**Project 2**

Title of Project

**Alex's New Blackjack Game**

Course

**CIS-17C 42475**

Date Due

**June 6th, 2018**

Created By

**Alex Castillo**

**Table of Contents**

[1 Introduction 3](#_gjdgxs)

[2 Game Play and Rules](#_30j0zll) 3

[3 Development Summary](#_1fob9te) 3

[4 Project Description and Specifications](#_2s8eyo1) 5

- [Sample Inputs/Outputs](#_17dp8vu) 5

- [Pseudocode](#_2jxsxqh) 7

- [Variables](#_z337ya) 8-11

5 Concepts used 11

6 [References](#_2xcytpi) 10

7 [Program Listing 1](#_1ci93xb)3-35

# **Introduction**

For this project I decided to challenge myself with making a card game based off of the game 21, or also known as Blackjack. I always enjoyed playing cards games with my family and friends and when I had this opportunity I wanted to do a card game and choosed Blackjack. The mental battle of a card game is always fun to play and I wanted to share that. It may not be as elaborate as poker, but it was still fun to create.

The alternation of this program was created to see how much I learned regarding classes, inheritance, the STL library, recursive sorts, hashing, trees, and more.

# **Game Play and Rules**

The program begins by displaying the introduction to the project which displays the choices of learning more about Blackjack, starting the game, or quiting the game.

After the introduction, the rules are then displayed to the player which state that Blackjack is played with one or more standard 52-card decks, with each card assigned a point value. The cards 2 through 10 are worth their face value. Kings, queens, and jacks are each worth 10, and aces may be used as either 1 or 11. The object for the player is to draw cards totaling closer to 21, without going over, than the dealer's cards. After the game, the player can then choose to play again and/or change their name for another player.

# **Project Development Summary**

|  |  |
| --- | --- |
| Total Lines of Code | **1482** |
| Lines of Code | **1113** |
| Pure Comment Lines | **150** |
| Pure Blank Lines | **219** |
| Total Lines of Source File | **94** |
| Number of Classes | **6** |
| Number of variables  (Minus repeating ones) | **56** |

## ***Project Aspects***

This C++ project is an application which is contained in eleven different files, my main file, my BlackjackGame.h with holds my classes and libraries, GeneralHashFunctions.h which holds general hashes for hashing, HashTable.h which makes a class to for the .cpp to hash a string, and Tree.h which creates the tree class so an AVL tree can be used. There are also the six .cpp files minus main: presentation, Card, and Deck, tree, hashtable, GeneralHashFunctions,. All these display usage of structures, strings, char arrays, static variables, classes, inheritance, polymorphism, the STL library with queues, algorithms, iterators, binary search, recursive sorts, hashing, trees, and sets. The programming was done using NetBeans 8.2 and outside help was originally needed for inserting an internal clock. The development of my second project took around a total work time of around 7 hours of brainstorming and actual coding.

## ***Project Difficulty***

The overall difficulty of this project was not too high and out of my level but I decided to go with Blackjack since I can set up the base and add other concepts after. The first main issue that I ran across while trying to create my project was creating the base for the game and following the rules for a game of Blackjack. Making sure the ace cards work as intended by always being an eleven first and act as a one if the total went over 21. However the first major issue was with the way my cards were being dealt. Originally I modeled my deck and hands to have a numeric value that would track how many cards have been dealt however when the dealer was given his hand he would constantly get the same card of the Ace of Spades. I had to change around the way the hands were dealt to fix it.

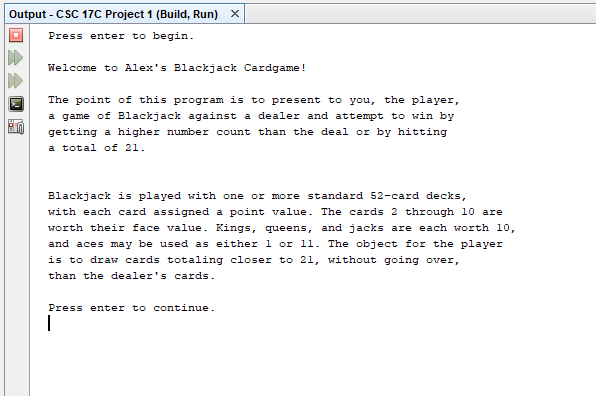
## ***Project Missed Opportunities***

Sadly being a straightforward game as it is, it was difficult to expand upon the game as it was. I had other ideas like making the game multiplayer with three other players or possibility implementing a betting system that could keep track of your cash over time which could definitely help with implementing other things I missed like graphs and maps. It was difficult to try to find a way I could incorporate a graph into the game without seeming unnecessary.

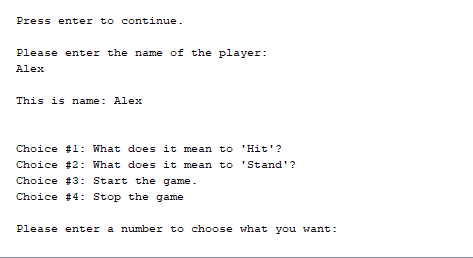
# **Project Description**

## ***Sample Inputs/Outputs***

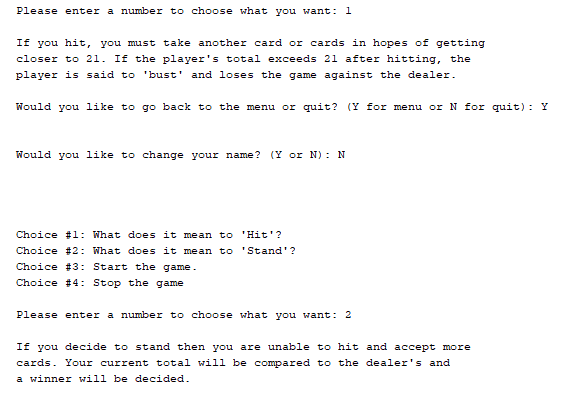
Since I created a game that uses random numbers for the cards, the inputs are more centered around being visually oriented and using one’s judgement skills as opposed to being data-centric with input and output. The beginning of the program asks the user to press enter to display the introduction and rules. After reading the user then presses enter again to enter their name.



