

Project 1

Blackjack

Christian Villanueva
CSC 17C - 43324
3 May 2021

Introduction:

I have created a program that runs a simple version of the card game Blackjack from scratch. This version of Blackjack is the card aspect only. I have not yet implemented a chip system for betting, nor have I included the ability to split when a pair is drawn. Blackjack is my favorite card game, and it seemed fitting to make, given the requirements for the project, and my prerequisite knowledge of the game's rules. The entire program was created from the ground up, and no reference code was used throughout the entirety of the project.

The player is dealt two cards and one of the dealer cards is shown. The player is then given the choice to "hit" or "stay". If the player chooses to hit, then the player is given another card. If the player's card total (each card is given a number value, and is added up to find the total) hits or exceeds 21, they automatically stay. The dealer then reveals their second card. After the dealer reveals their second card, if their total is less than or equal to sixteen, they draw until their total exceeds sixteen. The card total of the dealer and the player are compared, and whichever total is closer to 21 wins, given that the total does not exceed 21 (if the player or dealer exceeds a total of 21, they lose the game). If the difference from 21 is the same or if both the dealer and player exceed 21 or "bust" then the game results in a draw. If the player chooses to stay, then their total remains, and the dealer continues the same as previously described.

Summary:

Total Lines: 1086

Lines of Comments: 111

Number of Classes: 5

Number of Variables: 30

This project iteration took me about a week to complete, building off of my own Blackjack game from the past. Roughly 15 hours were put into the project including the documentation. The most challenging aspect of the project was implementing all of the required containers and algorithms into the program. I could not figure out how to include sets. Although I functionally understand how they work, implementing them into my project was difficult, and I ultimately could not think of a place for them.

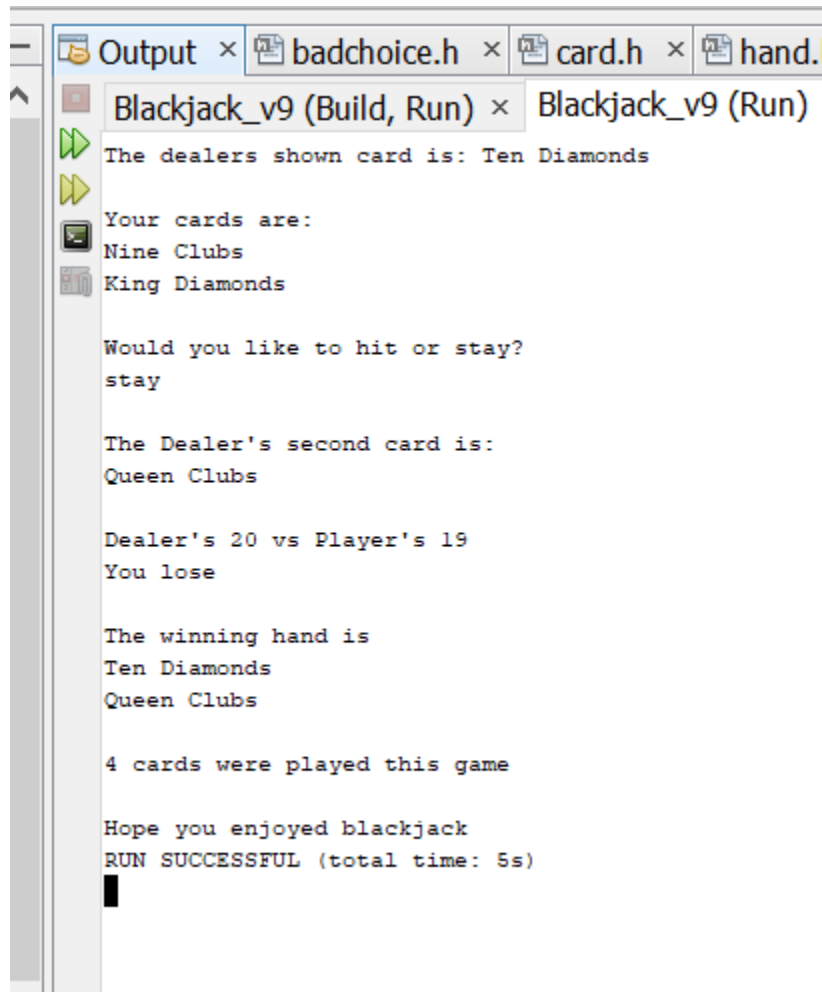
Github Link:

