

Project 2

Black-Jack

Danny Obeid

CIS-17C

6/7/2020

INTRODUCTION:

Title: Black-Jack

Rules:

The rules of Black-Jack are simple, it is you versus the dealer. The goal is beat the dealer's hand without going over 21. Face cards are worth 10 and Aces are worth 11 or 1, depending on your choice. Each player starts with two cards and can hit to add another until you're either satisfied with your hand, or you go over. If you go over 21, you lose. If the dealer goes over 21 you win! If you both get 21 it's a standoff. Good luck!

SUMMARY:

Lines: 600

Programming this new version of Black-Jack which had to be based on my previous version came with a lot of difficulty. If the understanding was that during the first project, that the next project would involve adding more data structures to it, I would have in a different direction than Black-Jack. The issue is Black-Jack has a limited scope, so while it did meet the requirements of the first project, some of the new data structures in this project, such as hashing, graphs and trees, had no natural place in Black-Jack and therefore couldn't be incorporated. It wasn't due to lack of trying, I sought out help and made multiple attempts to force it to work however it didn't. The process took a total of about 12 days for about 4-5 hours a day. It took a lot of ingenuity to make the data structures that could be incorporated to work. It was challenging however whatever was possible to be used, I did. This project is located on my public GitHub profile in a public Repo called Dobeid17/Project2-BlackJack.

PSEUDO CODE:

Create struct and initialize data

Bring up main menu with options

Switch choice

Case 1: hear rules is selected display rules then ask play or quit

Case 2: call function playGame()

Case 3: quit

playGame()

```

While balance >= 5 && play == true
    Set balance and take in bet
    While bet < 5 && bet > balance
        Invalid bet, try again
    Call loadRecursive to use recursive deck
    Call shuDeck() and assign to deck
    Draw cards to hand
    If you have an ace, ask to change it to an 11
    Show one card of dealers hand
    Ask to hit or stay
    While toHit == 1 && toPlay != 1
        Call hitCard() to add card
        Call isOver() check to see if it goes over 21
            If over 21 you lose bet ask to play again or quit
                If toPlay == 2 quit program
    If toPlay != 1
        display dealer hand
        Call compareHands() to check if dealer won before hitting
        If dealWon == true
            You lose ask to play again
            If toPlay == 2 quit program
        While isDealOver() == false && dealWon == false
            call hitCard()
        Call isDealOver() check if dealer hand goes over 21
        If dealer goes over 21 you win
            Update balance ask to play again or quit
        If dealBust == false
            If compareHands() == 1

```

```

        You win update balance
    Else if compareHands) == -1
        You lose display balance
    Else
        It's a draw return balance
    Ask to play again or quit
    If toPlay == 2
        return
If balance < 5
    Insufficient funds quit program

```

UML:

Card
String cardType Int cardValue + Card() + Card() String , Int

FLOWCHART:

- FUNCTIONS:**

that is where it was set. Once it is verified that the player has enough the play, this function gets called.

Card hitCard(stack<Card>& deck);

- Whenever the player or the dealer adds a new card to their card, this function is called. If the player wants to hit after their initial pair cards this function is called. Also called if the dealer is under 17 and will continue to be called until it is over 17, or busts.

bool isOver(list<Card> hand);

- Function that checks if either the player or dealer is over the set target of 21. If either are over, they automatically lose. Called after every hit, for the player or dealer.

bool isDealOver(list<Card> dealHand);

- This function checks if the dealer is over the target limit of 17 to decide whether or not the dealer needs to hit again. As long as the dealer is under 17, hitCard() will continue to run.

bool is21(list<Card> hand);

- Function to check if the dealer reaches the target amount of 21 exactly. This is called after the final hit by the dealer.

bool ifAce(list<Card> hand);

- Once the first pair of cards are distributed, this function is called to see if either the player or user is carrying an ace. If an ace is identified it returns true and then runs replaceAce().

void replaceAce(list<Card> &hand)

- This function is called if ifAce() returns true. If true it swaps the 1 the player or dealer is carrying with an 11.

int compareHands(list<Card> myHand, list<Card> dealHand); wins if neither are 21

- If neither the player nor the dealer get exactly 21, this function Is called to compare their hands and see who's is higher. It is called after the dealer reveals their full hand to check if the dealer even needs to hit. After that, it called after it checks out that no one got 21, and no one busted. Decides who has the greater hand.

void loadRecursive(stack<Card> &reDeck, int valCount)

- This is the function that gets us our new deck that is recursively established. This deck in then passed into the shuDeck() function to be shuffled and prepared for play.