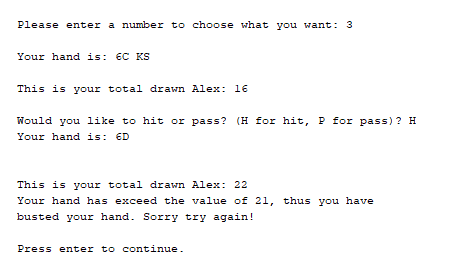
After enter a name the player is sent a prompt to choose a number 1-4, which will present the options to learned more about a hit and/or stand, start the game, or quit.



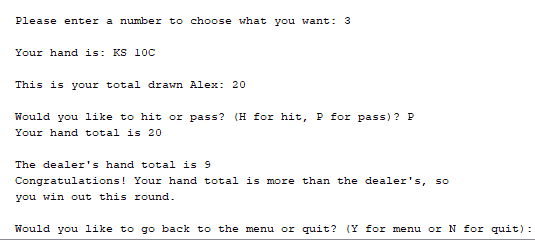
Choosing any of the first two options gives the player a detailed explanation of what a “hit” and “stand” is in Blackjack if some where not sure about it, like my brother when I asked him to test out the game.



When the game begins you get to see what your hand is with the first value indicating the face value you have, 1-10, J, Q, K, A and the second being the suit card. You are told what your total is and are given the option to hit or pass. Hitting grants another card but if the total goes over 21 than the player busts and loses.



If the player decides to pass then his total gets compared with the dealer’s and whoever’s total is the highest gets to win the game. (However I think if the player passes the dealer can hit however I wasn’t sure and can definitely implement it next time.)



## ***Pseudocode***

### **Version 1**

This pseudocode was made during the brainstorming aspect of the project and was a way to think about how I wanted the project to

**Display info:**

*Present the introduction, rules, and menu*

**Choose Option:**

*Send menu function*

*Switch function for player choice, call function for input*

*accept user input*

**Send card:**

*call function that player requested*

*pull randomly generated face and suit twice*

*display cards to user*

**Add points to the player’s score:**

*if (face card == number)*

*Ptotal += value of card;*

*Repeat once more*

**Add points to the dealer’s score**

*if (face card == number)*

*Ptotal += value of card;*

*Repeat once more*

**Decide to hit or pass**

*if(hit)*

*Add one more card to user*

*If (total > 21)*

*End game due to bust*

*Else*

*Ask to hit again, do while loop if hit is true, set hit to*

*Pass*

*if(pass)*

*Compare score from player and dealer*

**Stop the game:**

*if (Dealer > Player && Dealer <= 21)*

*Say the dealer won out*

*else if (Dealer = Pplayer)*

*No one wins*

*else if (Dealer < Player)*

*Player wins*

*else*

*Dealer busts out and player wins*

*Ask player if they want to go back to menu*

*if (player chooses yes)*

*display the subject menu*

*If (player chooses no*

*End the program*

## ***Variables***

main.cpp

string player; //Name of the player

char play = 'N'; //Character to hold player voice to replay

int inN; //Holds player choice

Intro start; //start variable for the intro class to present info

BlkJk game; //Variable to access the game

Card.cpp

int \_card

int n //Number of cards

auto f = lower\_bound(vect.begin(), vect.end(), "10") //Returns the first occurrence of 10

auto l = upper\_bound(vect.begin(), vect.end(), "J") //Returns the last occurrence of J

set <string, greater <string> > pcard1 //empty set container

set <string, greater <string> > :: iterator itr

string num

queue <string> cQue;

Deck.cpp

int randnum = rand() % 52 //Random numbers

BlackjackGame.h

Class Card

private:

int card // range 0 to 51

static const string CARD\_FACES[]; //Face values of cards

static const string CARD\_SUITS[]; //Suits of cards

class Deck

private:

Card cards[52]; //Array of cards in deck

int nxtCrdIndex; //Next card in index

class Intro

protected:

string name //Name of the player

class BlkJk : public Intro

public:

struct Num //Structure containing the elements for the first addition level

{

int num1; //First number to be created

int num2; //Second number to be created and added to num1

int num3; //Third number to be created and added if needed

int num4; //Fourth number to be created and added if needed

int num5; //Fifth number to be created and added if needed

int num6; //Sixth number to be created and added if needed

}

Presentation.cpp

int inN = 0 in Intro::getN() //Value to choose menu option

Variables in BlkJk::Num BlkJk::BlkGame(string name)

char HitPass //Choice to hit or pass

char hitAgain //Choice to hit again

int numOfCards = 2 // Input number for how many cards to deal.

int Ptotal = 0 //Total of player hand

int Dtotal = 0 //Total of dealer hand

Deck deck //Deck variable to shuffle and deal

Card car; //Card variable to test STL functions

Card c = deck.dealOneCard()

string suit = c.getSuit()

string face = c.getFace()

int hit = 1; //Access one card

HashTable.cpp

uint hash = RSHash(key);

uint index = hash % this->SIZE;

GeneralHashFunctions.cpp

unsigned int RSHash(const std::string& str)

{

unsigned int b = 378551;

unsigned int a = 63689;

unsigned int hash = 0;

}

unsigned int hash = 1315423911; in JSHash

PJWHash

unsigned int BitsInUnsignedInt = (unsigned int)(sizeof(unsigned int) \* 8);

unsigned int ThreeQuarters = (unsigned int)((BitsInUnsignedInt \* 3) / 4);

unsigned int OneEighth = (unsigned int)(BitsInUnsignedInt / 8);

unsigned int HighBits = (unsigned int)(0xFFFFFFFF) << (BitsInUnsignedInt - OneEighth);

unsigned int hash = 0;

unsigned int test = 0;

ELFHash

unsigned int hash = 0;

unsigned int x = 0;

BKDRHash

unsigned int seed = 131; // 31 131 1313 13131 131313 etc..

unsigned int hash = 0;

DJBHash

unsigned int hash = 5381;

DKEHash

unsigned int hash = static\_cast<unsigned int>(str.length());

FNVHash

const unsigned int fnv\_prime = 0x811C9DC5;

