Celeste Hernandez Mora

Professor Lehr

19 November 2023

CIS-17C 48784

Project Write-up: Blackjack

Introduction

I am coding Blackjack. I chose this game because I wanted to elaborate more on an incomplete Blackjack project of mine from a past term.

Approach to Development

* My old code was centered around class, arrays, and vectors, so I ended up rewriting it entirely. The only concept I kept from the previous was my use of structs, to store card data unique to each player and to each deck. Instead of arrays and vectors, I made much use of lists, and for decks, I used stack to make sure unique cards are pulled from a 52-card deck. Pairs were also very much utilized, as they held the basic identification of a unique card. As per lists, bidirectional iterators were employed to search through them and make certain calculations. Swaps were also employed to shuffle the deck randomly.

Game Rules

Blackjack is centered around number 21.

First, you draw two cards, and face the dealer, who has one visible card and one hidden card. You are to first make a bet, and then after you get your two cards, you either pull a new card or stand.

The goal is to hit 21 or to surpass your opponent's sum without going over 21. If it goes over 21, it is a bust. There are other actions you can take as well, such as splitting if you have two cards with identical values, doubling your bet for just one card pull after you get your first two cards, and side bets for fun.

Description of Code

The code is first organized into structs rather than classes. There are three structs: one for the cards themselves, one for the deck of cards, and one for both players' hands. Many functions are employed afterwards. Create, Shuffle, and modDeck are first of three functions which set up the shuffled deck ready to be played. Assign and Name functions provide the player's hand with cards that can already be identified. Sum is the sum of card values, which is used to determine the outcome and also ace values. updateCard consists of three functions, Assign, Name, and Sum, which come together to make the requested number of cards. All three display functions serve to display menus, outputs, and hands. pullNew plays the game continuously until 21, bust, or the player stands. If the player stands, the Stand function is called, which makes the dealer pull a card until their sum surpasses 16. The dealer's value is then compared to the player's in the Outcome function, which outputs messages and returns a value based on the outcome. The returned value is employed for calculating the bet, where the bet is modified based on the outcome and then added to the total of money the player has gained throughout their rounds. splitCheck and splitPlay are functions that, respectively, check for cards that can be split and play split hands.

Sample Input/Output

The two images below feature the first round of a Blackjack game. The inputs are the bet, the action menu for picking a new card, making a stand, or terminating, and the menu for a new round or terminating. Hands are output, with the dealer's full hand only coming to light at the end of the game. The earnings are calculated upon initiating a new round to show its changes.

```
Initiating new game ...
Place your bet ($5.00 - $1000.00): $15.00
-=- Round Begin -=-

Dealer's cards:
[ --- ] [3 of Spades]

Your cards:   -=Total: 13=-
[6 of Diamonds] [7 of Hearts]

What will you do?
[1] Pick a new card
[2] Stand
[Any] Terminate
1

Your newest card: [6 of Spades]


Dealer's cards:
[ --- ] [3 of Spades]

Your cards:   -=Total: 19=-
[6 of Diamonds] [7 of Hearts] [6 of Spades]

What will you do?
[1] Pick a new card
[2] Stand
[Any] Terminate
2

-=- You have chosen to stand. -=-
```

```
Dealer's cards:   -=Total: 23=-
[J of Spades] [3 of Spades] [10 of Hearts]

Your cards:   -=Total: 19=-
[6 of Diamonds] [7 of Hearts] [6 of Spades]

=-= Dealer busts! Your victory =-=
        [D: 23 - P: 19]

=-=-=- Round Ended -=-=-=
Would you like to play another round?
[1] Yes
[Any] No
1
Your earnings: $30.00
```

There are some issues that can be encountered regarding output, however. Earnings may occasionally fail to register Blackjacks and only one of the split hands calculates the bet and output outcome, though the other hand may still output Blackjacks and busts.

Checkoff Sheet

1. Container classes (Where in code did you put each of these Concepts and how were they used?
    1. Sequences (At least 1)
        1. **list** (used to hold the cards for each hand)
        2. slist
        3. bit_vector
    2. Associative Containers (At least 2)
        1. set
        2. map
        3. hash
    3. Container adaptors (At least 2)
        1. **stack** (used to get data of a new card from a deck of 52)
        2. queue
        3. priority_queue
2. Iterators
    1. Concepts (Describe the iterators utilized for each Container)
        1. Trivial Iterator
        2. Input Iterator
        3. Output Iterator
        4. Forward Iterator
        5. **Bidirectional Iterator** (Through links, iterate to find matches and do calculations)

6. Random Access Iterator

3. Algorithms (Choose at least 1 from each category)
    1. Non-mutating algorithms
        1. for_each
        2. find
        3. count
        4. equal
        5. search
    2. Mutating algorithms
        1. copy
        2. **Swap** (swapped values of a deck to shuffle them)
        3. Transform
        4. Replace
        5. fill
        6. Remove
        7. Random_Shuffle
    3. Organization
        1. Sort
        2. Binary search
        3. merge
        4. inplace_merge
        5. Minimum and maximum

Documentation of Code