**Project 2**

Title

**Blackjack Card Game**

Course

**CIS-17C**

Section

**47065**

Due Date

**June 7, 2023**

Author

**Daniel Chvat**

**Table of Contents**

Contents

[1 Introduction 3](#_Toc137057500)

[2 Project Statistics 3](#_Toc137057501)

[3 Concepts Map 3](#_Toc137057502)

[4 Major Classes 3](#_Toc137057503)

[5 Challenges Faced / Development 4](#_Toc137057504)

[6 Pseudocode 4](#_Toc137057505)

[7 UML Class Diagrams 6](#_Toc137057506)

[8 Main.cpp Partial Flowchart 8](#_Toc137057507)

[9 AbsPlayer.H Code 8](#_Toc137057508)

[10 Card.h Code 9](#_Toc137057509)

[11 Card.cpp Code 10](#_Toc137057510)

[12 Deck.h Code 14](#_Toc137057511)

[13 Deck.cpp Code 14](#_Toc137057512)

[14 Face.h Code 15](#_Toc137057513)

[15 File.h Code 15](#_Toc137057514)

[16 File.cpp Code 16](#_Toc137057515)

[17 Game.h Code 17](#_Toc137057516)

[18 Game.cpp Code 18](#_Toc137057517)

[19 MyVctr.h Code 18](#_Toc137057518)

[20 Player.h Code 21](#_Toc137057519)

[21 Player.cpp Code 23](#_Toc137057520)

[22 Stats.H Code 26](#_Toc137057521)

[23 Tree.H Code 26](#_Toc137057522)

[24 Tree.cpp Code 27](#_Toc137057523)

[25 Graph.h Code 32](#_Toc137057524)

[26 Main.cpp Code 33](#_Toc137057525)

[27 Sample Inputs / Outputs 43](#_Toc137057526)

[28 Versions 46](#_Toc137057527)

# Introduction

Game: Blackjack

The objective of blackjack is to get as close to the number 21 as possible without going over. The player is assigned a card with a specific value and decides whether they wish to “Hit” (take another card to increase their cards total value) or “Stay” (Keep the cards they currently have). Players can make bets on games and if they win they are typically paid out in a 3:2 ratio. ($200->$300).

Note: Project files can be found on GitHub using this link <https://github.com/DanielChvat/CIS-17C/tree/master/Project/Project1>

# Project Statistics

(Not Counting Linked List Class Since I took it off your GitHub)

|  |  |
| --- | --- |
|  |  |
| **Number of Lines including header file and Comments** | 1656 |
| **Number of Files** | 18 |
| **Number of Structures** | 2 |
| **Number of Functions in Main** | 13 |
| **Number of Classes** | 10 |

**Note:** This project is a modification of my final project from CIS-17A and CIS-5 and contains concepts outside of what was required for this project.

# Concepts Map

|  |  |
| --- | --- |
| **Concept** | **Line Number** |
| **Recursion** | 416-422 Main.cpp |
| **Recursive Sorts** | 416-422 Main.cpp |
| **Hashing** | 236 Main.cpp |
| **Trees** | 249 – 251 Main.cpp 1-51 Tree.h |
| **Graphs** | 275 – 280 Main.cpp 1-54 Graph.h |

# Major Classes

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Function | Length in Lines | Instance Name in Main |
| AbsPlayer.h | Abstract Player Class | 37 | Never Instantiated in main used in other classes |
| Card.h | Card Class to be inherited in Deck Class for | 55 | Never Instantiated in main used in other classes |
| Deck.h | Holds Information about the deck the player draws such as card values | 26 | deck |
| Face.h | Enum class to help when converting card faces to integer numebrs | 33 | Never instantiated in main used in other classes |
| File.h | Holds information about a files name as well as a pointer to the fstream file. | 42 | file |
| Game.h | Holds basic information about the game such as the players | 42 | game |
| MyVctr.h | Custom Vector Class to hold precomputed card values | 139 | Never instantiated in main used in other classes |
| Player.h | Holds information about the player such as their name, money, bet, etc | 64 | player |
| Tree.h | AVL Tree Class | 51 | tree |
| Graph.h | Graph Class with DFS search | 54 | g |
| LinkedList.h | Linked List Class | 172 (These Lines were not counted towards project total because the class was taken off your GitHub) | hMap |

# Challenges Faced / Development

The main challenge I faced was recursion as I found it somewhat confusing at times especially with the quick sort algorithm. However, by watching the call stack on a debugger I was able to eventually get algorithms requiring recursion to work.

# Pseudocode

*Ask player to input their name*

*Output Blackjack Card Game*

*Ask player how much money they wish to play with*

*Ensure player is entering valid amount of money*

*Generate Deck of Cards to files*

*Open Cards and Suits files*

*Tell Player how much money they currently have*

*Ask player how much they wish to bet on the game*

*If the games index is greater than the number of cards in the deck*

*Ensure bet amount is valid*

*Get card from card file for player*

*Output card to player*

*Output players total card value*

*Else*

*Tell the player there are no cards left to draw from*

*Tell the player the dealers shown card value*

*Ask the player if they wish to hit or stay*

*If the player wishes to hit grab a new card from the deck, display the card to the player, and add it onto their total card value*

*Keep prompting for hits until player says no*

*Check if player has won*

*If player has won*

*pay out bet at 3:2 ratio*

*Tell the player they have won*

*Tell the player their total card value*

*Tell the player their new balance*

*Tell the player the % increase in their balance*

*Else if player card value is equal to dealer card value or both player and dealer card values are over 21*

