Project 1

<Blackjack>

CSC-17A-45398

Name: Juan Enriquez

Date: 7/15/2021

Introduction

Title: Blackjack

This is a simple card game of Blackjack which can have up to 3 players plus the dealer. So there are at least 2 or maximum of 4 players at a time.

The reason why I am doing this is because it is for my class project, and this is important for me to do because this project helps me use the concepts that we have gone over in class.

Summary

Project size: about 542 lines

The number of variables: 35

Number of structure objects: 4

The number of functions: 10

The number of Structures: 3

The number of class enums: 2

What this project is, is a game that utilizes the concepts that we have gone over until chapter 12 for this summer semester. For this game, it is relatively simple to play. The project creates a deck, then deals 2 cards to each player. There can only be 2 players at a time. After these two actions are done, a function from the project (which is named Dealer()) is called and this gives the option to draw more cards. The first person to make it to the number 21, or the player closet to 21, wins the game. Anything over would be a buster hand.

The amount of time that I sat down and worked on my project is about 6 days, about 5-7 hours each day. I had no idea where to start so I watched some YouTube videos and started from there and was left with about 91 lines of code for the first day. As time went on, I had no idea how to continue so I asked ideas from the people at the Lab on how to visualize doing this and went on to do more code. What took me a while to understand was that when you create a **structure array** within a **structure**, each **structure array** could access the contents of where the **structure array** is being initialized. Some concepts that were not reviewed in class were **class enumerators,** which has to be strong-typed:

Ex.

enum class{ int something…}

…something

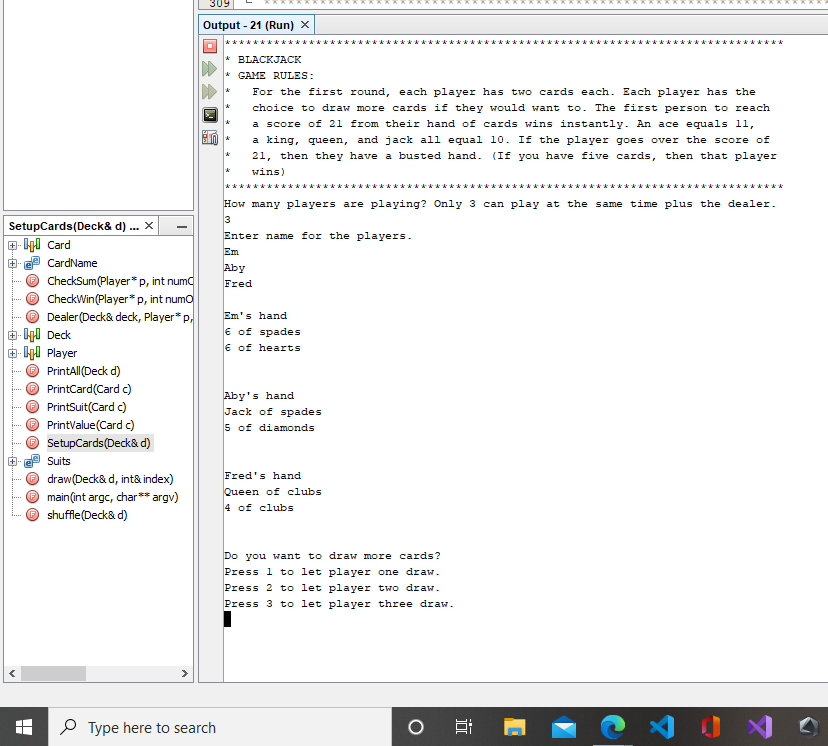
enum::something

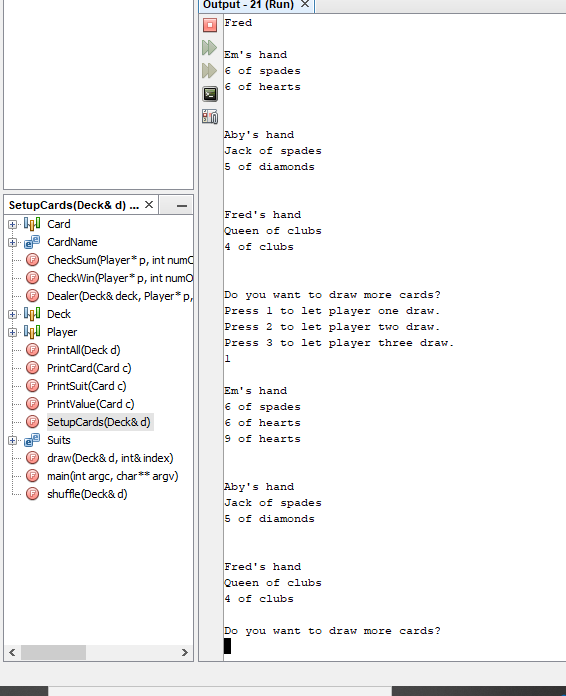
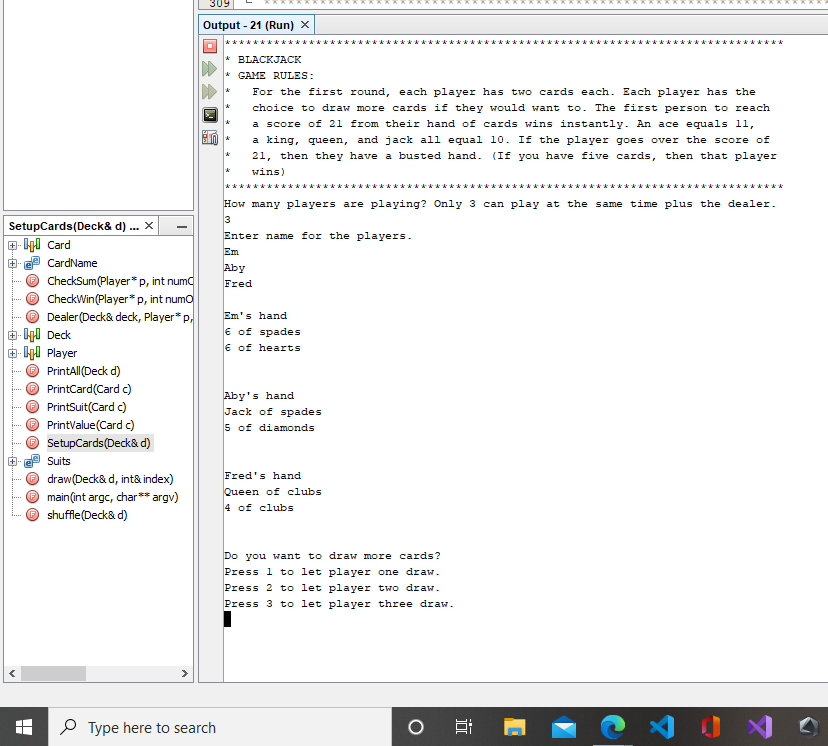
Although this project was a good learning experience, this project is not as completed as I would like, and I would like to add more functionality in the future. It works for now.