CHECK OFF LIST:

1. Container Class

1. Sequences

- List: `list<Card> myHand` , and `list<Card> dealHand` were used to store the cards in both the players hand, and the dealers hand. These were referenced and iterated through multiple times throughout the code.

2. Associative Containers

- Map: `unordered_map<int, Card> = newDeck`. An unordered map was used in the creation of the deck of cards to be used shuffled and distributed. It took in data of type `int`, and the struct object `Card` I created above.

3. Container Adapters:

- Stack: `stack<Card> doneDeck`. This stack contained the shuffled deck we got from the `unordered_map` above. This was a crucial piece of the program because without a deck, the game cannot be played.

2. Iterators

1. Forward Iterator:

- `list<Card>::iterator it = hand.begin()`. The forward iterator was used many times in the code since it was what I used to advance through the list to access the data in it in order. Using `advance(it, 1)` inside of a loop I was able to go through the list one element at a time and sum its values to determine the overall hands value.

2. Random Access Iterator:

- `unordered_map<int, Card>::iterator random_it = newDeck.begin()`. This was my random access iterator and it was used in the shuffling of the deck.

3. Algorithms

1. Mutating Algorithm

- `Emplace` – Used in the `shuDeck()` function to put the new card on top of the unordered map.

4. Recursion

- `loadRecursive()`

-Used to create a deck of cards by recursively creating cards and storing

them into a stack. By creating them this way, no need for a recursive sort to be used.

5. Recursive Sort

- There was nowhere to naturally fit a recursive sort into this game, rather at the bottom of the code I left an example demonstrating the use of Quick Sort to sort an array of numbers.

REFERENCES:

Stackoverflow.com

Textbook

Gaddis Getting started with C++

Sample Output:

Hello and welcome to Black Jack!

What would you like to do?

1.) Hear rules.

2.) Play.

3.) Quit.

1

The rules of Black Jack are simple, it is you versus the dealer. The goal is beat the dealers hand without going over 21. Face cards are worth 10 and Aces are worth 11 or 1. Each player starts with two cards and can hit to add another. If you go over 21, you lose. If the dealer goes over 21 you win! If it is a standoff your bet will get returned. Good luck!

2.) Play.

3.) Quit.

2

Here is your current balance: \$100

How much do you want to bet this hand?

Must be more than \$5 and less than \$ 100

15

Here is your hand: 10 Clovers 9 Clovers

Here is what the dealer is showing: 5 Clovers

What are you going to do?

1.) Hit

2.) Stay

2

Here is the dealers hand.

2 Clovers 5 Clovers

The dealer hits.

2 Clovers 5 Clovers 6 Spades

The dealer hits.

2 Clovers 5 Clovers 6 Spades 6 Hearts

Its a draw! Your bet will be returned

Here is your balance: \$100

What do you want to do?

1.) Play again.

2.) Quit

1

How much do you want to bet this hand?

Must be more than \$5 and less than \$ 100

15

Here is your hand: 1 Diamonds 10 Spades

You pulled an Ace. Would you like to turn your Ace into an 11? (y/n)

y

Here is your new hand:

11 Diamonds 10 Spades

Here is what the dealer is showing: 10 Hearts

What are you going to do?

1.) Hit

2.) Stay

2

Here is the dealers hand.

3 Diamonds 10 Hearts

The dealer hits.

3 Diamonds 10 Hearts 1 Diamonds

The dealer hits.

3 Diamonds 10 Hearts 1 Diamonds 5 Hearts

YOU WIN!

Here is your new balance: \$115

What do you want to do?

1.) Play again.

2.) Quit

1

How much do you want to bet this hand?

Must be more than \$5 and less than \$ 115

115

Here is your hand: 10 Hearts 8 Spades

Here is what the dealer is showing: 1 Diamonds

What are you going to do?

1.) Hit

2.) Stay

2

Here is the dealers hand.

10 Diamonds 11 Diamonds

You Lose!

Here is your new balance: \$ 0

What do you want to do?

1.) Play again.

2.) Quit

1

INSUFFICIENT FUNDS. You are unable to continue.