*Output Dealers Card Value*

*Output Nobody Wins*

*Else*

*Subtract bet from players money*

*Tell player they lost*

*Tell player their card value and the dealers card value*

*Tell player their new balance*

*Tell player the % decrease in their money*

*If player still has money left*

*Ask if player wants to play again*

*If player wants to play again*

*Run the game again*

*Else*

*Ask player if they want to see info about the game*

*Output the cards*

*Ask player if they want to search for a specific card in deck*

*While player wants to search*

*Search for the card*

*Ask if they want to search again*

*Display the suits, sorted suits, sorted cards*

*Display the values of each card in the deck*

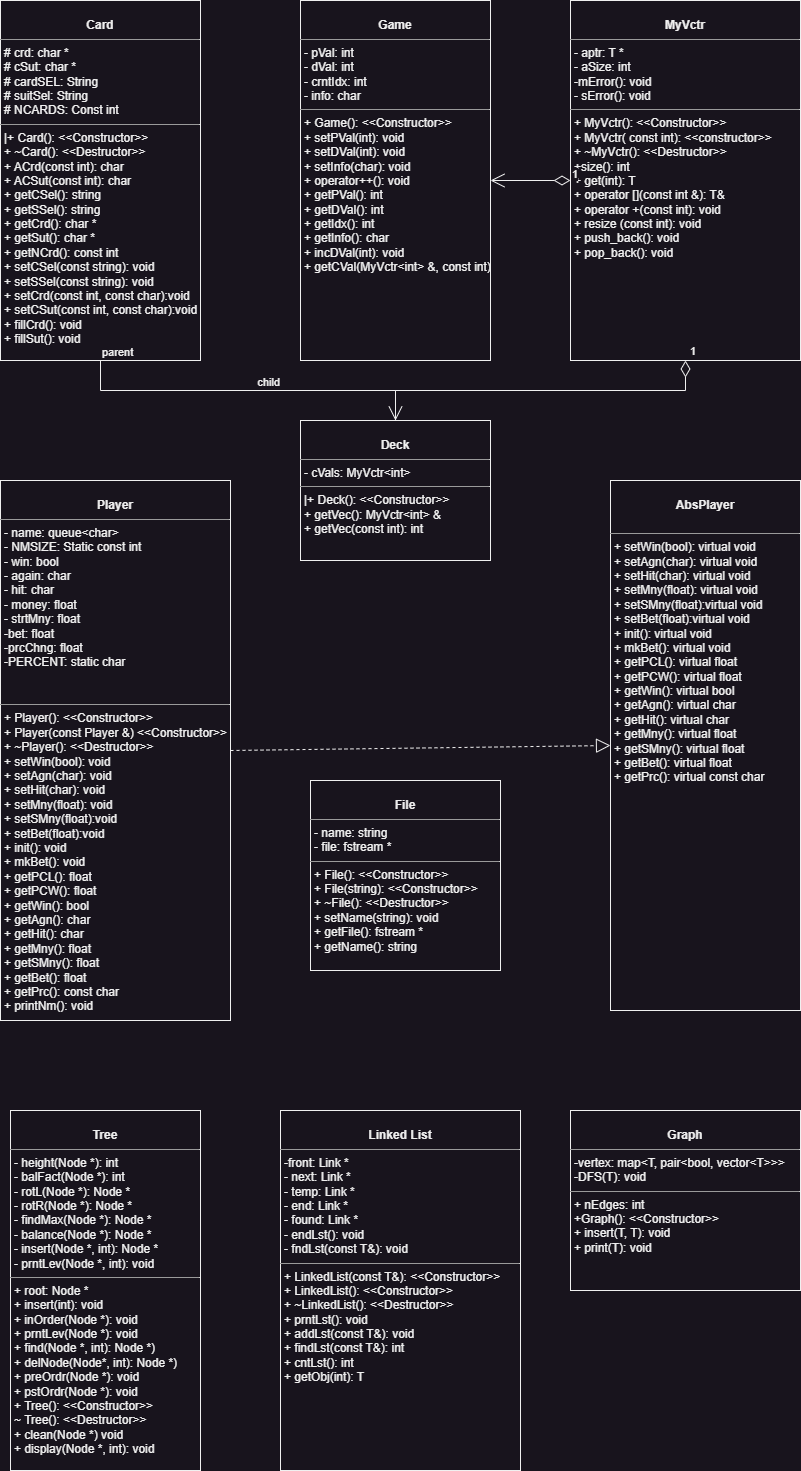
*Display sorted values of each card in the deck*

*Display unique card faces in deck*

*Display unique suits in deck  
Output Thanks for Playing [Players Name entered Earlier]*

# UML Class Diagrams

Can be found in project folder under filename: UML\_Class\_Diagrams\_Project\_2.drawio.pdf



# Main.cpp Partial Flowchart

Partial Flowchart of main.cpp can be seen in project folder under file name: Project\_2\_PartialFlowChart.drawio.pdf.

**Note**: Since this flowchart is only partially finished it has some sections which lead nowhere (for example a false if statement going to page 5A when 5A doesn’t exist)

# AbsPlayer.H Code

/\*

\* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license

\* Click nbfs://nbhost/SystemFileSystem/Templates/cppFiles/file.h to edit this template

\*/

/\*

\* File: AbsPlayer.h

\* Author: Daniel

\*

\* Created on December 15, 2022, 7:05 PM

\*/

#ifndef ABSPLAYER\_H

#define ABSPLAYER\_H

class AbsPlayer{

public:

virtual void setWin(bool)=0;

virtual void setAgn(char)=0;

virtual void setHit(char)=0;

virtual void setMny(float)=0;

virtual void setSMny(float)=0;

virtual void setBet(float)=0;

virtual void init()=0;

virtual float getPCL()=0;

virtual float getPCW()=0;

virtual bool getWin()=0;

virtual char getAgn()=0;

virtual char getHit()=0;

virtual float getMny()=0;

virtual float getSMny()=0;

virtual float getBet()=0;

virtual char getPrc() const=0;

virtual void mkBet()=0;

};

#endif /\* ABSPLAYER\_H \*/

# Card.h Code

/\*

\* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license

\* Click nbfs://nbhost/SystemFileSystem/Templates/cppFiles/file.h to edit this template

\*/

/\*

\* File: Card.h

\* Author: Daniel

\*

\* Created on December 15, 2022, 1:37 PM

\*/

#ifndef CARD\_H

#define CARD\_H

#include <string>

#include <fstream>

using namespace std;

class Card{

protected:

char \*crd;

char \*cSut;

string cardSel,//Card at a specific index in Cards.dat

suitSel;//Suit at a specific index in Suits.dat

const int NCARDS=52;//Number of Cards In the Deck

public:

//Constructor

Card();

//Destructor

~Card();

//Accessor Functions

char ACrd(const int i){return crd[i];}

char ACSut(const int i){return cSut[i];}

string getCSel(){return cardSel;}

string getSSel(){return suitSel;}

char \*getCrd(){return crd;}

char \*getSut(){return cSut;}

string trnslte(bool, const int i);

void genCrds();

int getVal(const int);

//Mutator Functions

void setCSel(const string c){cardSel=c;}

void setSSel(const string s){suitSel=s;}

void setCrd(const int i, const char n){crd[i]=n;}

void setCSut(const int i, const char n){cSut[i]=n;}

int getNCrd() const {return NCARDS;}

void fillCrd();

void fillSut();

};

#endif /\* CARD\_H \*/

# Card.cpp Code

/\*

\* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license

\* Click nbfs://nbhost/SystemFileSystem/Templates/cppFiles/file.cc to edit this template

\*/

#include "Card.h"

#include "Face.h"

#include <iostream>

#include <cstdlib>

#include <ctime>

#include <fstream>

using namespace std;

Card::Card(){

crd=new char[NCARDS];

cSut=new char[NCARDS];

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

}

Card::~Card(){

delete []crd;

delete []cSut;

}

void Card::fillCrd(){

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

crd[i]=cardSel[i];

}

}

void Card::fillSut(){

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

cSut[i]=suitSel[i];