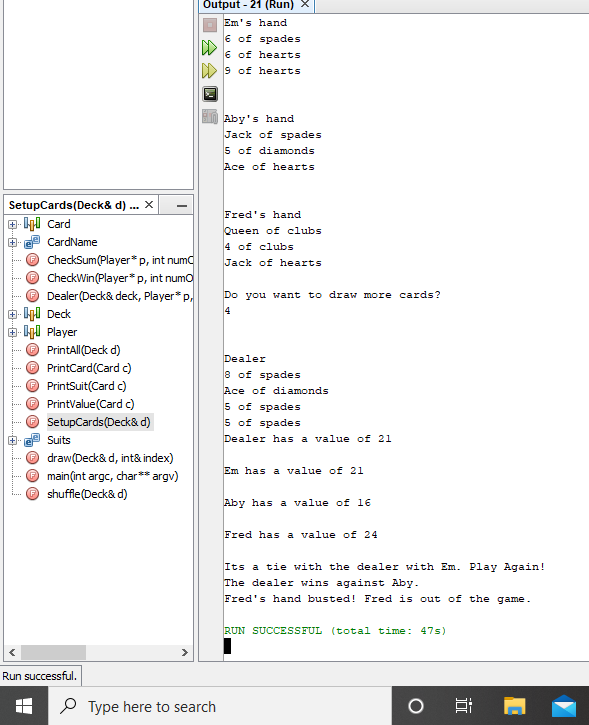
Description

I programmed my project so that everything that the user types, it would be transferred into the applied functions. What I pass to the functions are mostly structures or structure arrays. To go through the code, just put in the required input.