APHash

unsigned int hash = 0xAAAAAAAA;

## ***Concepts Used***

* ~~Arrays~~ (Lines 29,30 BlackjackGame.h)(Lines 11,13,35,46,51,72 Card.cpp) (Lines 16,29,41 Deck.cpp)
* ~~String arrays~~ (Lines 29,30 BlackjackGame.h) (Line 72 Vector string array Card.cpp)
* ~~String objects~~ (Line 72 Vector string array Card.cpp)
* ~~Structures~~  (Lines 76-84 BlackjackGame.h)s
* ~~Classes~~ (Lines 24,42,54,73 BlackjackGame.h, Lines 16-55 Tree.h, Lines 22-35 HashTable.h)
* ~~Constructors~~ (Lines 16-19 Card.cpp Lines 10-14 Tree.cpp)
* ~~Static Members~~ (Lines 29,30 BlackjackGame.h)
* ~~Copy Constructors~~ (Lines 23-26 Card.cpp)
* ~~Aggregation~~ (Lines 78-83,86 BlackjackGame.h)
* ~~Inheritance~~ (Line 73 BlackjackGame.h)
* ~~Protected Members~~ (Line 56-57 BlackjackGame.h)
* ~~Polymorphism~~ (Lines 118,124-126,173-175,233-235 Presentation.cpp)
* ~~Virtual Member Functions~~ (Line 68 BlackjackGame.h)
* ~~Error Handling~~ (Line 28 Deck.cpp)
* ~~STL Library~~ (49-82, 87.152, 157-163, 168-181 Card.cpp)(Lines 112-114 Presentation.cpp)
* ~~Recursive sort~~ (Lines 184-207 Card.cpp)
* ~~Hashing~~ (GeneralHashFunctions.cpp/.h Lines 63-67 main.cpp
* ~~Trees~~ (Trees.cpp/.h)

# **References**

**Google Searched Sites**

In points in this project where I needed guidance in what to do, when implementing some STL library, I looked through the book.

For usage of a queue I looked through page 640 of the STL book to find examples of a queue.

“#include <iostream>

#include <queue>

#include <string>

using namespace std;

int main()

{

queue<string> q;

// insert three elements into the queue

q.push("These ");

q.push("are ");

q.push("more than ");

// read and print two elements from the queue

cout << q.front();

q.pop();

cout << q.front();

q.pop();

// insert two new elements

q.push("four ");

q.push("words!");

// skip one element

q.pop();

// read and print two elements

cout << q.front();

12.3 Priority Queues 641

q.pop();

cout << q.front() << endl;

q.pop();

// print number of elements in the queue

cout << "number of elements in the queue: " << q.size()

<< endl;

}

Nicolai M. Josuttis-The C++ Standard Library\_ A Tutorial and Reference-Addison-Wesley (2012)

I also used the hash functions from earlier assignments, the .h and .cpp

typedef unsigned int (\*HashFunction)(const std::string&);

unsigned int RSHash (const std::string& str);

unsigned int JSHash (const std::string& str);

unsigned int PJWHash (const std::string& str);

unsigned int ELFHash (const std::string& str);

unsigned int BKDRHash(const std::string& str);

unsigned int SDBMHash(const std::string& str);

unsigned int DJBHash (const std::string& str);

unsigned int DEKHash (const std::string& str);

unsigned int BPHash (const std::string& str);

unsigned int FNVHash (const std::string& str);

unsigned int APHash (const std::string& str);

http://www.partow.net/programming/hashfunctions/index.html

# **Program Listing**

/\*

\* File: main.cpp

\* Author: Alex

\*

\* Created on June 3, 2018, 1:00 PM

\*/

#include "BlackjackGame.h"

#include "GeneralHashFunctions.h"

//Function Prototypes

// None

using namespace std;

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

{

//Variables for the player and game

string player; //Name of the player

string passwrd; //Player's password

string repeat; //Repeat password

string inputPass; //User input password

string msage; //Message for hash

string hash1; //Hash of message

string hash2; //Hash of password

string test2; //Hash of test password

int Hash; //Hash of hash1 and hash2

int Test; //Hash result of test

char play = 'N'; //Character to hold player voice to replay

int inN; //Holds player choice

Intro start; //start variable for the intro class to present info

BlkJk game; //Variable to access the game

//Initialize the random number seed

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

cout << "Press enter to begin.\n";

cin.get();

start.Introduction();

start.Rules();

cout << "Press enter to continue.\n";

cin.ignore(1,'\n'); //Pause screen

cout << "Please enter the name of the player: " << endl;

getline(cin,player); //Enter name

cout << "\nThis is name: " << player << endl;

start.setName(player); //Send name to class

cout << "Enter a Password: ";

cin >> passwrd;

cout << "Repeat your password: ";

cin >> repeat;

//If two input didn't match, ask for reenter

if (repeat != passwrd)

{

cout << "\n Error! Your repeat password different with your password!" << endl;

cout << " Press enter information again." << endl;

cin.ignore();

}

//Hash message

msage = "This program is awesome!";

hash1 = to\_string(RSHash(msage));

hash2 = to\_string(BPHash(passwrd));

Hash = ELFHash(hash1 + hash2);

do

{

player = start.difName(play, player);

cout << endl << endl;

start.Menu(); //Display menu

inN = start.getN(); //Accept user input for challenge choice

switch(inN) //Switch for deciding challenges or if misinput

{

case 1: start.Hit();break;

case 2: start.Stand();break;

case 3: game.BlkGame(player);break;

case 4: break;

default: start.getN();

}

cout << "Would you like to go back to the menu or quit? (Y for menu or N for quit): "; //Ask to play again

cin >> play; //User input for replay

cout << endl << endl;

}while (play == 'Y' || play == 'y');

//Exit stage right

return 0;