}

}

string Card::trnslte(bool c, const int i){

if(c){

string card;

if(crd[i]=='A')card="Ace";

if(crd[i]=='2')card="2";

if(crd[i]=='3')card="3";

if(crd[i]=='4')card="4";

if(crd[i]=='5')card="5";

if(crd[i]=='6')card="6";

if(crd[i]=='7')card="7";

if(crd[i]=='8')card="8";

if(crd[i]=='9')card="9";

if(crd[i]=='T')card="10";

if(crd[i]=='Q')card="Queen";

if(crd[i]=='K')card="King";

if(crd[i]=='J')card="Jack";

return card;

}else{

string suit;

suit=cSut[i]=='H'?"Hearts":

cSut[i]=='D'?"Diamonds":

cSut[i]=='C'?"Clubs":

cSut[i]=='S'?"Spades":"";

return suit;

}

}

void Card::genCrds(){

string c="";

string s="";

for(unsigned char card=0; card<NCARDS;card++){

int cardVal=rand()%13+1;

//Check to see if the random number generated matches a specific number

switch(cardVal){

case 1:{

c+="A";

unsigned char suit=rand()%4+1;

if(suit==1)s+="H";

else if(suit==2)s+="D";

else if(suit==3)s+="C";

else s+="S";

break;}

case 2:{

c+="2";

unsigned char suit=rand()%4+1;

if(suit==1)s+="H";

else if(suit==2)s+="D";

else if(suit==3)s+="C";

else s+="S";

break;}

case 3:{

c+="3";

unsigned char suit=rand()%4+1;

if(suit==1)s+="H";

else if(suit==2)s+="D";

else if(suit==3)s+="C";

else s+="S";

break;}

case 4:{

c+="4";

unsigned char suit=rand()%4+1;

if(suit==1)s+="H";

else if(suit==2)s+="D";

else if(suit==3)s+="C";

else s+="S";

break;}

case 5:{

c+="5";

unsigned char suit=rand()%4+1;

if(suit==1)s+="H";

else if(suit==2)s+="D";

else if(suit==3)s+="C";

else s+="S";

break;}

case 6:{

c+="6";

unsigned char suit=rand()%4+1;

if(suit==1)s+="H";

else if(suit==2)s+="D";

else if(suit==3)s+="C";

else s+="S";

break;}

case 7:{

c+="7";

unsigned char suit=rand()%4+1;

if(suit==1)s+="H";

else if(suit==2)s+="D";

else if(suit==3)s+="C";

else s+="S";

break;}

case 8:{

c+="8";

unsigned char suit=rand()%4+1;

if(suit==1)s+="H";

else if(suit==2)s+="D";

else if(suit==3)s+="C";

else s+="S";

break;}

case 9:{

c+="9";

unsigned char suit=rand()%4+1;

if(suit==1)s+="H";

else if(suit==2)s+="D";

else if(suit==3)s+="C";

else s+="S";

break;}

case 10:{

c+="T";

unsigned char suit=rand()%4+1;

if(suit==1)s+="H";

else if(suit==2)s+="D";

else if(suit==3)s+="C";

else s+="S";

break;}

case 11:{

c+="K";

unsigned char suit=rand()%4+1;

if(suit==1)s+="H";

else if(suit==2)s+="D";

else if(suit==3)s+="C";

else s+="S";

break;}

case 12:{

c+="Q";

unsigned char suit=rand()%4+1;

if(suit==1)s+="H";

else if(suit==2)s+="D";

else if(suit==3)s+="C";

else s+="S";

break;}

case 13:{

c+="J";

unsigned char suit=rand()%4+1;

if(suit==1)s+="H";

else if(suit==2)s+="D";

else if(suit==3)s+="C";

else s+="S";

break;}

}

}

cardSel=c;

suitSel=s;

fillCrd();

fillSut();

}

int Card::getVal(const int index){

Face value;

value=crd[index]=='A'?Face::Ace:

crd[index]=='2'?Face::Two:

crd[index]=='3'?Face::Three:

crd[index]=='4'?Face::Four:

crd[index]=='5'?Face::Five:

crd[index]=='6'?Face::Six:

crd[index]=='7'?Face::Seven:

crd[index]=='8'?Face::Eight:

crd[index]=='9'?Face::Nine:

crd[index]=='T'?Face::Ten:

crd[index]=='Q'?Face::Queen:

crd[index]=='K'?Face::King:

crd[index]=='J'?Face::Jack:value;

return (int)value;

}

# Deck.h Code

/\*

\* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license

\* Click nbfs://nbhost/SystemFileSystem/Templates/cppFiles/file.h to edit this template

\*/

/\*

\* File: Deck.h

\* Author: Daniel

\*

\* Created on December 16, 2022, 4:05 PM

\*/

#ifndef DECK\_H

#define DECK\_H

#include "Card.h"

#include "MyVctr.h"

class Deck:public Card{

private:

MyVctr<int> cVals;

public:

Deck();

MyVctr<int> &getVec(){return cVals;}

int getVec(const int i){return cVals[i];}

};

#endif /\* DECK\_H \*/

# Deck.cpp Code

/\*

\* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license

\* Click nbfs://nbhost/SystemFileSystem/Templates/cppFiles/file.cc to edit this template

\*/

#include "Deck.h"

#include "MyVctr.h"

Deck::Deck(){

Card::genCrds();

cVals.resize(52);

for(int i=0; i<Card::NCARDS; i++){

cVals[i]=Card::getVal(i);

}

}

# Face.h Code

/\*

\* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license

\* Click nbfs://nbhost/SystemFileSystem/Templates/cppFiles/file.h to edit this template

\*/

/\*

\* File: Face.h

\* Author: Daniel

\*

\* Created on December 15, 2022, 4:26 PM

\*/

#ifndef FACE\_H

#define FACE\_H

enum class Face {

Ace = 1,

Two = 2,

Three = 3,

Four = 4,

Five = 5,

Six = 6,

Seven = 7,

Eight = 8,

Nine = 9,

Ten = 10,

Queen = 10,

King = 10,

Jack = 10,

lAce = 11

};

#endif /\* FACE\_H \*/

# File.h Code

/\*

\* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license

\* Click nbfs://nbhost/SystemFileSystem/Templates/cppFiles/file.h to edit this template

\*/

/\*

\* File: File.h

\* Author: Daniel

\*

\* Created on December 16, 2022, 1:42 PM

\*/

#ifndef FILE\_H

#define FILE\_H

#include <string>

#include <fstream>

using namespace std;

class File{

private:

string name;

fstream \*file;

public:

//Default Constructor

File();

//Constructor

File(string);

//Destructor

~File(){delete file;}

//Mutator Functions

void setName(string n ){name=n;}

//Accessor Functions

fstream \*getFile(){return file;}

string getName(){return name;}

};

#endif /\* FILE\_H \*/

# File.cpp Code

/\*

\* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license

\* Click nbfs://nbhost/SystemFileSystem/Templates/cppFiles/file.cc to edit this template

\*/

#include "File.h"

//Default Constructor

File::File(){

file = new fstream;

name = "";

}

//Constructor

File::File(string n){

name = n;

file = new fstream;