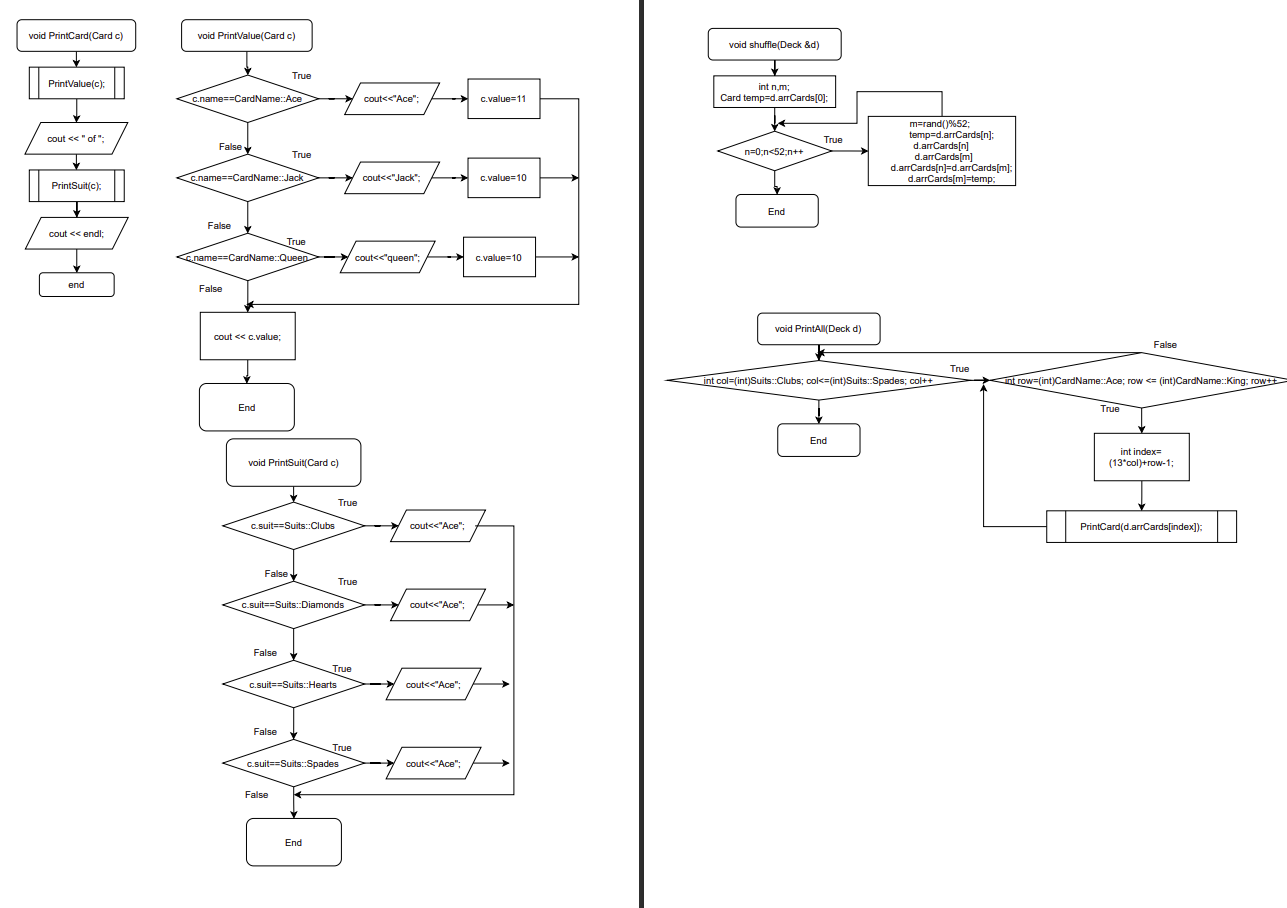
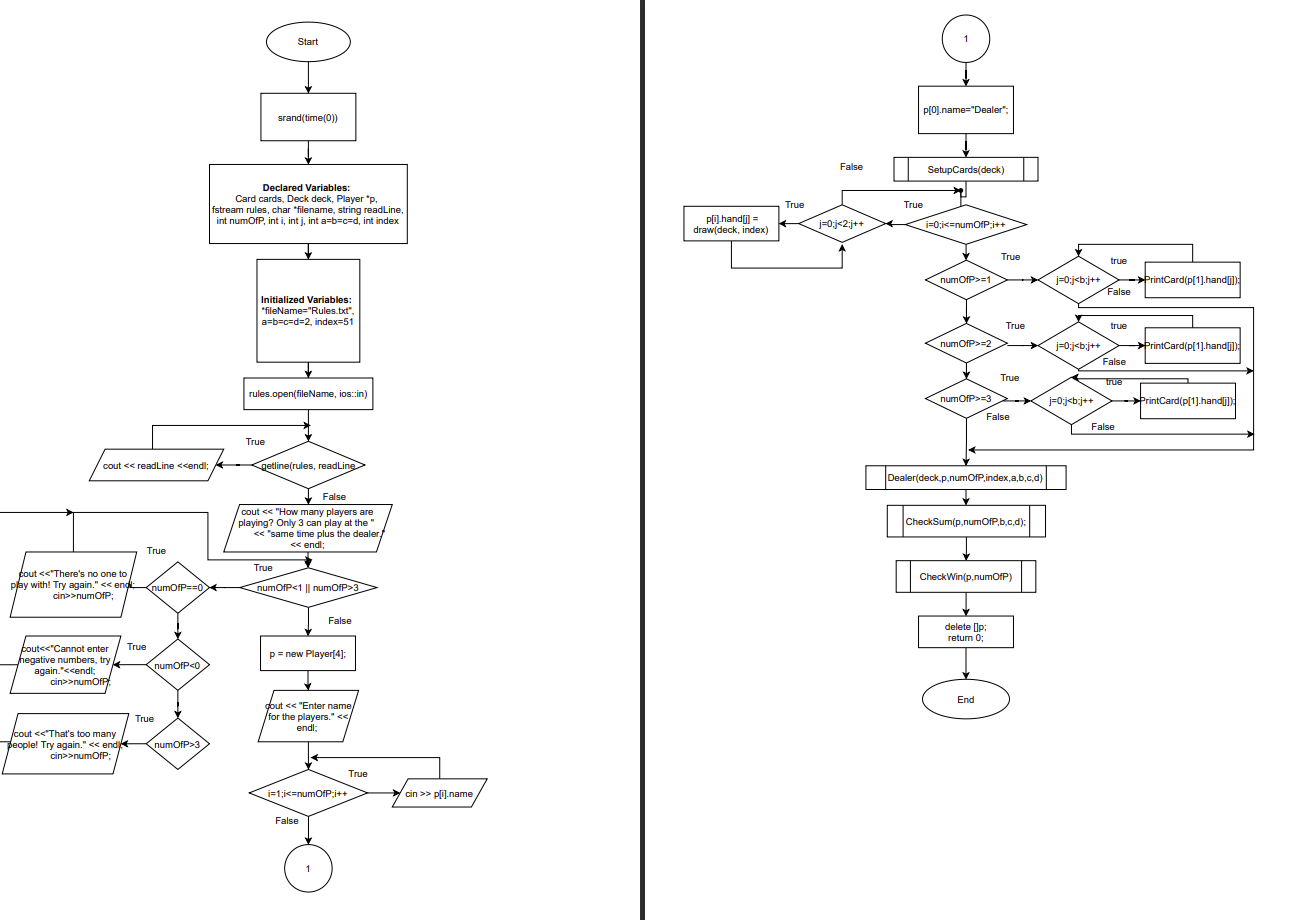
Graphical user interface, text, application

