Project 1

Black Jack

By

Cory Chesebro

CSC-17a 43454

1. **Intro**

Black Jack is a simple game that has been played for some time through American history. It consists of one deck of cards, a dealer, and one or more players. The goal of the game is to acquire a hand of cards worth 21 points. Cards with numbers are worth their number value, Face cards excluding the Ace card are worth 10 points. The Ace card counts as either 1 or 11 points, which ever value is in your favor. So, say you have a 2 and an Ace, your value would be 13, but if you get another card and its worth 10 points, your value would still be 13 because if the Ace counted as 11 still, you would go over 21 and lose the game. The way you gain cards is by asking for a “hit”, the way you signal that you don’t want any more cards is by “standing” which means you can no longer hit. Now, if you don’t hit 21 and neither does the dealer or any other player, then the person with the highest value wins. Dealer rules go as such; if you have less than 17 you must hit, if you have over 17 you must stand. So, if you have over 17 in your starting cards its safer, statistically, to stand instead of hitting since you have roughly over a 50% chance of going over 21, and since you don’t have to hit, and the dealer does, you have a better chance of winning the stand than the dealer does winning the hit.

1. Description
   1. Input

There are 4 characters used for input – y for yes, n for no, h for hit, and s for stand. The game reads you the rules and asks if you would like to play, afterwards it gives you a hand assuming you said yes to the rules, and asks if you would like to hit or stand on the hand you receive. After you have won or lost, it again asks for a yes or no answer regarding the game recap.

* 1. Variables

I used 3 structs to hold the important variables needed for the game. One called “Game” held the deck of cards, and an fstream object used for logging the game actions. Another Struct I used was called Cards, which contained a character array and the size of that character array which was used as the players hand. The final struct was the player struct. It contains a Cards object and an integer that was intentionally going to be used for holding money for betting but I didn’t have time to implement betting in this version.

* 1. Concepts

|  |  |
| --- | --- |
| Concept | Line |
| Pointers | Main.cpp line 81 |
| Dynamic mem alloc | Main.cpp line 81 |
| String class | Main.cpp line 141 |
| Structures | Cards.h line 17 |
| Struct Member accessing | Main.cpp line 28 |
| Struct Init | Main.cpp line 27 |
| File writing | Main.cpp line 134 |
| Cstring | Main.cpp line 127 |
| Binary file | Main.cpp line 133 |
| Nestend struct | Cards.h line 29 |

* 1. Difficulties

One of the biggest problems I had was finding a meaningful way to use structs in this program. The way I ended up using them was mostly just to fulfill the requirements. I also had a hard time finding a way to use enums and ended up just skipping out on them, any way I could think to use them would have just complicated the program more than it was worth doing. One way I thought of using them was for the cards, but characters already do that perfectly well as is. I will try to find a better way to use them for error checking or something in the next project. Such as having an enum assigned to a value and if that value is present after a certain action then there was an error in that action and I can return it to the main game loop for output to the user on what happened incorrectly.

1. Code

**Main.cpp**

#include <cstdlib>

#include <iostream>

#include <cmath>

#include <fstream>

#include "Cards.h"

char genCard(Game&);

void fillDeck(Game&);

void genHand(Cards&, Game&);

int chkVal(const Cards&);

void output(const std::string&, Game&);

void recap(Game&);

void expand(char\*, int&, const int&);

void hit(Cards&, Game&);

void prntHnd(const Cards&, Game&);

void cmpTrn(Cards&, Game&);

void myLog(const std::string&, Game&);

void BJGame(Game&);

int main() {

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

Game game;

game.moves.open("moves.binary", std::ios::out | std::ios::trunc);

game.moves.close();

BJGame(game);

return 0;

}

char genCard(Game &deck){

int size = deck.deckSize;

bool valid;

int random;

char card;

do{

valid = true;

random = rand() % size;// no + 1 because arrays start at 0

card = deck.deck[random];

if(card == '\*'){

valid = false;

}

else{

deck.deck[random] = '\*';

}

}while(!valid);

return card;

}

void genHand(Cards &hand, Game &game){

hand.size = 2;

hand.cards = new char[hand.size];

hand.cards[0] = genCard(game);

hand.cards[1] = genCard(game);

}

void fillDeck(Game &game){

int size = game.deckSize;

int count = 0;

game.deck = new char[size];

char array[13] = {'2','3','4','5','6','7','8','9','T','Q','K','J','A'};

for(int i = 0; i < 4; i++){

for(int j = 0; j < 13; j++){

game.deck[count] = array[j];

count++;

}

}

}

void expand(char \*arr, int &size, const int &inc){

char \*temp = new char[size + inc];

for(int i = 0; i < size; i++){

temp[i] = arr[i];

}

for(int j = size; j < size + inc; j++){

temp[j] = '\0';

}

arr = temp;

size += inc;

}

int chkVal(const Cards &hand){

int total = 0;

int size = hand.size;

int aceCnt = 0;

for(int i = 0; i < size; i++){

char temp = hand.cards[i];

switch(temp){

case '2': total += 2;break;

case '3': total += 3;break;

case '4': total += 4;break;

case '5': total += 5;break;

case '6': total += 6;break;

case '7': total += 7;break;

case '8': total += 8;break;

case '9': total += 9;break;

case 'T': total += 10;break;

case 'J': total += 10;break;

case 'Q': total += 10;break;

case 'K': total += 10;break;

case 'A': aceCnt += 1;break;

default: std::cout<<"Something is wrong in your chkVal function";