}

# Game.h Code

/\*

\* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license

\* Click nbfs://nbhost/SystemFileSystem/Templates/cppFiles/file.h to edit this template

\*/

/\*

\* File: Game.h

\* Author: Daniel

\*

\* Created on December 16, 2022, 2:23 PM

\*/

#ifndef GAME\_H

#define GAME\_H

#include "MyVctr.h"

class Game{

private:

int pVal;

int dVal;

int crntIdx;

char info;

public:

//Default Constructor

Game();

//Mutator Functions

void setPVal(int n){pVal=n;}

void setDVal(int n){dVal=n;}

void setInfo(char i){info = i;}

//Overloaded Operator

void operator++();

//Accessor Functions

int getPVal(){return pVal;}

int getDVal(){return dVal;}

int getIdx(){return crntIdx;}

char getInfo(){return info;}

void incDVal(int n){dVal+=n;}

int getCVal(MyVctr<int> &, const int);

};

#endif /\* GAME\_H \*/

# Game.cpp Code

/\*

\* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license

\* Click nbfs://nbhost/SystemFileSystem/Templates/cppFiles/file.cc to edit this template

\*/

#include "Game.h"

#include "MyVctr.h"

#include <string>

using namespace std;

Game::Game(){

pVal=0;

dVal=0;

crntIdx=0;

info='\0';

}

int Game::getCVal(MyVctr<int> &cards, const int prvVal){

int value=0;

for(int i=0; i<crntIdx+1;i++){

value+=cards[i];

}

return value-prvVal;

}

void Game::operator++(){

if(crntIdx+1 > 52){

string error = "Sorry We Ran Out Of Cards";

throw error;

}

else ++crntIdx;

}

# MyVctr.h Code

/\*

\* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license

\* Click nbfs://nbhost/SystemFileSystem/Templates/cppFiles/file.h to edit this template

\*/

/\*

\* File: MyVctr.h

\* Author: Daniel

\*

\* Created on December 15, 2022, 4:04 PM

\*/

#ifndef MYVCTR\_H

#define MYVCTR\_H

#include <iostream>

#include <new>

#include <cstdlib>

using namespace std;

template <class T>

class MyVctr{

private:

T \*aptr; //To point to the allocated array

int aSize; //Number of elements in the array

void mError(); //Handles Memory Allocation Errors

void sError(); //Handles Subscripts out of range

public:

//Default Constructor

MyVctr()

{aptr=0; aSize=0;}

//Constructor Declaration

MyVctr(const int);

//Destructor declaration

~MyVctr();

//Accessor to return the array Size

int size()

{return aSize;}

//Accessor to return a specific element

T get(int);

//Overloaded [] operator declaration

T &operator[] (const int &);

//Overloaded + operator declaration

void operator+(const int);

void resize(const int);

//Push Function

void push\_back(T);

void pop\_back();

};

template <class T>

MyVctr<T>::MyVctr(const int s){

aSize=s;

//Allocate Memory

try{

aptr=new T[s];

}

catch (bad\_alloc){

mError();

}

//Initialize the Array

for(int i=0; i<aSize; i++)aptr[i]=0;

}

template <class T>

MyVctr<T>::~MyVctr(){

if(aSize>0)delete []aptr;

}

template <class T>

void MyVctr<T>::mError(){

cout<<"ERROR: Cannot allocate Memory!"<<endl;

exit(EXIT\_FAILURE);

}

template <class T>

void MyVctr<T>::sError(){

cout<< "ERROR: Subscript out of range!"<<endl;

exit(EXIT\_FAILURE);

}

template <class T>

T MyVctr<T>::get(int sub){

if(sub<0 || sub>=aSize)sError();

return aptr[sub];

}

template <class T>

T &MyVctr<T>::operator [](const int &sub){

if(sub<0 || sub>=aSize)sError();

return aptr[sub];

}

template <class T>

void MyVctr<T>::push\_back(T data){

T \*temp = aptr;

++aSize;

delete []aptr;

aptr=new T[aSize];

for(int i=0; i<aSize-1; i++)aptr[i]=temp[i];

aptr[aSize-1]=data;

delete []temp;

}

template <class T>

void MyVctr<T>::pop\_back(){

T \*temp = aptr;

--aSize;

delete []aptr;

aptr=new T[aSize];

for(int i=0; i<aSize; i++)aptr[i]=temp[i];

delete []temp;

}

template <class T>

void MyVctr<T>::resize(const int size){

aSize=size;

aptr=new T[size];

}

template <class T>

void MyVctr<T>::operator+(const int n){

if(n>0){

T \*temp = aptr;

aSize+=n;

delete []aptr;

aptr=new T[aSize];

for(int i=0; i<aSize-n; i++)aptr[i]=temp[i];

delete []temp;

}

}

#endif /\* MYVCTR\_H \*/

# Player.h Code

/\*

\* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license

\* Click nbfs://nbhost/SystemFileSystem/Templates/cppFiles/file.h to edit this template

\*/

/\*

\* File: Player.h

\* Author: Daniel

\*

\* Created on December 15, 2022, 6:58 PM

\*/

#ifndef PLAYER\_H

#define PLAYER\_H

#include "AbsPlayer.h"

class Player:public AbsPlayer{

private:

queue<char> name;

bool win;

char again;

char hit;

float money;

float strtMny;

float bet;

float prcChng;

static char PERCENT;

public:

//Default Constructor

Player();

//Copy Constructor

Player(const Player &);

//Destructor

~Player();

//Mutator Functions

void setWin(bool);

void setAgn(char);

void setHit(char);

void setMny(float);

void setSMny(float);

void setBet(float);

void init();

//Accessor Functions

float getPCL();

float getPCW();

bool getWin();

char getAgn();

char getHit();

float getMny();

float getSMny();

float getBet();

char getPrc() const;

void mkBet();

void printNm();

};

#endif /\* PLAYER\_H \*/

# Player.cpp Code

/\*

\* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license

\* Click nbfs://nbhost/SystemFileSystem/Templates/cppFiles/file.cc to edit this template

\*/

#include "Player.h"

#include "iostream"

using namespace std;

//Default Constructor

Player::Player(){

init();

}

//Copy Constructor

Player::Player(const Player &p){

name=p.name;

win=p.win;

again=p.again;

hit=p.hit;

money=p.money;

strtMny=p.strtMny;

bet=p.bet;

prcChng=p.prcChng;

}

//Destructor

Player::~Player(){

}

//Mutator Functions

void Player::setWin(bool w){

win=w;

}

void Player::setAgn(char a){

again=a;

}

void Player::setHit(char h){

hit=h;

}

void Player::setMny(float m){

money=m;

}

void Player::setSMny(float s){

strtMny=s;

}

void Player::setBet(float b){

bet=b;

}

void Player::mkBet(){

money-=bet;

}

void Player::printNm(){

queue<char> n = name;

while(!n.empty()){

cout<<n.front();

n.pop();