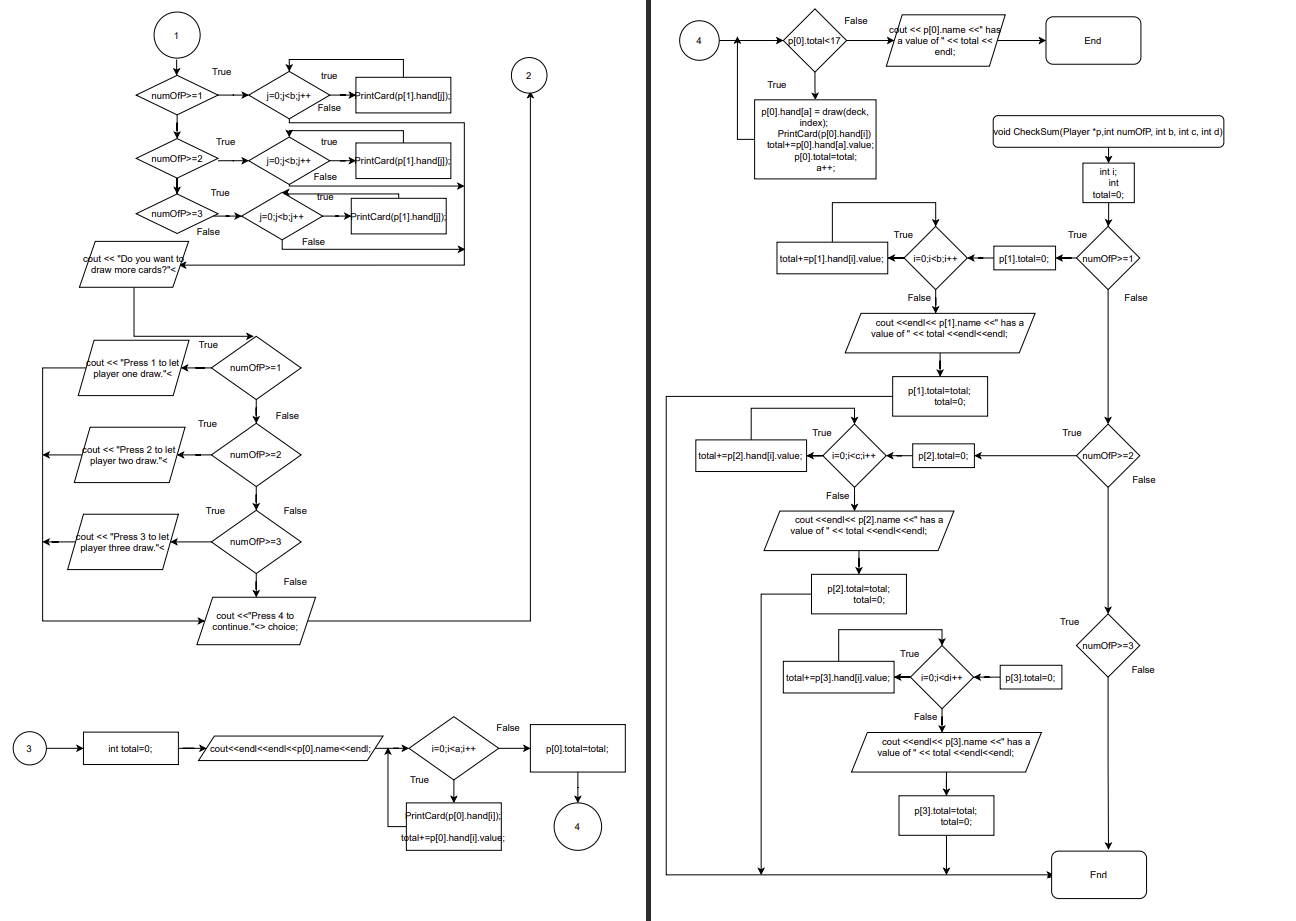
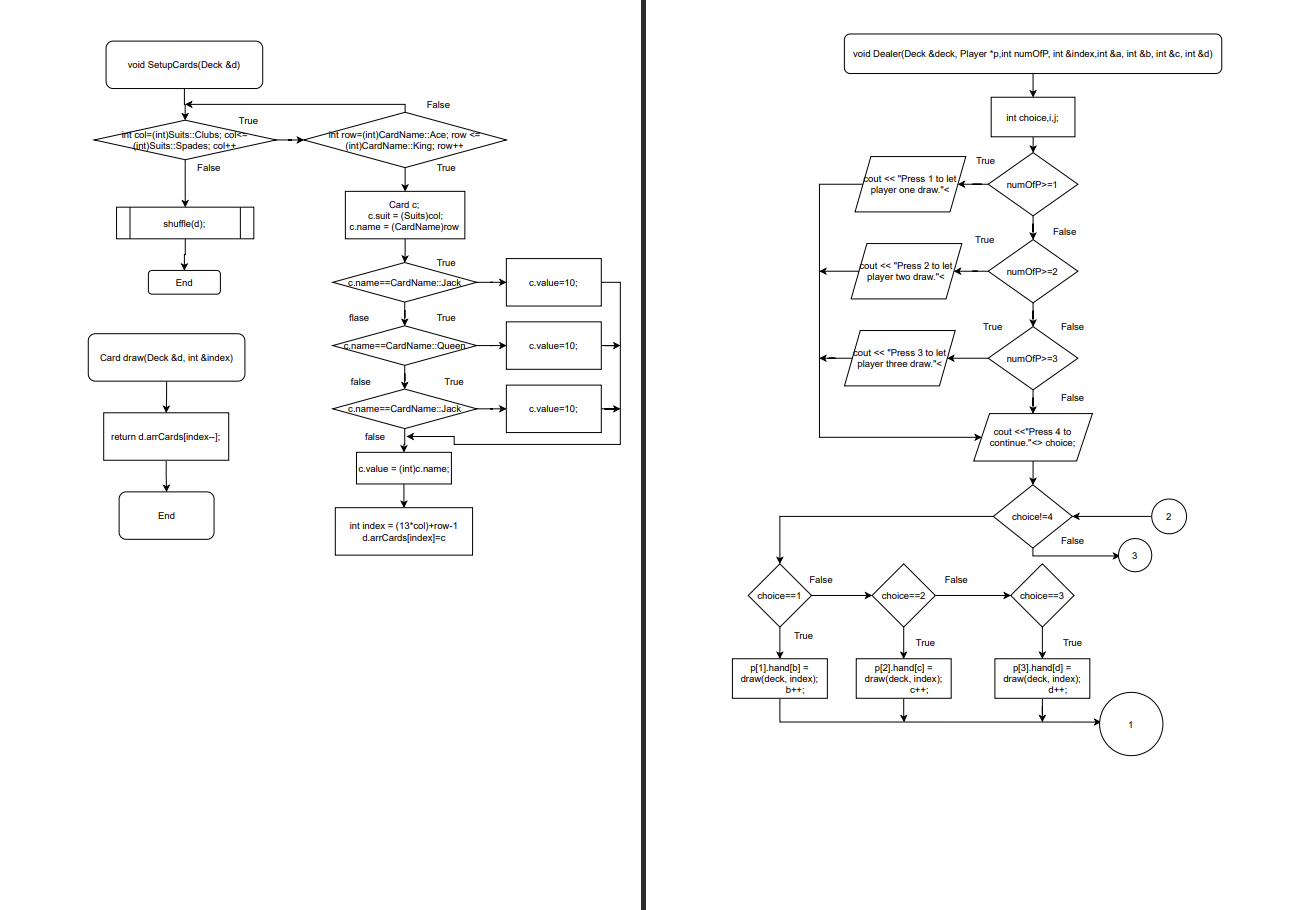
Description automatically generated

