**Project 1**

Title

**Blackjack Card Game**

Course

**CIS-17C**

Section

**47065**

Due Date

**April 23, 2023**

Author

**Daniel Chvat**

**Table of Contents**

Table of Contents

[1 Introduction 3](#_Toc133172423)

[2 Project Statistics 3](#_Toc133172424)

[3. Containers Utilized 3](#_Toc133172425)

[4 Iterators 3](#_Toc133172426)

[5 Algorithms Utilized 4](#_Toc133172427)

[6 Major Classes 4](#_Toc133172428)

[7 Challenges Faced / Development 4](#_Toc133172429)

[8 Pseudocode 5](#_Toc133172430)

[9 UML Class Diagrams 7](#_Toc133172431)

[10 AbsPlayer.H Code 8](#_Toc133172432)

[11 Card.h Code 8](#_Toc133172433)

[12 Card.cpp Code 10](#_Toc133172434)

[13 Deck.h Code 14](#_Toc133172435)

[14 Deck.cpp Code 14](#_Toc133172436)

[15 Face.h Code 14](#_Toc133172437)

[16 File.h Code 15](#_Toc133172438)

[17 File.cpp Code 16](#_Toc133172439)

[18 Game.h Code 17](#_Toc133172440)

[19 Game.cpp Code 17](#_Toc133172441)

[20 MyVctr.h Code 18](#_Toc133172442)

[21 Player.h Code 21](#_Toc133172443)

[22 Player.cpp Code 22](#_Toc133172444)

[23 Stats.H Code 25](#_Toc133172445)

[24 Main.cpp Code 26](#_Toc133172446)

[25 Sample Inputs / Outputs 39](#_Toc133172447)

[26 Versions 40](#_Toc133172448)

# 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

|  |  |
| --- | --- |
|  |  |
| **Number of Lines including header file and Comments** | 1202 |
| **Number of Files** | 15 |
| **Number of Structures** | 1 |
| **Number of Functions in Main** | 8 |
| **Number of Classes** | 8 |

**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.

# Containers Utilized

|  |  |  |
| --- | --- | --- |
| Container Type | Container Utilized | Location in Code |
| Sequences | List | Stats.h Line 24 |
| Associative Containers | Map, Set | Stats.h Lines 26 and 27 respectively |
| Container Adapters | Stack, queue | Stats.h Line 26, Player.h Line 20 respecitvely |

# Iterators

|  |  |
| --- | --- |
| Iterator Type | Container |
| Trivial Iterator | Not utilized by any container in program |
| Input Iterator | Not utilized by any container in program |
| Output Iterator | Not utilized by any container in program |
| Forward Iterator | Not utilized by any container in program |
| Bidirectional Iterator | List, Map, set, |
| Random Access Iterator | Array, String, Vector |

# Algorithms Utilized

|  |  |  |
| --- | --- | --- |
| Algorithm Type | Algorithm Utilized | Location in Code |
| Non-mutating algorithms | for\_each | Main.cpp Line 225 |
| Mutating Algorithms | copy | Main.cpp Line 302 |
| Organization | Sort | Main.cpp Line 224 |

# 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 |

# Challenges Faced / Development

The main challenge I faced was just understanding the code that I had wrote in the previous classes as it had been a while since I had last read the code. Additionally, I also struggled with the individual iterators and getting them to work the way I intended.