}

}

void Player::init(){

win=false;

again='\0';

hit='\0';

money=0;

strtMny=0;

bet=0;

prcChng=0;

cout<<"What is Your Name"<<endl;

string temp;

cin>>temp;

for(int i=0; i<temp.size(); i++)name.push(temp[i]);

//Ask Player to input amount of money they wish to play with

cout<<"Blackjack Card Game"<<endl;

cout<<"Input the Amount of Money in Dollars You Wish to Play With"<<endl;

cin>>money;

strtMny=money;

//Ensure the player is not entering negative amounts of money

while(money<0){

cout<<money<<" Is Not a Valid Amount of Money"<<endl;

cout<<"Input the Amount of Money You Wish to Play With"<<endl;

cin>>money;

strtMny=money;

}

}

//Accessor Functions

float Player::getPCW(){

money=money +(bet/2)\*3;

prcChng=money/strtMny\*PERCENT;

strtMny=money;

return prcChng;

}

float Player::getPCL(){

money-=bet;

prcChng=money/strtMny\*PERCENT;

strtMny=money;

return prcChng;

}

char \*Player::getName(){

return name;

}

bool Player::getWin(){

return win;

}

char Player::getAgn(){

return again;

}

char Player::getHit(){

return hit;

}

float Player::getMny(){

return money;

}

float Player::getSMny(){

return strtMny;

}

float Player::getBet(){

return bet;

}

int Player::getNMS() const {

return NMSIZE;

}

char Player::getPrc() const{

return PERCENT;

}

//Static Members

char Player::PERCENT=100;

const int Player::NMSIZE = 81;

# Stats.H Code

/\*

\* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license

\* Click nbfs://nbhost/SystemFileSystem/Templates/cppFiles/file.h to edit this template

\*/

/\*

\* File: Stats.h

\* Author: Daniel

\*

\* Created on October 27, 2022, 3:40 PM

\*/

#include <fstream>

#include <map>

#include <set>

#ifndef STATS\_H

#define STATS\_H

struct Stats{

char cols; //Number of Cols in 2d Array

char card[52];

char suit[52];

list<char> s;

stack<int> vals;

map<char, int> m; //Map to translate card faces to integer values

set<char> uCard; //Set of Unique Cards in the deck};

#endif /\* STATS\_H \*/

# Tree.H Code

/\*

\* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt

\* to change this license Click

\* nbfs://nbhost/SystemFileSystem/Templates/cppFiles/file.h to edit this

\* template

\*/

/\*

\* File: Tree.h

\* Author: Daniel

\*

\* Created on May 27, 2023, 1:03 PM

\*/

#ifndef TREE\_H

#define TREE\_H

struct Node {

int val;

Node \*left;

Node \*right;

int height;

};

class Tree {

private:

int height(Node \*);

int balFact(Node \*);

Node \*rotL(Node \*);

Node \*rotR(Node \*);

Node \*findMax(Node \*);

Node \*balance(Node \*);

Node \*insert(Node \*, int);

void prntLev(Node \*, int);

public:

Node \*root;

void insert(int);

void inOrder(Node \*);

void prntLev(Node \*);

Node \*find(Node \*, int);

Node \*delNode(Node \*, int);

void preOrdr(Node \*);

void pstOrdr(Node \*);

Tree() { root = nullptr; }

~Tree();

void clean(Node \*);

void display(Node \*, int);

};

#endif /\* TREE\_H \*/

# Tree.cpp Code

/\*

\* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt

\* to change this license Click

\* nbfs://nbhost/SystemFileSystem/Templates/cppFiles/file.cc to edit this

\* template

\*/

#include "Tree.h"

#include <algorithm>

#include <iostream>

using namespace std;

void Tree::display(Node \*ptr, int level) {

int i;

if (ptr != nullptr) {

display(ptr->right, level + 1);

cout << endl;

if (ptr == root)

cout << "R -> ";

for (i = 0; i < level && ptr != root; i++)

cout << " ";

cout << ptr->val;

display(ptr->left, level + 1);

}

}

int Tree::height(Node \*node) {

if (node == nullptr)

return 0;

return node->height;

}

int Tree::balFact(Node \*node) {

if (node == nullptr)

return 0;

return height(node->left) - height(node->right);

}

Node \*Tree::rotL(Node \*node) {

Node \*r = node->right;

node->right = r->left;

r->left = node;

node->height = max(height(node->left), height(node->right)) + 1;

r->height = max(height(r->left), height(r->right)) + 1;

return r;

}

Node \*Tree::rotR(Node \*node) {

Node \*r = node->left;

node->left = r->right;

r->right = node;

node->height = max(height(node->left), height(node->right)) + 1;

r->height = max(height(r->left), height(r->right)) + 1;

return r;

}

Node \*Tree::balance(Node \*node) {

int bf = balFact(node);

// Right Heavy with all links in straight chain

if (bf < -1 && balFact(node->right) <= 0) {

return rotL(node);

}

// Right Heavy with not all links being in a straight chain

// 5 BF : -2 Need to Rotate Right Node Right First to Fix Non-Straight

// Chain

// \

// 7 BF : -1

// /

// 6 BF : 0

//

//

// After Right Rotation Looks like this

// 5 BF : -2 Need to Rotate Left Now

// \

// 6 BF : -1

// \

// 7 BF : 0

//

// After Left Rotation Looks like this

// 6 BF: 0

// / \

// 5 7 BF : 0 BF: 0

// Clearly Tree is Now Balanced as all balance factors fall within given

// tolerance BF -> [-1, 1] while still following rules of BST

// Operations Performed

// 1. Insert Node (already performed by insert function before calling

// balance)

// 2. Attempt to Return to Root node

// 3. Once a unbalanced node has been found check if the value just inserted

// was to the left or right of its left or right nodes

// 4. If value was inserted to the left of the nodes right node then we must

// rotate twice.

// 5. Rotate about the right node of the node we are currently on to get all

// nodes into straight chain

// 6. Rotate Left about our current node

if (bf < -1 && balFact(node->right) > 0) {

node->right = rotR(node->right);

return rotL(node);

}

// Left Heavy with not all links being in a straight chain (same steps as in

// right heavy with not all links in straight chain just reversed)

if (bf > 1 && balFact(node->left) < 0) {

node->left = rotL(node->left);

return rotR(node);

}

// Left Heavy with all links being straight chain

if (bf > 1 && balFact(node->left) >= 0) {

return rotR(node);

}

return node;

}

Node \*Tree::insert(Node \*node, int val) {

if (node == nullptr) {

Node \*r = new Node;

r->val = val;

r->left = nullptr;

r->right = nullptr;

r->height = 1;

return r;

}

if (val < node->val)

node->left = insert(node->left, val);

else

node->right = insert(node->right, val);

node->height = max(height(node->right), height(node->left)) + 1;

return balance(node);