}

/\*

\* File: BlackjackGame.h

\* Author: Alex

\*

\* Created on June 3, 2018, 1:00 PM

\*/

#ifndef BLACKJACKGAME\_H

#define BLACKJACKGAME\_H

#include <iostream> //To accept user input

#include <cstdlib> //For randomized numbers

#include <ctime> //For randomized numbers

#include <string> //For user inputed name

#include <cassert> //Help with error handeling

#include <algorithm> //Use of binary search

#include <iterator> //For iterators

#include <vector> //Use of STL library

#include <set> //Use of showing off a set

#include <queue> //Use of showing off a queue

using namespace std;

class Card

{

private:

int card; // range 0 to 51

static const string CARD\_FACES[];

static const string CARD\_SUITS[];

public:

Card();

Card(int card);

void SearchCard();

void SetCard();

void QueueCard();

void ShowQue(queue <string>);

void recSort(string [], int);

void prtSort(string [], int);

string getSuit() const;

string getFace() const;

};

class Deck

{

private:

Card cards[52];

int nxtCrdIndex;

public:

Deck();

Card dealOneCard();

void shuffle();

};

class Intro

{

protected:

string name; //Name of the player

public:

Intro() //Default constructor

{name;}

Intro(string); //Constructor declaration

void Introduction(); //Introduces the program and its usage to the player

void Rules(); //Introduces the rules to the player

void Menu(); //Presents the menu selection to the user

void Hit(); //Presents meaning of a 'Hit' in Blackjack

void Stand(); //Presents meaning of 'Stand' in Blackjack

virtual void setName(string); //Set the player's name to the protected name string

string difName(char, string); //Asks if the user wants to change their name

int getN(); //Asks the user which challenge they want to partake in

};

class BlkJk : public Intro

{

public:

struct Num //Structure containing the elements for the first addition level

{

int num1; //First number to be created

int num2; //Second number to be created and added to num1

int num3; //Third number to be created and added if needed

int num4; //Fourth number to be created and added if needed

int num5; //Fifth number to be created and added if needed

int num6; //Sixth number to be created and added if needed

};

Num BlkGame (string);

};

#endif /\* BLACKJACKGAME\_H \*/

/\*

\* File: Card.cpp

\* Author: Alex

\*

\* Created on June 3, 2018, 1:00 PM

\*/

#include "BlackjackGame.h"

// Static Constants

const string Card::CARD\_FACES[] = {"A", "2", "3", "4", "5", "6", "7"

, "8", "9", "10", "J", "Q", "K"};

const string Card::CARD\_SUITS[] = {"S", "H", "C", "D"};

//Constructor

Card::Card()

{

card = 0; // Should initialize to a Joker.

}

// Constructor

Card::Card(int \_card)

{

card = \_card;

}

//getFace

//Returns face value of card.

//and a string representing card face

string Card::getFace() const

{

return CARD\_FACES[card%13];

}

//getSuit

//Returns suit of a card value.

// a string "S" (Spades), "H", (Hearts),

// "C" (Clubs), or "D" (Diamonds).

string Card::getSuit() const

{

return CARD\_SUITS[card/13];

}

void Card::SearchCard()

{

int n = sizeof(CARD\_FACES)/sizeof(CARD\_FACES[0]); //Number of cards

vector<string> vect(CARD\_FACES, CARD\_FACES+n);

// Sort the array to make sure that lower\_bound()

// and upper\_bound() work.

sort(vect.begin(), vect.end());

// Returns the first occurrence of 10

auto f = lower\_bound(vect.begin(), vect.end(), "10");

// Returns the last occurrence of J

auto l = upper\_bound(vect.begin(), vect.end(), "J");

cout << "The lower bound is at position: ";

cout << f-vect.begin() << endl;

cout << "The upper bound is at position: ";

cout << l-vect.begin() << endl;

//Show begin() return the beginning position of the container.

//and end() used to return the end position of the container.

vector<string> ar = {"A", "2", "3", "4", "5", "6", "7",

"8", "9", "10", "J", "Q", "K"};

// Declaring iterator to a vector

vector<string>::iterator ptr;

// Displaying vector elements using begin() and end()

cout << "The vector elements are : ";

for (ptr = ar.begin(); ptr < ar.end(); ptr++)

cout << \*ptr << " ";

}

void Card::SetCard()

{

// empty set container

set <string, greater <string> > pcard1;

// insert elements in random order

pcard1.insert("2");

pcard1.insert("7");

pcard1.insert("5");

pcard1.insert("1");

pcard1.insert("3");

pcard1.insert("2"); // only one 2 should be added to the set

pcard1.insert("4");

pcard1.insert("6");

pcard1.insert("10");

pcard1.insert("8");

pcard1.insert("9");

// printing set pcard1

set <string, greater <string> > :: iterator itr;

cout << "\nThe set pcard1 is : ";

for (itr = pcard1.begin(); itr != pcard1.end(); ++itr)

{

cout << '\t' << \*itr;

}

cout << endl;

// assigning the elements from pcard1 to pcard2

set <string> pcard2(pcard1.begin(), pcard1.end());

// print all elements of the set gquiz2

cout << "\nThe set pcard2 after assign from pcard1 is : ";

for (itr = pcard2.begin(); itr != pcard2.end(); ++itr)

{

cout << '\t' << \*itr;

}

cout << endl;

// remove all elements up to 30 in pcard2

cout << "\npcard2 after removal of elements less than '5' : ";

pcard2.erase(pcard2.begin(), pcard2.find("5"));

for (itr = pcard2.begin(); itr != pcard2.end(); ++itr)

{

cout << '\t' << \*itr;

}

// remove all elements with value 50 in pcard2

string num;

num = pcard2.erase ("8");

cout << "\npcard2.erase('8') : ";

cout << num << " removed \t" ;

for (itr = pcard2.begin(); itr != pcard2.end(); ++itr)

{

cout << '\t' << \*itr;

}

cout << endl;

//lower bound and upper bound for set pcard1

cout << "pcard1.lower\_bound('7') : "

<< \*pcard1.lower\_bound("7") << endl;

cout << "pcard1.upper\_bound('7') : "

<< \*pcard1.upper\_bound("7") << endl;

//lower bound and upper bound for set pcard2

cout << "pcard2.lower\_bound('7') : "

<< \*pcard2.lower\_bound("7") << endl;

cout << "pcard2.upper\_bound('7') : "

<< \*pcard2.upper\_bound("7") << endl;