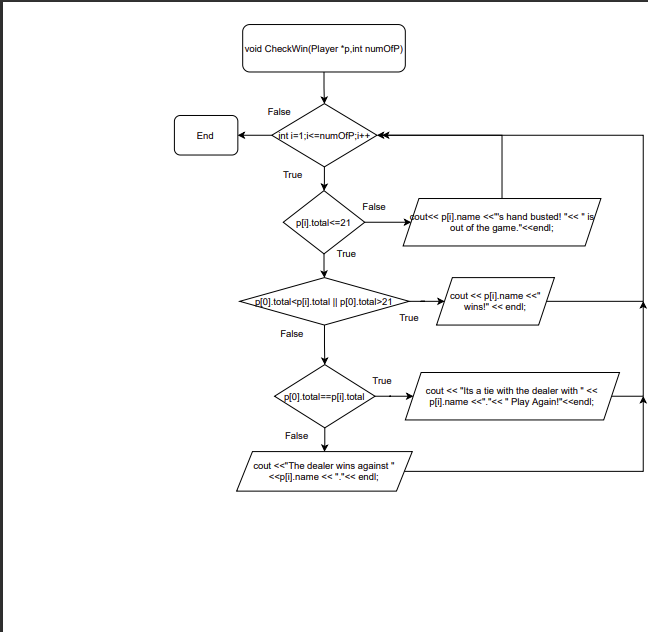
Graphical user interface, text, application

Description automatically generated

Flow Chart (Read from left to right):







Variables of note

|  |  |  |  |
| --- | --- | --- | --- |
| Type | Var Name | Description | Location |
| Integer | numOfP | Tells how many people are playing | int main() |
| index | Gives the position of the deck | int main(), PrintAll(), SetupCards(), draw(), Dealer() |
| enum | Suit | Has the contents of the card type | enum class Suits |
| CardName | Has the contents of the card values | enum class CardName |
| struct | Card | Holds the contents of each card | struct Card{}; //line 46 |
| Deck | Has an array of the Card structure that holds 52 structures | struct Deck{}; //line 55 |
| Player | Holds the contents of the info of the player | struct Player{}; //line 62 |
| srand | (time(0)) | Random generator | Int main()//line 86 |
| fstream | rules | Used to open the file Rules.txt | int main()//Line 94 |

**Concepts**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Chapter** | **Section** | Checklist | Location | | Comments |
| **9** |  | **Pointers/Memory Allocation** | |  |  |
|  | **1** | **Memory Addresses** | int main(), Line  **91** | |  |
|  | **2** | **Potinter Variables** | **int main(), Line 95** | |  |
|  | **3** | **Arrays/Pointers** | **int main(), Line 135** | |  |
|  | **7** | **Function Parameters** | **int main(), Line 78, Line188** | |  |
|  | **8** | **Memory Allocation** | **Int main(), Line 135, Line 195** | |  |
|  | **9** | **Return Parameters** | **Card draw(), Line 347** | |  |
| **10** |  | **Char Arrays and Strings** | |  |  |
|  | **3** | **C-Strings** | **int main(), Line 95** | |  |
|  | **7** | **Strings** | **int main(), Line 96** | |  |
| **11** |  | **Structured Data** | |  |  |
|  | **5** | **Arrays** | **int main(), Line 56** | |  |
|  | **6** | **Nested** | **int main(), Line 154** | |  |
|  | **7** | **Function Arguments** | **int main(), Line 188** | |  |
|  | **8** | **Function Return** | **Card draw(), Line 345** | |  |
|  | **9** | **Pointers** | **int main(), Line 91** | |  |
|  | **11** | **Enumeration** | **class enum CardName(), Line 20,**  **class enum Suits, Line27** | |  |
| **12** |  | **Binary Files** | |  |  |
|  | **2** | **Formatting** | **int main(), Line 107** | |  |
|  | **3** | **Function Parameters** | **int main(), Line 104** | |  |
|  | **5** | **Member Functions** |  | |  |
|  | **6** | **Multiple Files** | **Rules.txt** | | **This is a separate file that is included with the program file.** |
|  | **7** | **Binary Files** |  | |  |
|  | **8** | **Records with structures** | **struct Deck()** | |  |
|  | **9** | **Random access file** |  | |  |
|  | **10** | **Input/Output Simultaneous** | **Int main, Line 104** | |  |