}

void Tree::inOrder(Node \*node) {

if (node == nullptr)

return;

inOrder(node->left);

cout << (char)node->val;

inOrder(node->right);

}

void Tree::prntLev(Node \*node, int level) {

if (node == nullptr)

return;

if (level == 1)

cout << (char)node->val << " ";

else if (level > 1) {

prntLev(node->left, level - 1);

prntLev(node->right, level - 1);

}

}

void Tree::prntLev(Node \*node) {

int h = height(node);

for (int i = 1; i <= h; i++)

prntLev(node, i);

}

void Tree::preOrdr(Node \*node) {

if (node == nullptr)

return;

cout << node->val << " ";

preOrdr(node->left);

preOrdr(node->right);

}

void Tree::pstOrdr(Node \*node){

if (node == nullptr) return;

pstOrdr(node->left);

pstOrdr(node->right);

cout<<node->val<<" ";

}

void Tree::insert(int val) { root = insert(root, val); }

void Tree::clean(Node \*node) {

if (node != nullptr) {

clean(node->right);

clean(node->left);

delete node;

}

}

Node \*Tree::find(Node \*node, int val) {

if (node == nullptr)

return node;

else if (val == node->val)

return node;

else if (val > node->val) {

node = find(node->right, val);

} else if (val < node->val) {

node = find(node->left, val);

}

return node;

}

Node \*Tree::findMax(Node \*node) {

if (node == nullptr)

return node;

while (node->right != nullptr)

node = node->right;

return node;

}

Node \*Tree::delNode(Node \*node, int val) {

// Check to see if node entered is valid

if (node == nullptr)

return node;

if (val < node->val)

node->left = delNode(node->left, val);

else if (val > node->val)

node->right = delNode(node->right, val);

else {

if (node->left == nullptr ||node->right == nullptr) {

Node \*temp = node->left? node->left: node->right;

if (temp == nullptr) {

temp = node;

node = nullptr;

delete temp;

} else {

Node \*temp2 = node;

node = temp;

delete temp2;

}

} else {

Node \*temp = findMax(node->left);

node->val = temp->val;

node->left = delNode(node->left, temp->val);

}

}

if (node == nullptr)

return node;

node->height = max(height(node->left), height(node->right))+1;

balance(node);

return node;

}

Tree::~Tree() { clean(root); }

# Graph.h Code

/\*

\* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license

\* Click nbfs://nbhost/SystemFileSystem/Templates/cppFiles/file.h to edit this template

\*/

/\*

\* File: Graph.h

\* Author: Daniel

\*

\* Created on June 4, 2023, 11:18 AM

\*/

#include <map>

#include <utility>

#include <vector>

using namespace std;

#ifndef GRAPH\_H

#define GRAPH\_H

template<class T>

class Graph{

private:

map<T, pair<bool, vector<T>>> vertex;

void DFS(T);

public:

int nEdges;

Graph(){nEdges = 0;}

void insert(T, T);

void print(T);

};

template<class T>

void Graph<T>::DFS(T v){

vertex[v].first = true;

cout<<v<<" ";

vector<T> p = vertex[v].second;

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

if(vertex[p[i]].first == false)DFS(p[i]);

}

}

template<class T>

void Graph<T>::insert(T v1, T v2){

vertex[v1].second.push\_back(v2);

vertex[v2].second.push\_back(v1);

nEdges++;

}

template<class T>

void Graph<T>::print(T v){

DFS(v);

cout<<endl;

}

#endif /\* GRAPH\_H \*/

# Main.cpp Code

/\*

\*File: main.cpp

\*Author: Daniel Chvat

\*Created on June 4 2023, 12:23 PM

\*Purpose: Blackjack Card Game with STL concepts in CIS/CSC 17C as well as Recursion, Recursive Sorts

\*Trees, and Graphs

\*Project Length: 1656 Lines

\*/

//System Libraries

#include <iostream> //Input Output Library

#include <ctime> //Time Library

#include <string> //String Library

#include <cmath> //Math Library

#include <cstdlib> //C Standard Library

#include <iomanip> //Formatting Library

#include <fstream> //File Input Output Library

#include <vector> //Vector Library

#include <algorithm>

#include <set>

#include <map>

#include <list>

#include <queue>

#include <stack>

using namespace std;

//User Libraries

#include "Game.h"

#include "Card.h"

#include "MyVctr.h"

#include "Face.h"

#include "Player.h"

#include "File.h"

#include "Stats.h"

#include "Deck.h"

#include "Tree.h"

#include "Graph.h"

#include "LinkedList.h"

//Global Constants

//Mathematical/Physics/Conversions/Higher Dimensioned Arrays

const unsigned char PERCENT=100;

//Function Prototypes

void stats(File \*, const int, const char \*, const char \*);

void wrtBin(Stats, File \*);

int prtion(int[], int, int);

void qSort(int[], int, int);

void swap(int[], int, int);

Stats \*readBin(File \*);

bool hasWon(int, int, int limit=21);

string getSuit(char \*, int);

void destroy(Stats \*);

void srtVals(MyVctr<int> &, Stats \*);

void print(char);

int cHash(char);

bool hshSrch(char, LinkedList<char> \*\*, const int);

//Execution begins here

int main(){

//Declare Variables

int prvVal = 0;

const char NFILES = 3;

File \*\*file = new File\*[NFILES];

Player player;

Game game;

char again='\0';

float chng=0;

float bet=0;

char hit = '\0';

bool win = false;

char info = '\0';

char search = '\0';

const int NCHARS = 13;

LinkedList<char> \*\*hMap = new LinkedList<char> \*[NCHARS];

//Initialize Variables

//Create all 3 file classes

for(int i=0; i<NFILES; i++)file[i] = new File();

for(int i=0; i<NCHARS; i++)hMap[i] = new LinkedList<char>;

file[0]->setName("Cards.dat");

file[1]->setName("Suits.dat");

file[2]->setName("Stats.dat");

file[0]->getFile()->open(file[0]->getName(), ios::in|ios::out|ios::binary|ios::trunc);

file[1]->getFile()->open(file[1]->getName(), ios::in|ios::out|ios::binary|ios::trunc);

file[2]->getFile()->open(file[2]->getName(), ios::in|ios::out|ios::binary|ios::trunc);

Deck deck;

file[0]->getFile()->write(deck.getCrd(), sizeof(char)\*deck.getNCrd());

file[1]->getFile()->write(deck.getSut(), sizeof(char)\*deck.getNCrd());

//Map Inputs to Outputs

//Generate Random Deck of Cards with suits into two different files

file[0]->getFile()->close();

file[1]->getFile()->close();

file[2]->getFile()->close();

//Display Results

//Game Play

do{

game.setPVal(0);

game.setDVal(0);

//crdwSut="";

//Open the Cards.dat and Suits.dat files to be read and used later on in the game

file[0]->getFile()->open(file[0]->getName(), ios::in|ios::binary);

file[1]->getFile()->open(file[1]->getName(), ios::in|ios::binary);

cout<<endl;

cout<<"You Currently Have "<<fixed<<setprecision(2)<<player.getMny()<<" Dollars Left!"<<endl;