# 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*

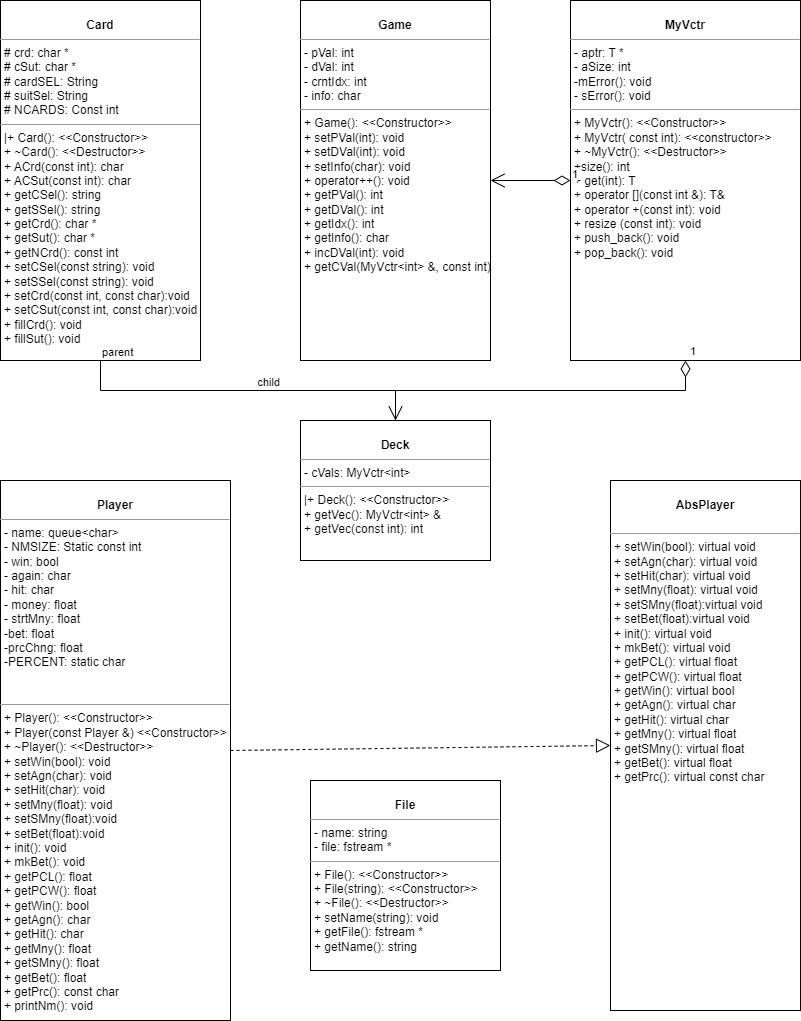
*Output the cards, suits, and sorted versions of cards and suits*

*Display the values of each card in the deck*

*Display sorted values of each card in the deck*

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

# UML Class Diagrams



# Main.cpp Flowchart

Flowchart can be seen in project folder under file name Project\_1\_PartialFlowChart.drawio.pdf

# 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 \*/

# Main.cpp Code

#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"

//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 \*);

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);

//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';

//Initialize Variables

//Create all 3 file classes

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

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<<"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;

cout<<"Sorted Card: ";

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

player.printNm();

cout<<endl;

//De-Allocate Memory

destroy(stats);

//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 = new int[52];

for(int i=0; i<52; i++)temp[i]=v[i];

sort(temp, temp + 52);

cout<<endl;

cout<<"Sorted Card Values: ";

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

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

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

}

cout<<endl;

cout<<endl;

delete temp;

}

//De-Allocate Memory

void destroy(Stats \*stats){

delete stats;

}

void print(char n){

cout<<n;

}#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"

//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 \*);

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);

//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';

//Initialize Variables

//Create all 3 file classes

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

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<<"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;

cout<<"Sorted Card: ";

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

player.printNm();

cout<<endl;

//De-Allocate Memory

destroy(stats);

//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 = new int[52];

for(int i=0; i<52; i++)temp[i]=v[i];

sort(temp, temp + 52);

cout<<endl;

cout<<"Sorted Card Values: ";

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

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

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

}

cout<<endl;

cout<<endl;

delete temp;

}

//De-Allocate Memory

void destroy(Stats \*stats){

delete stats;

}

void print(char n){

cout<<n;

}

# Sample Inputs / Outputs

Graphical user interface, text

Description automatically generated with medium confidence

Graphical user interface, text, application

Description automatically generated

# Versions

* V1: Copy of Final Project from CIS-17A
* V2: Added Map to translate from card face values to integer values, copy algorithm when reading in from files, set to contain unique card faces, for\_each loop, output of sorted card array and sorted suits array.
* V3: Added queue to hold players name and stack to print sorted card values.

**Note**: This same version info can be found in the Versions.txt file included in the project folder