**Pseudo code**

**Create Cards:**

Create variables to hole the cards type and value,

Insert them into a structure array

**Input Players:** Create a structure that hold the contents of each player

**Enter Main Program:**

Initialize variables to access structures,

Make a file that has the output for the rules,

Output file Rules.txt,

Ask user how many players they want,

while {

the amount is not correct, tell user that input is incorrect and to try again

},

Ask user for each players name,

**SetupCards** Function call that sets up the card deck,

Output the contents of each players hand,

**Dealer** Function gets called so that each player can draw cards, Dealer also draws,

**CheckSum** Function call gets called which checks the sum of each players hand,

**CheckWin** Function call used to check the win conditions of each player including Dealer

**Exit Program;**

**Create Functions for the program:**

**SetupCards(){**

Go through the contents of the structure array of cards and set the deck,

**Shuffle** function call to shuffle the deck

**}**

**Dealer(){**

Deal cards to each player in a menu. The dealer also draws until sum of hand reaches

17 or over

**}**

**CheckSum(){**

Function checks the sum of each players hand

**{**

**CheckWin(){**

Check each players hand sum and compare scores with the dealer

**}**

**Shuffle(){**

Shuffles the array of cards in the Deck structure array

**}**

**PrintCard(){**

Prints the contents of the contents the **Deck** structure array

**}**

**PrintSuit(){**

Defines the type of card

**}**

**PrintValue(){**

Defines the value of card

**}**

**Draw(){**

Draws a card,

Index decrements the deck position,

**}**

**References**

1. **Textbook**
2. **Lecture**
3. **YouTube** 
   1. (280) [C++ Tutorial #11.2] Ace Combinations - GameDev Prep Course – YouTube
   2. (280) [C++ Tutorial #11.1] Instantiating a Deck of Cards – YouTube
4. Reddit

**Program**

**/\***

**\* File: game.cpp**

**\* Author: Juan Enriquez**

**\* Purpose: To create a program that plays the Blackjack card game**

**\* Version: 12**

**\* Created on July 13, 2021, 6:03 PM**

**\*/**

**//System Libraries - Post Here**

**#include <iostream>**

**#include <string>**

**#include <fstream>**

**#include <iomanip>**

**#include <cstdlib>**

**#include <ctime>**

**using namespace std;**

**// after done with project, redo and put structs/enums into seperate files**

**enum class Suits{**

**Clubs,**

**Diamonds,**

**Hearts,**

**Spades,**

**};**

**enum class CardName{**

**Ace=1, // makes ace start at one in the enum**

**Two,**

**Three,**

**Four,**

**Five,**

**Six,**

**Seven,**

**Eight,**

**Nine,**

**Ten,**

**Jack,**

**Queen,**

**King**

**};**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*Card Structure**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**struct Card {**

**CardName name;**

**Suits suit;**

**int value;**

**};**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\* Deck Structure**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**struct Deck{**

**Card arrCards[52]; // make into a dynamically allocated structure array**

**};**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\* Player Structure**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**struct Player{**

