

cout<<"GOOD LUCK!";

cout<<endl<<endl;

char choice2;//input of choice

bool phit = true;//player hitting

int pscore = kpScore (pcard, pccount1);//player score

do

{

cout<<"Dealer's Card"<<endl;

cout<<"Hidden Card, ";

disCard(dcard[1]);

cout<<endl;

cout<<"Player's score: "<<kpScore(pcard,pccount1)<<endl;

curCard(pcard, pccount1);

cout<<"Hit or Stand (h or s) ";

cin>>choice2;

if (choice2 =='h')

{

pcard[pccount1]=newCard(cdeck);

++pccount1;

}

else if (choice2 == 's')

{

phit=false;

}

else

{

cout<<"Sorry, not an accepted choice";

cout<<endl;

}

cout<<endl;

pscore = kpScore(pcard,pccount1);

} while (phit && pscore<22);

if (pscore>21)

{

cout<<"YOU LOST"<<endl;

cout<<"NO PAYOUT"<<endl<<endl;

disRes(dcard, dccount1, pcard, pccount1);

}

else

{

int dscore = kpScore(dcard, dccount1);

while (dscore<17)

{

dcard[dccount1]=newCard(cdeck);

++dccount1;

dscore = kpScore(dcard, dccount1);

}

bool dloss = (dscore>21);

if (dloss)

{

cout<<"YOU WON!!"<<endl;

cout<<"PAYOUT: $"<<betwon<<endl<<endl;

disRes(dcard, dccount1, pcard, pccount1);

}

else

{

if (pscore==dscore)

{

cout<<"Tie!"<<endl;

cout<<"PAYOUT: $"<<bet<<endl<<endl;

disRes(dcard, dccount1, pcard, pccount1);

}

else if (pscore > dscore)

{

cout<<"YOU WIN!!"<<endl;

cout<<"PAYOUT: $"<<betwon<<endl<<endl;

disRes(dcard, dccount1, pcard, pccount1);

}

else

{

cout<<"You Lost"<<endl;

cout<<"NO PAYOUT"<<endl<<endl;

disRes(dcard, dccount1, pcard, pccount1);

}

}

}

}

if (choice1=='n'){

cout<<"Goodbye";

}

cout<<endl;

cout<<"Do you want to play again? (y or n) ";

cin>>choice3;

cout<<endl;

}while(choice3=='y');

if (choice3=='n')

{

cout<<"Goodbye";

}

//Output the transformed data

//Exit stage right!

return 0;

}

void shuCard(bool cdeck[])

{

for (int i =0; i<52; ++i)

{

cdeck[i]=false;

}

}

void disCard(int cards )

{

int crank = (cards%13);//ranks in card deck

if (crank ==0)

{

cout<<"Ace of ";

}

else if (crank<9)

{

cout<<(crank+1)<<" of ";

}

else if (crank==9)

{

cout<<"Ten ";

}

else if (crank ==10)

{

cout<<"Jack of ";

}

else if (crank ==11)

{

cout<<"Queen of ";

}

else

{

cout<<"King of ";

}

int csuit = (cards/13);//suits in card deck

if (csuit==0)

{

cout<<"clubs";

}

else if (csuit==1)

{

cout<<"diamonds";

}

else if (csuit==2)

{

cout<<"hearts";

}

else

{

cout<<"spades";

}

}

void curCard(int current[], int ccount)

{

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

{

int ncard = current[i];

disCard(ncard);

cout<<" ";

}

cout<<endl;

}

int newCard(bool cdeck[])

{

bool cdealt = true;//card being dealt

int newcard = -1;//new card dealt

do

{

newcard = (rand()%52);

if (!cdeck[newcard])

{

cdealt = false;

}

}while (cdealt);

return newcard;

}

int kpScore (int current[], int ccount)

{

int acount=0;//counting of aces

int score=0;//score

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

{

int ncard = current[i];//the next card

int crank= (ncard%13);//card ranks

if (crank==0)

{

++acount;

++score;

}

else if (crank<9)

{

score=score + (crank+1);

}

else

{

score = score+10;

}

}

if (acount>0 && score<12)

{

--acount;

score = score+10;

}

return score;

}

void disRes (int dcard[], int dccount2, int pcard[], int pccount2)