}

void Card::ShowQue(queue <string> cardQ)

{

queue <string> c = cardQ;

while (!c.empty())

{

cout << '\t' << c.front();

c.pop();

}

cout << '\n';

}

void Card::QueueCard()

{

queue <string> cQue;

cQue.push("A");

cQue.push("B");

cout << "The queue cQue is : ";

ShowQue(cQue);

cout << "\ncQue.size() : " << cQue.size();

cout << "\ncQue.front() : " << cQue.front();

cout << "\ncQue.back() : " << cQue.back();

cout << "\ncQue.pop() : ";

cQue.pop();

ShowQue(cQue);

}

void Card::recSort(string ary[], int n)

{

// Base case

if (n == 1)

return;

// One pass of bubble sort. After

// this pass, the largest element

// is moved (or bubbled) to end.

for (int index=0; index < n-1; index++)

if (ary[index] > ary[index+1])

swap(ary[index], ary[index+1]);

// Largest element is fixed,

// recur for remaining array

recSort(ary, n-1);

}

//Function to print recursive sorted array

void Card::prtSort(string ary[], int n)

{

for (int index=0; index < n; index++)

cout << ary[index] << endl;

}

/\*

\* File: Deck.cpp

\* Author: Alex

\*

\* Created on June 3, 2018, 1:00 PM

\*/

#include "BlackjackGame.h"

//Constructor

Deck::Deck()

{

// Initialize the array to the nums 0-51

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

{

cards[i] = Card(i);

}

shuffle();

nxtCrdIndex = 0; // index of next available card

}

//dealOneCard

//Returns random Card.

Card Deck::dealOneCard()

{

assert(nxtCrdIndex >= 0 && nxtCrdIndex<52);

return cards[nxtCrdIndex++];

}

//shuffle

//Shuffles the Deck.

void Deck::shuffle()

{

// Shuffle by exchanging each element randomly

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

{

int randnum = rand() % 52;

swap(cards[i], cards[randnum]);

}

nxtCrdIndex = 0;

}

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* \*

\* General Purpose Hash Function Algorithms Library \*

\* \*

\* Author: Arash Partow - 2002 \*

\* URL: http://www.partow.net \*

\* URL: http://www.partow.net/programming/hashfunctions/index.html \*

\* \*

\* Copyright notice: \*

\* Free use of the General Purpose Hash Function Algorithms Library is \*

\* permitted under the guidelines and in accordance with the most current \*

\* version of the Common Public License. \*

\* http://www.opensource.org/licenses/cpl1.0.php \*

\* \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

#ifndef INCLUDE\_GENERALHASHFUNCTION\_CPP\_H

#define INCLUDE\_GENERALHASHFUNCTION\_CPP\_H

#include <string>

typedef unsigned int (\*HashFunction)(const std::string&);

unsigned int RSHash (const std::string& str);

unsigned int JSHash (const std::string& str);

unsigned int PJWHash (const std::string& str);

unsigned int ELFHash (const std::string& str);

unsigned int BKDRHash(const std::string& str);

unsigned int SDBMHash(const std::string& str);

unsigned int DJBHash (const std::string& str);

unsigned int DEKHash (const std::string& str);

unsigned int BPHash (const std::string& str);

unsigned int FNVHash (const std::string& str);

unsigned int APHash (const std::string& str);

#endif

#include "GeneralHashFunctions.h"

unsigned int RSHash(const std::string& str)

{

unsigned int b = 378551;

unsigned int a = 63689;

unsigned int hash = 0;

for(std::size\_t i = 0; i < str.length(); i++)

{

hash = hash \* a + str[i];

a = a \* b;

}

return hash;

}

/\* End Of RS Hash Function \*/

unsigned int JSHash(const std::string& str)

{

unsigned int hash = 1315423911;

for(std::size\_t i = 0; i < str.length(); i++)

{

hash ^= ((hash << 5) + str[i] + (hash >> 2));

}

return hash;

}

/\* End Of JS Hash Function \*/

unsigned int PJWHash(const std::string& str)

{

unsigned int BitsInUnsignedInt = (unsigned int)(sizeof(unsigned int) \* 8);

unsigned int ThreeQuarters = (unsigned int)((BitsInUnsignedInt \* 3) / 4);

unsigned int OneEighth = (unsigned int)(BitsInUnsignedInt / 8);

unsigned int HighBits = (unsigned int)(0xFFFFFFFF) << (BitsInUnsignedInt - OneEighth);

unsigned int hash = 0;

unsigned int test = 0;

for(std::size\_t i = 0; i < str.length(); i++)

{

hash = (hash << OneEighth) + str[i];

if((test = hash & HighBits) != 0)

{

hash = (( hash ^ (test >> ThreeQuarters)) & (~HighBits));

}

}

return hash;

}

/\* End Of P. J. Weinberger Hash Function \*/

unsigned int ELFHash(const std::string& str)

{

unsigned int hash = 0;

unsigned int x = 0;

for(std::size\_t i = 0; i < str.length(); i++)

{

hash = (hash << 4) + str[i];

if((x = hash & 0xF0000000L) != 0)

{

hash ^= (x >> 24);

}

hash &= ~x;

}

return hash;

}

/\* End Of ELF Hash Function \*/

unsigned int BKDRHash(const std::string& str)

{

unsigned int seed = 131; // 31 131 1313 13131 131313 etc..

unsigned int hash = 0;

for(std::size\_t i = 0; i < str.length(); i++)

{

hash = (hash \* seed) + str[i];

}

return hash;

}

/\* End Of BKDR Hash Function \*/

unsigned int SDBMHash(const std::string& str)

{

unsigned int hash = 0;

for(std::size\_t i = 0; i < str.length(); i++)

{

hash = str[i] + (hash << 6) + (hash << 16) - hash;

}

return hash;

}

/\* End Of SDBM Hash Function \*/

unsigned int DJBHash(const std::string& str)

{

unsigned int hash = 5381;

for(std::size\_t i = 0; i < str.length(); i++)

{

hash = ((hash << 5) + hash) + str[i];

}

return hash;

}

/\* End Of DJB Hash Function \*/

unsigned int DEKHash(const std::string& str)