**string name;**

**Card hand[20];**

**int total;**

**};**

**//User Libraries - Post Here**

**//Function Prototypes - Post Here**

**void PrintCard(Card);**

**void PrintValue(Card);**

**void PrintSuit(Card);**

**void shuffle(Deck &);**

**void PrintAll(Deck);**

**void SetupCards(Deck &);**

**Card draw(Deck &, int &); //returns drawn card**

**void Dealer(Deck &, Player \*,int ,int &, int &, int &, int &, int &);**

**void CheckSum(Player \*,int,int,int,int);**

**void CheckWin(Player \*,int);**

**//Execution Begins Here**

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

**//**

**// random generator seed**

**srand(time(0)); // required to make the random operator work**

**//**

**// Structure Variables**

**Card cards;**

**Deck deck;**

**Player \*p; // struct array of 4 to hold 4 players**

**//**

**// Variables**

**fstream rules; // set up to open file for da rules**

**char \*fileName="Rules.txt"; // c-string thing**

**string readLine;**

**int numOfP; // Number of players**

**int i; //counter**

**int j; //counter**

**int a=2,b=2,c=2,d=2;**

**int index=51;**

**//**

**// Open file to print out rules**

**rules.open(fileName, ios::in | ios::out); // opens a file and put it into input mode**

**while(getline(rules, readLine)) // while loop used to print out contents**

**{**

**cout << readLine <<endl;**

**}**

**rules.close(); //closes file**

**//**

**cout << "How many players are playing? Only 3 can play at the "**

**<< "same time plus the dealer." << endl;**

**cin >> numOfP;**

**//**

**//While statement tests user input for amount of players**

**while(numOfP<1 || numOfP>3){ // add more later so that it only accepts ints**

**if(numOfP==0)**

**{**

**cout <<"There's no one to play with! Try again." << endl;**

**cin>>numOfP;**

**}**

**else if(numOfP<0)**

**{**

**cout<<"Cannot enter negative numbers, try again."<<endl;**

**cin>>numOfP;**

**}**

**else if(numOfP>3)**

**{**

**cout <<"That's too many people! Try again." << endl;**

**cin>>numOfP;**

**}**

**}**

**//**

**// Create a dynamic struct for people**

**p = new Player[4];**

**//**

**// Input name for players**

**cout << "Enter name for the players." << endl;**

**for(i=1;i<=numOfP;i++) //for loop used to enter name to each Player**

**{ //structure array**

**cin >> p[i].name;**

**}**

**cout << endl;**

**p[0].name="Dealer"; // Sets the name for the Dealer**

**//**

**//function call to set the deck of cards**

**SetupCards(deck);**

**//**

**// for loop that draws cards for each player until there are 2 cards**

**for(i=0;i<=numOfP;i++)**

**{**

**for(j=0;j<2;j++)**

**{**

**p[i].hand[j] = draw(deck, index);**

**}**

**}**

**//**

**//Output the contents of each players hand**

**if(numOfP>=1)**

**{**

**cout<<p[1].name << "'s hand" << endl;**

**for(j=0;j<b;j++)**

**{**

**PrintCard(p[1].hand[j]);**

**}**

**cout<<endl;**

**}**

**if(numOfP>=2)**

**{**

**cout<<endl<<p[2].name << "'s hand" << endl;**

**for(j=0;j<c;j++)**

**{**

**PrintCard(p[2].hand[j]);**

**}**

**cout<<endl;**

**}**

**if(numOfP>=3)**

**{**

**cout<<endl<<p[3].name << "'s hand" << endl;**

**for(j=0;j<d;j++)**

**{**

**PrintCard(p[3].hand[j]);**

**}**

**cout<<endl;**

**}**

**//**

**// Function call so that players can draw more cards**

**Dealer(deck,p,numOfP,index,a,b,c,d);**

**//**

**// Function call to check the sum and win conditions**

**CheckSum(p,numOfP,b,c,d);**

**CheckWin(p,numOfP);**

**//delete allocated data here**

**delete []p;**

**// Exit stage right**

**return 0;**

**}**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\* PrintCard Function:**

**\* Will print out the card that is called.**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**void PrintCard(Card c)**

**{**

**PrintValue(c);**

**cout << " of ";**

**PrintSuit(c);**

**cout << endl;**

**}**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\* PrintValue Function:**

**\* Prints out the value**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**void PrintValue(Card c)**

**{**

**if(c.name==CardName::Ace)**

**{**

**cout<<"Ace";**

**c.value=11;**

**}**

**else if(c.name==CardName::Jack)**

**{**

**cout<<"Jack";**

**c.value=10;**

**}**

**else if(c.name==CardName::Queen)**

**{**

**cout<<"Queen";**

**c.value=10;**

**}**

**else if(c.name==CardName::King)**

**{**

**cout<<"King";**

**c.value=10;**

**}**

**else**

**{**

**cout << c.value;**

**}**

**}**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\* PrintSuit Function:**

**\* Prints out the suit to each respective card**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**void PrintSuit(Card c)**

**{**

**if (c.suit==Suits::Clubs)**

**{**

**cout << "clubs";**

**}**

**else if(c.suit==Suits::Diamonds)**

**{**

**cout << "diamonds";**

**}**

**else if(c.suit==Suits::Hearts)**

**{**

**cout << "hearts";**

**}**

**else if(c.suit==Suits::Spades)**

**{**

**cout << "spades";**

**}**

**}**

**//**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\* Shuffle Function:**

**\* Will shuffle the deck of cards**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**void shuffle(Deck &d)**

**{**

**int n,m;**

**Card temp=d.arrCards[0]; // makes temp start at the first Card struct**

**for (n=0;n<52;n++)**

**{**

**m=rand()%52; // gives a random number**

**temp=d.arrCards[n]; // switches the contents of d.arrCards[n] with what**

**// is in d.arrCards[m]**

**d.arrCards[n]=d.arrCards[m];**

**d.arrCards[m]=temp;**

**}**

**}**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\* PrintAll Function:**

**\* This function will print the cards of the deck(just a test case)**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**void PrintAll(Deck d)**

**{**

**for(int col=(int)Suits::Clubs; col<=(int)Suits::Spades; col++)**

**{**

**for(int row=(int)CardName::Ace; row <= (int)CardName::King; row++)**

**{**

**int index=(13\*col)+row-1; // accesses the contents of arrCards**

**PrintCard(d.arrCards[index]); // Prints contents in a card struct**

**}**

**}**

**//PrintCard(d.arrCards[0]); // this can call each individually**

**}**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\* SetupCards Function:**

**\* This function will initialize the each card**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**void SetupCards(Deck &d)**

**{**

**for(int col=(int)Suits::Clubs; col<=(int)Suits::Spades; col++)**

**{**

**for(int row=(int)CardName::Ace; row <= (int)CardName::King; row++)**

**{**

**Card c;**

**c.suit = (Suits)col; // same as Suits[col]**

**c.name = (CardName)row; // these two access the enum class**

**// This gives each name its value**

**if (c.name==CardName::Jack){**

**c.value=10;**

**}**

**else if(c.name==CardName::Queen){**

**c.value=10;**

**}**

**else if(c.name==CardName::King){**

**c.value=10;**

**}**

**else**

**{**

**c.value = (int)c.name;**

**}**

**int index = (13\*col)+row-1; // gives position in deck**

**d.arrCards[index]=c;**

**}**

**}**

**shuffle(d); //call shuffle function**

**}**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\* Draw Structure Function:**

**\* This function allows to draw a card from the deck of cards**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**Card draw(Deck &d, int &index)**