//Ensure player has more than 0 Dollars Left

if(player.getMny()>0){

cout<<"How Much Would You Like to Bet on This Game?"<<endl;

cin>>bet;

player.setBet(bet);

//Check if the bet is valid

while(player.getBet()>player.getMny() || player.getBet()<0){

cout<<"That Bet is Not Valid!"<<endl;

cout<<"Please Enter How Much You Wish to Bet:"<<endl;

cin>>bet;

player.setBet(bet);

player.mkBet();

}

//Grab a card from the cards file to start the player with

if(game.getIdx()<deck.getNCrd()){

//Fill cards and Suits Table to be printed later

//Display a message of the card and its suit that the player received

//Calculate the current value of all the players cards

game.setPVal(game.getCVal(deck.getVec(), prvVal));

cout<<endl;

cout<<"You Received: "<<deck.trnslte(true,game.getIdx())<<" of "<<deck.trnslte(false, game.getIdx())<<endl;

cout<<endl;

cout<<"Your Cards Total Value is Currently: "<<game.getPVal()<<endl<<endl;

try{

++game;

}catch(string error){

cout<<error;

exit(-1);

}

}else{

cout<<"No cards left in deck to draw from";

exit(0);

}

game.incDVal(rand()%11+1);

cout<<"The Dealers Shown Card Value is: "<<game.getDVal()<<endl<<endl;

cout<<"Would You Like to Hit (Y/N)"<<endl;

cin>>hit;

player.setHit(hit);

if(game.getIdx()<deck.getNCrd()){

do{

if(player.getHit()=='Y'||player.getHit()=='y'){

if(game.getIdx()<deck.getNCrd()){

//Calculate the current value of all the players cards

game.setPVal(game.getCVal(deck.getVec(), prvVal));

cout<<endl;

//Display Message to player saying the card they received their current total card value and the dealers show card value

cout<<"You Received: "<<deck.trnslte(true, game.getIdx())<<" of "<<deck.trnslte(false, game.getIdx())<<endl;

cout<<endl;

cout<<"Your Cards Total Value is Currently: "<<game.getPVal()<<endl<<endl;

cout<<"The Dealers Shown Card Value is: "<<game.getDVal()<<endl<<endl;

try{

++game;

}catch(string error){

cout<<error;

exit(-1);

}

}else{

//Tell the player there are no cards left in the deck and exit program

cout<<"No cards left in deck to draw from";

exit(0);

}

if(game.getPVal()<21){

cout<<"Would you like to hit (Y/N)"<<endl;

cin>>hit;

player.setHit(hit);

}else player.setHit('N');

}

}while(game.getIdx()<deck.getNCrd() && (player.getHit()=='Y'||player.getHit()=='y'));

//Increase dealers card value by a random value from 1 to 10 if their total card value is less than 17

while(game.getDVal()<17)game.incDVal(rand()%11+1);

}

//Check if the player has won

win=hasWon(game.getPVal(), game.getDVal());

player.setWin(win);

cout<<endl<<endl<<"Results:"<<endl;

if(player.getWin()){

//Pay out the bet at a 3:2 Ratio(found on blackjack.org) if the player has won

chng=player.getPCW();

//Display message saying player won

cout<<"Congratulations You Have Won :) "<<endl;

cout<<"Your Total Card Value Was: "<<game.getPVal()<<endl;

cout<<"The Dealer's Total Card Value Was: "<<game.getDVal()<<endl;

cout<<"Your New Balance is $"<<player.getMny()<<endl;

cout<<"Your money increased by around "<<fixed<<setprecision(0)<<round(chng)<<"%"<<endl;

}else if(game.getPVal()==game.getDVal() ||(game.getPVal()>21 && game.getDVal()>21)){

//Display a message saying nobody won

cout<<"Your Total Card Value Was: "<<game.getPVal()<<endl;

cout<<"The Dealer's Total Card Value Was: "<<game.getDVal()<<endl;

cout<<"Nobody Wins"<<endl;

}else{

chng=player.getPCL();

cout<<endl;

//Display message saying player lost

cout<<"Sorry You Lost :( "<<endl;

cout<<"Your Total Card Value Was: "<<game.getPVal()<<endl;

cout<<"The Dealer's Total Card Value Was: "<<game.getDVal()<<endl;

cout<<"Your New Balance is "<<fixed<<setprecision(2)<<player.getMny()<<endl;

//Display how much player's money decreased and round to the nearest Value

cout<<"Your money decreased by around "<<fixed<<setprecision(0)<<round(player.getPrc()-chng)<<"%"<<endl;

}

}

if(player.getMny()>0){

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

cin>>again;

player.setAgn(again);

}

prvVal+=game.getPVal();

}while(player.getMny()>0 && (player.getAgn()=='Y'||player.getAgn()=='y'));

//Write Game Stats into file

//Close Files Correctly

file[0]->getFile()->close();

file[1]->getFile()->close();

file[2]->getFile()->close();

stats(file[2], deck.getNCrd(), deck.getCrd(), deck.getSut());

Stats \*stats = readBin(file[2]);

//Ask Player if they want to see info about the game

cout<<"Would You Like To See Some Info About The Game?(Y/N)"<<endl;

cin>>info;

game.setInfo(info);

if(game.getInfo()=='Y' || game.getInfo()=='y'){

cout<<endl<<endl;

cout<<"Stats: "<<endl;

cout<<"-------"<<endl;

cout<<endl<<"Heres A List Of All The Cards That Were In The Deck With Their Suits"<<endl<<endl;

cout<<"Card: ";

int cVals[stats->cols];

for(int i=0; i<deck.getNCrd();i++){

cout<<stats->card[i];

cVals[i] = stats->m[stats->card[i]]; //Bidirectional Iterator

}

cout<<endl;

for(int i=0; i<deck.getNCrd(); i++)hMap[cHash(stats->card[i])%NCHARS]->addLst(stats->card[i]);

cout<<"Would You Like To Search the Deck To See If That Card is In the Deck? (Y/N)"<<endl;

cin>>info;

while(info == 'Y' || info == 'y'){

cout<<"Please Input The Card You Would Like To Search For"<<endl;

char search = '\0';

cin>>search;

if(hshSrch(search, hMap, NCHARS))cout << endl << search << " Is In The Deck" << endl;

else cout << endl << search << " Is Not In The Deck" << endl;

cout<<"Would You Like to Search For Another Card? (Y/N)"<<endl;

cin>>info;

}

cout<<"Sorted Card: ";

Tree tree;

for(int i=0; i<deck.getNCrd(); i++)tree.insert(static\_cast<int>(stats->card[i]));

tree.inOrder(tree.root);

cout<<endl;

//sort(stats->card, stats->card + stats->cols);