{

unsigned int hash = static\_cast<unsigned int>(str.length());

for(std::size\_t i = 0; i < str.length(); i++)

{

hash = ((hash << 5) ^ (hash >> 27)) ^ str[i];

}

return hash;

}

/\* End Of DEK Hash Function \*/

unsigned int BPHash(const std::string& str)

{

unsigned int hash = 0;

for(std::size\_t i = 0; i < str.length(); i++)

{

hash = hash << 7 ^ str[i];

}

return hash;

}

/\* End Of BP Hash Function \*/

unsigned int FNVHash(const std::string& str)

{

const unsigned int fnv\_prime = 0x811C9DC5;

unsigned int hash = 0;

for(std::size\_t i = 0; i < str.length(); i++)

{

hash \*= fnv\_prime;

hash ^= str[i];

}

return hash;

}

/\* End Of FNV Hash Function \*/

unsigned int APHash(const std::string& str)

{

unsigned int hash = 0xAAAAAAAA;

for(std::size\_t i = 0; i < str.length(); i++)

{

hash ^= ((i & 1) == 0) ? ( (hash << 7) ^ str[i] \* (hash >> 3)) :

(~((hash << 11) + (str[i] ^ (hash >> 5))));

}

return hash;

}

/\* End Of AP Hash Function \*/

/\*

\* File: HashTable.h

\* Author: Alex

\*

\* Created on June 3, 2018, 1:00 PM

\*/

#ifndef HASHTABLE\_H

#define HASHTABLE\_H

#include <string>

#include <cstdlib>

#include <list>

#include "GeneralHashFunctions.h"

using namespace std;

typedef unsigned int uint;

class HashTable

{

private:

list<string> \*table;

int SIZE;

uint hash(string key);

public:

HashTable(int length);

virtual ~HashTable();

void put(string key, string val);

string at(string key);

bool hashMeet(string key);

bool exists(string key);

};

#endif /\* HASHTABLE\_H \*/

/\*

\* File: HashTable.cpp

\* Author: Alex

\*

\* Created on June 3, 2018, 1:00 PM

\*/

#include <string>

#include <cstdlib>

#include <list>

#include "HashTable.h"

using namespace std;

HashTable::HashTable(int length)

{

this->table = new list<string>[length];

this->SIZE = length;

}

HashTable::~HashTable( )

{

delete [] this->table;

}

uint HashTable::hash(string key)

{

uint hash = RSHash(key);

uint index = hash % this->SIZE;

return index;

}

void HashTable::put(string key, string val)

{

uint index = hash( key );

//If true then we have a collision

if(this->table[index].size() > 1)

{

this->table[index].push\_back(val);

}

else

{

//If no value here, no collision

this->table[index].push\_back(val);

}

}

string HashTable::at(string key)

{

uint index = hash(key);

if( this->table[index].size() > 0)

{

//If true return collision

if(this->table[index].size() == 1)

{

return this->table[index].front();

}

else

{

return "collision";

}

}

return "";

}

bool HashTable::hashMeet(string key)

{

uint index = hash( key );

if( this->table[index].size() > 1)

{

return true;

}

else

{

return false;

}

}

bool HashTable::exists(string key)

{

uint index = hash(key);

if( this->table[index].size() > 0)

{

return true;

}

else

{

return false;

}

}

/\*

\* File: Tree.h

\* Author: Alex

\*

\* Created on June 4, 2018, 8:00 AM

\*/

#ifndef TREE\_H

#define TREE\_H

#include <iostream>

using namespace std;

#include "BlackjackGame.h"

class Tree

{

private:

struct Node

{

int value;

struct Node \*left;

struct Node \*right;

int height;

};

Node \*node;

void insert(Node \*&, Node \*&);

void delet(Node \*&, int);

int difference(Node \*);

void clean(Node \*);

int getHeight(Node \*);

int max(int, int);

//Functions for printing

void preOrder(Node \*);

void inOrder(Node \*);

void postOrder(Node \*);

void prntNode(Node \*);

void rotateRR(Node \*&);

void rotateLL(Node \*&);

void rotateRL(Node \*&);

void rotateLR(Node \*&);

public:

Tree();

~Tree();

void insert(int);

void delet(int);

void prePrnt();

void inPrnt();

void postPrnt();

};

#endif /\* TREE\_H \*/

/\*

\* File: Tree.cpp

\* Author: Alex

\*

\* Created on June 4, 2018, 8:00 AM

\*/

#include "Tree.h"

//Constructor

Tree::Tree()

{

node = NULL;

}

//Destructor

Tree::~Tree()

{

clean(node);

}

void Tree::clean(Node \*node)

{

if (node != NULL)

{

clean(node->left);

clean(node->right);

delete node;

}

}

int Tree::getHeight(Node\* n)

{

if (n == NULL) return 0;

return n->height;

}

void Tree::insert(int key)

{

//Create a new node for insertion

Node \*newNode = new Node;

newNode->value = key;

newNode->left = NULL;

newNode->right = NULL;

newNode->height = 1;

//Insert with recursion

insert(node, newNode);

}

//Private insertion

void Tree::insert(Node \*&node, Node \*&newNode)

{

if (node == NULL)

{

node = newNode;

} else if (newNode->value <= node->value)

{

insert(node->left, newNode);

} else

{

insert(node->right, newNode);

}

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

//Right case

if (difference(node)<-1)

{

if (newNode->value >= node->right->value)

{

rotateRR(node);

} else

{

rotateRL(node);

}

}

if (difference(node) > 1)

{

if (newNode->value <= node->left->value)

{

rotateLL(node);

} else

{

rotateLR(node);

}

}

}

void Tree::prePrnt()

{

preOrder(node);

}

void Tree::inPrnt()

{

inOrder(node);

}

void Tree::postPrnt()

{

postOrder(node);

}

int Tree::difference(Node \*node)

{

if (node == NULL) return 0;

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

}

void Tree::delet(int n)

{

delet(node, n);

}

void Tree::rotateRR(Node \*&parent)

{

Node \*node = parent->right;

parent->right = node->left;

node->left = parent;

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

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

parent = node;