**{**

**return d.arrCards[index--]; //returns whole struct array**

**}**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\* Dealer Function:**

**\* This function allows player to draw additional cards from the deck.**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**void Dealer(Deck &deck, Player \*p,int numOfP, int &index,int &a,**

**int &b, int &c, int &d)**

**{**

**int choice,i,j;**

**cout << "\nDo you want to draw more cards?"<<endl;**

**// if statement used to control what should be outputted**

**if (numOfP>=1)**

**{**

**cout << "Press 1 to let player one draw."<<endl;**

**}**

**if (numOfP>=2)**

**{**

**cout << "Press 2 to let player two draw."<<endl;**

**}**

**if (numOfP==3)**

**{**

**cout << "Press 3 to let player three draw."<<endl;**

**}**

**cout <<"Press 4 to continue."<<endl;**

**cin >> choice;**

**//**

**while(choice!=4){**

**//**

**if(choice==1)**

**{**

**p[1].hand[b] = draw(deck, index);**

**b++;**

**}**

**else if(choice==2)**

**{**

**p[2].hand[c] = draw(deck, index);**

**c++;**

**}**

**else if(choice==3)**

**{**

**p[3].hand[d] = draw(deck, index);**

**d++;**

**}**

**else**

**{**

**cout<<"That is not a valid choice! Try again." << endl;**

**}**

**//**

**//Output the contents of each players hand with the if statement**

**if(numOfP>=1)**

**{**

**cout<<endl<<p[1].name << "'s hand" << endl;**

**for(j=0;j<b;j++)**

**{**

**PrintCard(p[1].hand[j]);**

**}**

**cout<<endl;**

**}**

**if(numOfP>=2)**

**{**

**cout<<endl<<p[2].name << "'s hand" << endl;**

**for(j=0;j<c;j++)**

**{**

**PrintCard(p[2].hand[j]);**

**}**

**cout<<endl;**

**}**

**if(numOfP>=3)**

**{**

**cout<<endl<<p[3].name << "'s hand" << endl;**

**for(j=0;j<d;j++)**

**{**

**PrintCard(p[3].hand[j]);**

**}**

**cout<<endl;**

**}**

**//**

**cout << "Do you want to draw more cards?"<<endl;**

**if (numOfP>=1)**

**{**

**cout << "Press 1 to let player one draw."<<endl;**

**}**

**if (numOfP>=2)**

**{**

**cout << "Press 2 to let player two draw."<<endl;**

**}**

**if (numOfP==3)**

**{**

**cout << "Press 3 to let player three draw."<<endl;**

**}**

**cout <<"Press 4 to continue."<<endl;**

**cin >> choice;**

**}**

**//**

**int total=0;**

**// calculates total for dealer**

**cout<<endl<<endl<<p[0].name<<endl;**

**for(i=0;i<a;i++)**

**{**

**PrintCard(p[0].hand[i]);**

**total+=p[0].hand[i].value;**

**}**

**//**

**p[0].total=total;**

**while(p[0].total<17) // should draw until busts or wins**

**{**

**p[0].hand[a] = draw(deck, index); //draws card**

**PrintCard(p[0].hand[i]);**

**total+=p[0].hand[a].value;**

**p[0].total=total;**

**a++;**

**}**

**cout << p[0].name <<" has a value of " << total << endl;**

**}**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\* CheckSum Function:**

**\* After the player(s) are done drawing cards form the Dealer function, this**

**\* function gets called to sum up the contents of each players hand.**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**void CheckSum(Player \*p,int numOfP, int b, int c, int d)**

**{**

**int i;**

**int total=0;**

**// if statement**

**if(numOfP>=1)**

**{**

**p[1].total=0;**

**for(i=0;i<b;i++)**

**{**

**total+=p[1].hand[i].value;**

**}**

**cout <<endl<< p[1].name <<" has a value of " << total <<endl<<endl;**

**p[1].total=total;**

**total=0;**

**}**

**if(numOfP>=2)**

**{**

**p[2].total=0;**

**for(i=0;i<c;i++)**

**{**

**total+=p[2].hand[i].value;**

**}**

**cout << p[2].name <<" has a value of " << total << endl<<endl;**

**p[2].total=total;**

**total=0;**

**}**

**if(numOfP>=3)**

**{**

**p[3].total=0;**

**for(i=0;i<d;i++)**

**{**

**total+=p[3].hand[i].value;**

**}**

**cout << p[3].name <<" has a value of " << total << endl<<endl;**

**p[3].total=total;**

**total=0;**

**}**

**}**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\* CheckWin Function:**

**\* After the CheckSum function is called, this function gets called to see who**

**\* won. If sum is over 21, the function will say who busted.**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**void CheckWin(Player \*p,int numOfP)**

**{**

**for(int i=1;i<=numOfP;i++)**

**{**

**if(p[i].total<=21)**

**{**

**if(p[0].total<p[i].total || p[0].total>21)**

**{**

**cout << p[i].name <<" wins!" << endl;//player wins**

**}**

**else if(p[0].total==p[i].total)**

**{**

**cout << "Its a tie with the dealer with " << p[i].name <<"."**

**<< " Play Again!"<<endl;//tie**

**}**

**else**

**{**

**cout <<"The dealer wins against "<<p[i].name << "."**

**<< endl;//Dealer Wins**

**}**

**}**

**else**

**{**

**cout<< p[i].name <<"'s hand busted! " <<p[i].name**

**<< " is out of the game."<< endl;**

**}**

**}**

**}**