//for\_each(stats->card, stats->card+stats->cols, print);

for(int i=0; i<stats->cols; i++)stats->uCard.insert(stats->card[i]);

cout<<endl;

cout<<"Suit: ";

list<char>::iterator it; //Bidirectional Iterator

for(it = stats->s.begin(); it!=stats->s.end(); it++)cout<<\*it;

cout<<endl;

cout<<"Sorted Suit: ";

sort(stats->suit, stats->suit + stats->cols);

for\_each(stats->suit, stats->suit+stats->cols, print);

cout<<endl<<endl;

cout<<"Card Values: ";

for(int i=0; i<deck.getNCrd();i++)cout<<cVals[i]<<" ";

//Sort the Card Values

srtVals(deck.getVec(), stats);

cout<<endl;

cout<<"Unique Card Faces in Deck: ";

set<char>::iterator itr;

for(itr = stats->uCard.begin(); itr!=stats->uCard.end();itr++)cout<<\*itr<<" "; //Bidirectional Iterator

cout<<endl;

cout<<"Unique Card Suits in Deck: ";

Graph<char> g;

for(int i=1; i<deck.getNCrd(); i++){

g.insert(stats->suit[0], stats->suit[i]);

g.insert(stats->suit[i-1], stats->suit[i]); //Proof of Concept to show that this is a graph and not a tree

}

g.print(stats->suit[0]);

cout<<endl;

cout<<"Stats for Graph Used to Determine Unique Suits in Deck: "<<endl;

cout<<"--------------------------------------------------------"<<endl;

cout<<"Number of Vertices: "<<deck.getNCrd()<<endl;

cout<<"Number of Edges : "<<g.nEdges<<endl;

cout<<endl;

cout<<"Tree Used To Store Cards: "<<endl;

cout<<"------------------------------"<<endl<<endl;

tree.display(tree.root, 0);

}

cout<<endl<<endl<<"Thanks For Playing ";

player.printNm();

cout<<endl;

//De-Allocate Memory

destroy(stats);

for(int i=0; i<NCHARS; i++)delete hMap[i];

delete []hMap;

//Exit Stage Right

return 0;

}

bool hasWon(int pVal, int dVal, int limit){

if(pVal>dVal && pVal<=limit || (pVal<=21 && dVal>21))return true;

else return false;

}

//Get Initial Values for player structure

void stats(File \*f, const int NCARDS, const char \*a, const char \*b){

Stats s;

s.cols=NCARDS;

//Row 1 is Cards

//Row 2 is Suits

for(int i=0; i<s.cols; i++){

s.card[i]=a[i];

s.suit[i]=b[i];

}

wrtBin(s, f);

}

void wrtBin(Stats s, File \*f){

f->getFile()->open(f->getName(),ios::out|ios::binary);

f->getFile()->write(&s.cols, sizeof(char));

for(int i=0; i<s.cols; i++)f->getFile()->write(&s.card[i], sizeof(char));

for(int i=0; i<s.cols; i++)f->getFile()->write(&s.suit[i], sizeof(char));

f->getFile()->close();

}

Stats \*readBin(File \*f){

Stats \*s = new Stats;

long cursor=0L;

map<char, int>::iterator it = s->m.begin();

s->m.insert(pair<char, int>('A', 1));

s->m.insert(pair<char, int>('2', 2));

s->m.insert(pair<char, int>('3', 3));

s->m.insert(pair<char, int>('4', 4));

s->m.insert(pair<char, int>('5', 5));

s->m.insert(pair<char, int>('6', 6));

s->m.insert(pair<char, int>('7', 7));

s->m.insert(pair<char, int>('8', 8));

s->m.insert(pair<char, int>('9', 9));

s->m.insert(pair<char, int>('T', 10));

s->m.insert(pair<char, int>('Q', 10));

s->m.insert(pair<char, int>('K', 10));

s->m.insert(pair<char, int>('J', 10));

f->getFile()->open(f->getName(),ios::in|ios::binary);

f->getFile()->seekg(cursor, ios::beg);

f->getFile()->read(&s->cols, sizeof(char));

char \*c = new char[s->cols];

for(int i=0; i<s->cols; i++)f->getFile()->read(&c[i], sizeof(char));

copy(c, c+s->cols, s->card);

for(int i=0; i<s->cols; i++)f->getFile()->read(&s->suit[i], sizeof(char));

for(int i=0; i<s->cols; i++)s->s.push\_back(s->suit[i]);

return s;

f->getFile()->close();

}

void srtVals(MyVctr<int> &v, Stats \*s){

//Copy the vector into an int array for sorting

int temp[v.size()];

for(int i=0; i<v.size(); i++)temp[i]=v[i];

qSort(temp, 0, (sizeof(temp) / sizeof(temp[0]))-1);

cout<<endl;

cout<<"Sorted Card Values: ";

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

s->vals.push(temp[i]);

cout<<s->vals.top()<<" ";

}

cout<<endl;

cout<<endl;

}

//De-Allocate Memory

void destroy(Stats \*stats){

delete stats;

}

void print(char n){

cout<<n;

}

int prtion(int arr[], int low, int high) {

if (high == low)

return high;

else if (high == (low + 1)) {

if (arr[high] < arr[low])

swap(arr, low, high);

return high;

}

int mid = (high + low) / 2;

if (arr[mid] != arr[high])

swap(arr, mid, high);

int pivot = high;

int i = low;

int j = high - 1;

while (i < j) {

while (arr[i] < arr[pivot])

i++;

while (arr[i] >= arr[pivot] && arr[j] >= arr[pivot] && j > i)

j--;

if (arr[j] < arr[pivot] && j > i)

swap(arr, i, j);

}

if (arr[i] != arr[pivot])

swap(arr, i, pivot);

return i;

}

void qSort(int arr[], int low, int high) {

if (low < high) {

int p = prtion(arr, low, high);

qSort(arr, low, p - 1);

qSort(arr, p + 1, high);

}

}

void swap(int arr[], int a, int b) {

arr[a] = arr[a] ^ arr[b];

arr[b] = arr[b] ^ arr[a];

arr[a] = arr[a] ^ arr[b];

}

int cHash(char n){

int a = 0x352F8C; //Arbitrary Value I Randomly Generated

return (a + static\_cast<int>(n));

}

bool hshSrch(char find, LinkedList<char > \*\*hMap, const int SIZE){

int index = cHash(find)% SIZE;

if(hMap[index]->findLst(find) != -1) return true;

else return false;

}

# Sample Inputs / Outputs

A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated with medium confidence

A picture containing screenshot, text

Description automatically generated

# Versions

* V1: Copy of Project1 from CIS-17C
* V2: Added Tree for printing deck in order, added graph to find unique suits, added quicksort algorithm for sorting cards
* V3: Added Hashing Algorithm and hash map to allow user to search for and determine if a certain card was in the randomly generated deck

**Note**: This same version info can be found in the Versions.txt file included in the project folder and on the GitHub link posted in the introduction section