{

cout<<"FINAL RESULTS"<<endl;

cout<<"Dealer's Score: "<<kpScore(dcard,dccount2)<<endl;

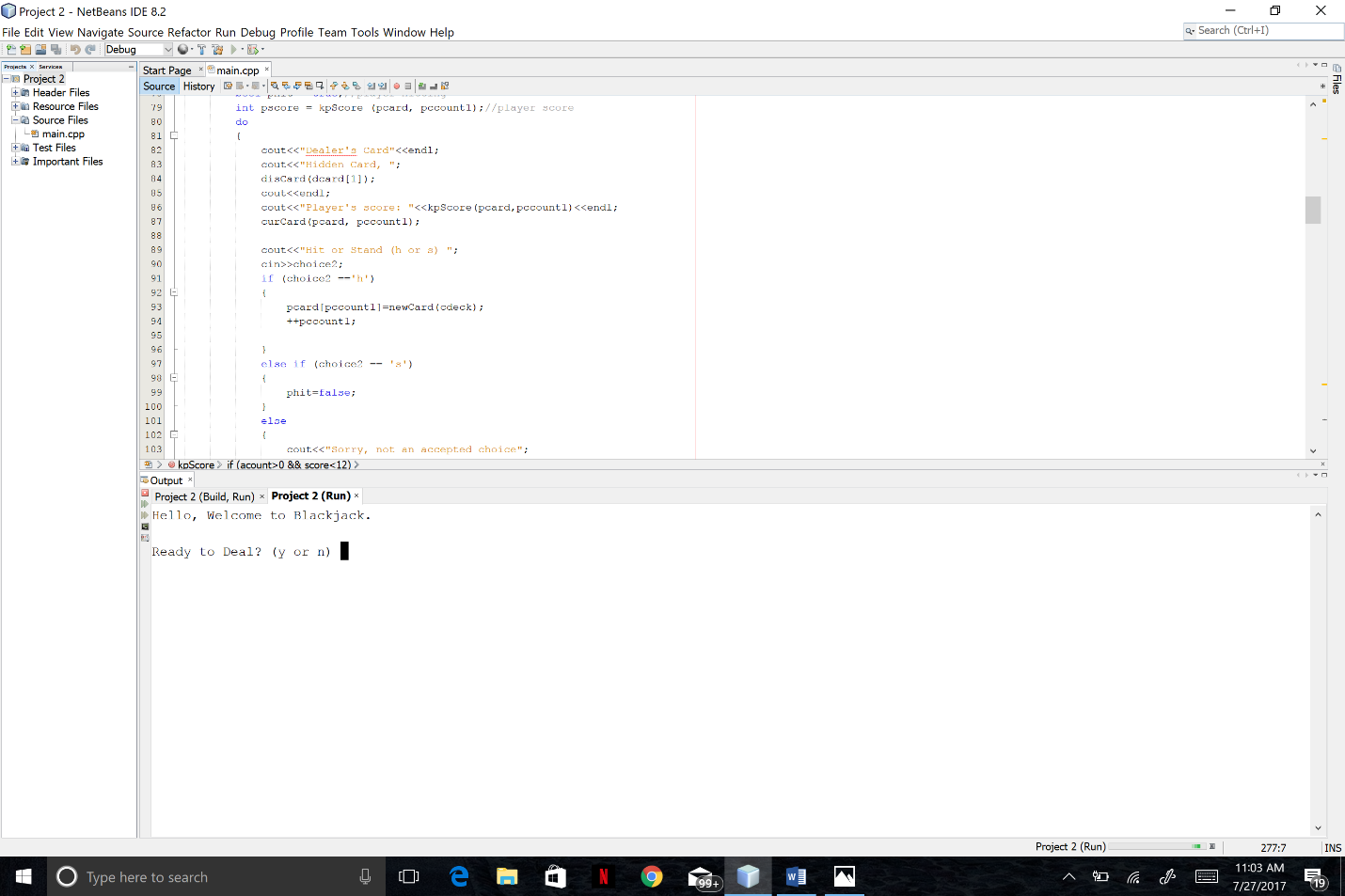
curCard(dcard,dccount2);

cout<<"Player's Score: "<<kpScore(pcard, pccount2)<<endl;

curCard(pcard,pccount2);

cout<<"END OF GAME"<<endl<<endl;

}

**Sample Input/Output:**