}

void Tree::rotateLL(Node \*&parent)

{

Node \*node = parent->left;

parent->left = node->right;

node->right = parent;

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

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

parent = node;

}

void Tree::rotateRL(Node \*&z)

{

rotateLL(z->right);

rotateRR(z);

}

void Tree::rotateLR(Node \*&z)

{

rotateRR(z->left);

rotateLL(z);

}

void Tree::delet(Node \*&node, int key)

{

if (node == NULL) return;

if (key < node->value)

{

delet(node->left, key);

}

else if (key > node->value)

{

delet(node->right, key);

}

else

{

//If less than one child

if ((node->left == NULL) || (node->right == NULL))

{

//If 0 child

if ((node->left == NULL) && (node->right == NULL))

{

Node \*temp = node;

node = NULL;

delete temp;

}

//If 1 child

else

{

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

Node \*temp2 = node;

node = temp;

delete temp2;

}

}

//If 2 children

else

{

Node \*temp = node->left;

if (temp != NULL)

{

while (temp->right != NULL)

{

temp = temp->right;

}

}

node->value = temp->value;

delet(node->left, node->value);

}

}

if (node == NULL) return;

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

//Balance tree

if (difference(node)<-1)

{

if (difference(node->right) < 0)

{

rotateRR(node);

}

else

{

rotateRL(node);

}

}

if (difference(node) > 1)

{

if (difference(node->left) > 0)

{

rotateLL(node);

}

else

{

rotateLR(node);

}

}

}

void Tree::prntNode(Node \*node)

{

cout << node->value << " ";

}

void Tree::preOrder(Node \*node)

{

if (node != NULL)

{

prntNode(node);

preOrder(node->left);

preOrder(node->right);

}

}

void Tree::inOrder(Node \*node)

{

if (node != NULL)

{

inOrder(node->left);

prntNode(node);

inOrder(node->right);

}

}

void Tree::postOrder(Node \*node)

{

if (node != NULL)

{

postOrder(node->left);

postOrder(node->right);

prntNode(node);

}

}

int Tree::max(int a, int b)

{

return a >= b ? a : b;

}

/\*

\* File: Presentation.cpp

\* Author: Alex

\*

\* Created on June 3, 2018, 1:00 PM

\*/

#include "BlackjackGame.h"

#include "Tree.h"

//Constructor that sets the name to the protected variable

Intro::Intro(string player)

{

player = name;

}

//Sets the user inputed name to the protected name variable

void Intro::setName(string player)

{

name = player;

}

//Presents introduction for when the player boots up program

void Intro::Introduction()

{

cout << "Welcome to Alex's Blackjack Cardgame!\n" << endl;

cout << "The point of this program is to present to you, the player,\n";

cout << "a game of Blackjack against a dealer and attempt to win by\n";

cout << "getting a higher number count than the deal or by hitting\n";

cout << "a total of 21.\n\n";

}

//Presents rules for a game of blackjack

void Intro::Rules()

{

cout << "\nBlackjack is played with one or more standard 52-card decks,\n";

cout << "with each card assigned a point value. The cards 2 through 10 are\n";

cout << "worth their face value. Kings, queens, and jacks are each worth 10,\n";

cout << "and aces may be used as either 1 or 11. The object for the player\n";

cout << "is to draw cards totaling closer to 21, without going over, \n";

cout << "than the dealer's cards.\n\n";

}

//Presents the menu for the player to look more into a hit or stand move

void Intro::Menu()

{

cout << "Choice #1: What does it mean to 'Hit'?\n";

cout << "Choice #2: What does it mean to 'Stand'?\n";

cout << "Choice #3: Start the game.\n";

cout << "Choice #4: Stop the game\n\n";

}

//Presents meaning of a 'Hit' in Blackjack

void Intro::Hit()

{

cout << "If you hit, you must take another card or cards in hopes of getting\n";

cout << "closer to 21. If the player's total exceeds 21 after hitting, the\n";

cout << "player is said to 'bust' and loses the game against the dealer.\n\n";

}

//Presents meaning of 'Stand' in Blackjack

void Intro::Stand()

{

cout << "If you decide to stand then you are unable to hit and accept more\n";

cout << "cards. Your current total will be compared to the dealer's and\n";

cout << "a winner will be decided.\n\n";

}

//Function if the user wants to change their name after finishing a challenge

string Intro::difName(char change, string player)

{

if (change == 'Y' || change == 'y') //Only activates on replay

{

cout << "Would you like to change your name? (Y or N): ";

cin >> change; //Character to hold player choice of name change

cout << endl << endl;

if (change == 'Y' || change == 'y') //If name wants to be changed

{

cin.ignore(1000,'\n'); //Prevent skipping of name on retry

cout << "Please enter the name of the player: ";

getline(cin,player); //Enter new name

Intro::setName(player);

cout << endl << endl;

}

}

return player;

}

//Get the user variable for the player choice

int Intro::getN()

{

int inN = 0;

cout << "Please enter a number to choose what you want: ";

cin >> inN;

cout << endl;

return inN;

}

BlkJk::Num BlkJk::BlkGame(string name)

{

char HitPass; //Choice to hit or pass

char hitAgain; //Choice to hit again

int numOfCards = 2; // Input number for how many cards to deal.

int Ptotal = 0; //Total of player hand

int Dtotal = 0; //Total of dealer hand

srand(time(0)); // Initializes random "seed".