}

}

for(int i = 0; i < aceCnt; i++){

((total + 11) > 21)?(total += 1):(total += 11);

}

return total;

}

void output(const std::string &move, Game &game){

int size = move.length();

char \*temp = new char[size];

for(int i = 0; i < size; i++){

temp[i] = move[i];

}

game.moves.open("moves.binary", std::ios::out | std::ios::binary | std::ios::app);

game.moves.write(temp, size);

game.moves.close();

}

void recap(Game &game){

std::string recapMoves;

game.moves.open("moves.binary", std::ios::in | std::ios::binary);

game.moves.seekg(0, game.moves.end);// Puts seek at end and records length

int length = game.moves.tellg();// records length part

game.moves.seekg(0, game.moves.beg);// Puts seek back at begginging

char \*buf = new char[length];

game.moves.read(buf, length);

for(int i = 0; i < length; i++){

recapMoves += buf[i];

}

std::cout<<"The recap of last game is.. "<<std::endl<<std::endl;

std::cout<<recapMoves<<std::endl;

}

void hit(Cards &cards, Game &game){

expand(cards.cards, cards.size, 1);

cards.cards[cards.size - 1] = genCard(game); // put card in end of arr

}

void prntHnd(const Cards &hand, Game &game){

int value = chkVal(hand);

int size = hand.size;

std::string card;

myLog("Your hand contains: ", game);

for(int i = 0; i < size; i++){

card = hand.cards[i];

card += " ";

myLog(card, game);

}

myLog(" which has a value of - " + std::to\_string(value) + " \n", game);

}

void cmpTrn(Cards &hand, Game &game){

int val = 0;

do{

val = chkVal(hand);

if(val < 17){

hit(hand, game);

}

}while(val < 17);

}

void myLog(const std::string &in, Game &game){

std::cout<<in;

std::cout<<std::flush;

output(in, game);

}

void BJGame(Game &game){

Player player;

Player cmpter;

fillDeck(game);

genHand(player.hand,game);

genHand(cmpter.hand,game);

int playerVal = 0;

int cmpterVal = 0;

std::string input;

bool valid;

std::cout<<"Welcome to BlackJack!"<<std::endl;

std::cout<<"Standard rules, you start with two cards \n"

"if you have less than 21 you are allowed to hit \n"

"to reach a max value of 21 to win the game! But, \n"

"go over 21 and you lose! Cards with number values \n"

"are worth their number, cards with face values \n"

"excluding aces are worth 10 points. The ace card is worth \n"

"either 1 or 11 points, whichever is in your favor."<<std::endl;

std::cout<<"Are you ready to play?: y/n"<<std::endl;

std::cin>>input;

std::cout<<input[0]<<std::endl;

std::cin.ignore();

if(input[0] != 'y' && input[0] != 'Y'){

std::cout<<"Okay, cya later!";

exit(EXIT\_FAILURE);

}

input = "";

do{

valid = true;

input = "";

prntHnd(player.hand, game);

if(chkVal(player.hand) >= 21){

break;

}

std::cout<<"would you like to [h]it, or [s]tand?: ";

//myLog("would you like to [h]it, or [s]tand?: ", game);

std::cin>>input;

std::cout<<input[0]<<std::endl;

std::cin.ignore();

if(input[0] != 'h' && input[0] != 's'){

valid = false;

do{

std::cout<<"Invalid input try again: ";

std::cin>>input;

std::cout<<input[0]<<std::endl;

std::cin.ignore();

if(input[0] == 'h' || input[0] == 's'){

valid = true;

}

}while(!valid);

}

else if(input[0] == 's'){

cmpTrn(cmpter.hand, game);

}

else if(input[0] == 'h'){

hit(player.hand, game);

}

else{

std::cout<<"something went wrong ";

}

}while(input[0] == 'h');

playerVal = chkVal(player.hand);

cmpterVal = chkVal(cmpter.hand);

if(playerVal == 21){

myLog("Congratulations! You won \n", game);

}

else if(playerVal > 21){

myLog("Busted!! You lost since you went over 21 \n", game);

}

else if(cmpterVal > 21){

myLog("Dealer busted! You win! \n", game);

}

else if(playerVal > cmpterVal){

myLog("Congratulations! You won! the dealer had a hand worth " +

std::to\_string(cmpterVal) + "\n", game);

}

else if(playerVal == cmpterVal){

myLog("Draw! You and the dealer had the same value \n", game);

}

else{

myLog("You lose! The dealer had a value of " + std::to\_string(cmpterVal) + "\n", game);

}

do{

std::cout<<"Would you like a recap of the game you just played?: y/n"<<std::endl;

std::cin>>input;

std::cout<<input[0]<<std::endl;

std::cin.ignore();

if(input[0] != 'y' && input[0] != 'n'){

std::cout<<"Invalid choice, try again: "<<std::endl;

}

}while(input[0] != 'y' && input[0] != 'n');

if(input[0] == 'y'){

recap(game);

}

std::cout<<std::endl<<"Cya later!";

}

**Cards.h**

/\*

\* File: Cards.h

\* Author: Cory

\*

\* Created on July 15, 2018, 3:18 PM

\*/

#ifndef CARDS\_H

#define CARDS\_H

struct Game{

std::fstream moves;

char \*deck;

int deckSize = 52;

};

struct Cards{

char \*cards;

int size;

};

struct Player{

Cards hand;

int money;

};

#endif /\* CARDS\_H \*/