Deck deck; //Deck variable to shuffle and deal

Card card; //Card variable to test STL functions

Tree tree; //AVL tree to display

// card.SearchCard(); //Binary search algorithm test

// card.SetCard(); //Set test for the program

// card.QueueCard() //Queue test for program

if (numOfCards == 2)

{

deck.shuffle();

//Player hand

cout << "Your hand is: ";

for (int cardNum = 0; cardNum < numOfCards; cardNum++)

{

Card c = deck.dealOneCard();

string suit = c.getSuit();

string face = c.getFace();

cout << face << suit << " ";

if (c.getFace() == "1")

Ptotal+=1;

else if (c.getFace() == "2")

Ptotal+=2;

else if (c.getFace() == "3")

Ptotal+=3;

else if (c.getFace() == "4")

Ptotal+=4;

else if (c.getFace() == "5")

Ptotal+=5;

else if (c.getFace() == "6")

Ptotal+=6;

else if (c.getFace() == "7")

Ptotal+=7;

else if (c.getFace() == "8")

Ptotal+=8;

else if (c.getFace() == "9")

Ptotal+=9;

else if (c.getFace() == "10")

Ptotal+=10;

else if (c.getFace() == "J")

Ptotal+=10;

else if (c.getFace() == "Q")

Ptotal+=10;

else if (c.getFace() == "K")

Ptotal+=10;

else if (c.getFace() == "A")

{

Ptotal+=11;

if (Ptotal > 21)

{

Ptotal - 10;

}

}

else

Ptotal+=0;

}

cout << endl;

}

//Dealer hand

if (numOfCards == 2)

{

for (int cardNum = 0; cardNum < numOfCards; cardNum++)

{

Card c = deck.dealOneCard();

string suit = c.getSuit();

string face = c.getFace();

if (c.getFace() == "1")

Dtotal+=1;

else if (c.getFace() == "2")

Dtotal+=2;

else if (c.getFace() == "3")

Dtotal+=3;

else if (c.getFace() == "4")

Dtotal+=4;

else if (c.getFace() == "5")

Dtotal+=5;

else if (c.getFace() == "6")

Dtotal+=6;

else if (c.getFace() == "7")

Dtotal+=7;

else if (c.getFace() == "8")

Dtotal+=8;

else if (c.getFace() == "9")

Dtotal+=9;

else if (c.getFace() == "10")

Dtotal+=10;

else if (c.getFace() == "J")

Dtotal+=10;

else if (c.getFace() == "Q")

Dtotal+=10;

else if (c.getFace() == "K")

Dtotal+=10;

else if (c.getFace() == "A")

{

Dtotal+=11;

if (Dtotal > 21)

{

Dtotal - 10;

}

}

else

Dtotal+=0;

}

cout << endl;

}

cout << "This is your total drawn " << name << ": "<< Ptotal << endl;

cout << "\nWould you like to hit or pass? (H for hit, P for pass)? ";

cin >> HitPass;

if (HitPass == 'H' || HitPass == 'h')

{

do

{

int hit = 1; //Access one card

if (hit == 1)

{

//Player hand

cout << "Your hand is: ";

for (int cardNum = 0; cardNum < hit; cardNum++)

{

Card c = deck.dealOneCard();

string suit = c.getSuit();

string face = c.getFace();

cout << face << suit << " ";

int Ptotal2 = 0;

if (c.getFace() == "1")

{

Ptotal2+=1;

Ptotal = Ptotal + Ptotal2;

}

else if (c.getFace() == "2")

{

Ptotal2+=2;

Ptotal = Ptotal + Ptotal2;

}

else if (c.getFace() == "3")

{

Ptotal2+=3;

Ptotal = Ptotal + Ptotal2;

}

else if (c.getFace() == "4")

{

Ptotal2+=4;

Ptotal = Ptotal + Ptotal2;

}

else if (c.getFace() == "5")

{

Ptotal2+=5;

Ptotal = Ptotal + Ptotal2;

}

else if (c.getFace() == "6")

{

Ptotal2+=6;

Ptotal = Ptotal + Ptotal2;

}

else if (c.getFace() == "7")

{

Ptotal2+=7;

Ptotal = Ptotal + Ptotal2;

}

else if (c.getFace() == "8")

{

Ptotal2+=8;

Ptotal = Ptotal + Ptotal2;

}

else if (c.getFace() == "9")

{

Ptotal2+=9;

Ptotal = Ptotal + Ptotal2;

}

else if (c.getFace() == "10")

{

Ptotal2+=10;

Ptotal = Ptotal + Ptotal2;

}

else if (c.getFace() == "J")

{

Ptotal2+=10;

Ptotal = Ptotal + Ptotal2;

}

else if (c.getFace() == "Q")

{

Ptotal2+=10;

Ptotal = Ptotal + Ptotal2;

}

else if (c.getFace() == "K")

{

Ptotal2+=10;

Ptotal = Ptotal + Ptotal2;

}

else if (c.getFace() == "A")

{

Ptotal2+=11;

if ((Ptotal + Ptotal2) > 21)

{

Ptotal += Ptotal2;

Ptotal = Ptotal - 10;

}

}

else

Ptotal2+=0;

}

cout << endl;

}

cout << endl << endl;

cout << "This is your total drawn " << name << ": "<< Ptotal << endl;

if (Ptotal > 21)

{

cout << "Your hand has exceed the value of 21, thus you have\n";

cout << "busted your hand. Sorry try again!\n\n";

cout << "Press enter to continue.\n";

cin.get();

cin.ignore(1);

HitPass = '1';

hitAgain = 'N';

}

else

{

cout << "Would you like to hit again? (H for Hit or P for Pass)? ";

cin >> hitAgain;

if (hitAgain == 'P' || hitAgain == 'p')

{

HitPass = 'P';

}

}

}while(hitAgain == 'H' || hitAgain == 'h');

}

if (HitPass == 'P' || HitPass == 'p')

{

cout << "Your hand total is " << Ptotal << endl;

if (Dtotal > Ptotal && Dtotal <= 21)

{

cout << "\nThe dealer's hand total is " << Dtotal << endl;

cout << "Sadly your hand total is less than the dealer's, so\n";

cout << "you lose out this round.\n\n";

}

else if (Dtotal == Ptotal)

{

cout << "\nThe dealer's hand total is " << Dtotal << endl;

cout << "Sadly your hand total is equal to the dealer's, so\n";

cout << "nobody wins this round.\n\n";

}

else if (Dtotal < Ptotal)

{

cout << "\nThe dealer's hand total is " << Dtotal << endl;

cout << "Congratulations! Your hand total is more than the dealer's, so\n";

cout << "you win out this round.\n\n";

}

else

{

cout << "\nThe dealer's hand total is " << Dtotal << endl;

cout << "Congratulations! Since the dealer busted out his hand,\n";

cout << "you win out this round.\n\n";

}

}

}