https://github.com/cv2808089/Villanueva_Christian_CIS_17/tree/master/CSC%2017C/Projects/Project%201

Description:

The game initially started as just a program that drew cards and output them. The card values and the ability to total them were then added, as well as the conditions for winning the game. Next came checks to ensure that repeat cards did not show up. After this came the rest of the concepts from the class up until this point such as pointers and binary files. After this came the splitting of the project into multiple source files, followed by conversion into classes. After converting into classes, I added exceptions, static variables and utilized a part of the STL. The next version added the copy constructor, and other minor tweaks. Following this was the addition of lists to the project. The final version added stacks, maps, and queues, which ultimately made the game more efficient, removing the need to check for repeat cards every time someone drew a card.

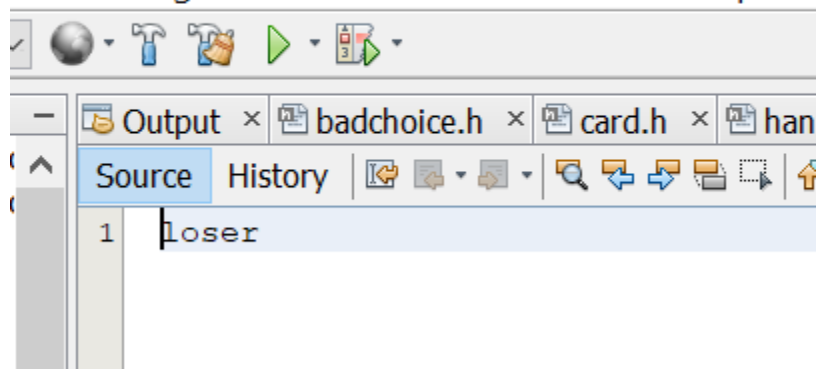
Sample output:



The screenshot shows a C++ IDE with a tabbed interface. The active tab is 'Output', which displays the following text:

```
Blackjack_v9 (Build, Run) x Blackjack_v9 (Run)
The dealers shown card is: Ten Diamonds
Your cards are:
Nine Clubs
King Diamonds
Would you like to hit or stay?
stay
The Dealer's second card is:
Queen Clubs
Dealer's 20 vs Player's 19
You lose
The winning hand is
Ten Diamonds
Queen Clubs
4 cards were played this game
Hope you enjoyed blackjack
RUN SUCCESSFUL (total time: 5s)
```

Run Debug Profile Team Tools Window Help



The screenshot shows a C++ IDE with a tabbed interface. The active tab is 'Source', which displays the following code:

```
1 loser
```

Output × badchoice.h × card.h × hand.

Blackjack_v9 (Build, Run) × Blackjack_v9 (Run)

The dealers shown card is: Six Diamonds

Your cards are:

Seven Clubs

Two Clubs

Would you like to hit or stay?

hit

You drew Seven Diamonds

Would you like to hit or stay?

hit

You drew Nine Diamonds

The Dealer's second card is:

Four Hearts

The dealer draws a card

The next card is

Five Spades

The dealer draws a card

The next card is

Queen Diamonds

Dealer's 25 vs Player's 25

No winner

No winner

The winning hand is

No winning hand

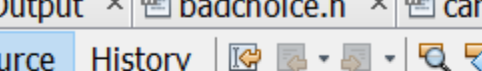





It was a draw

8 cards were played this game

Hope you enjoyed blackjack

RUN SUCCESSFUL (total time: 6s)

Output × badchoice.h × card.h ×

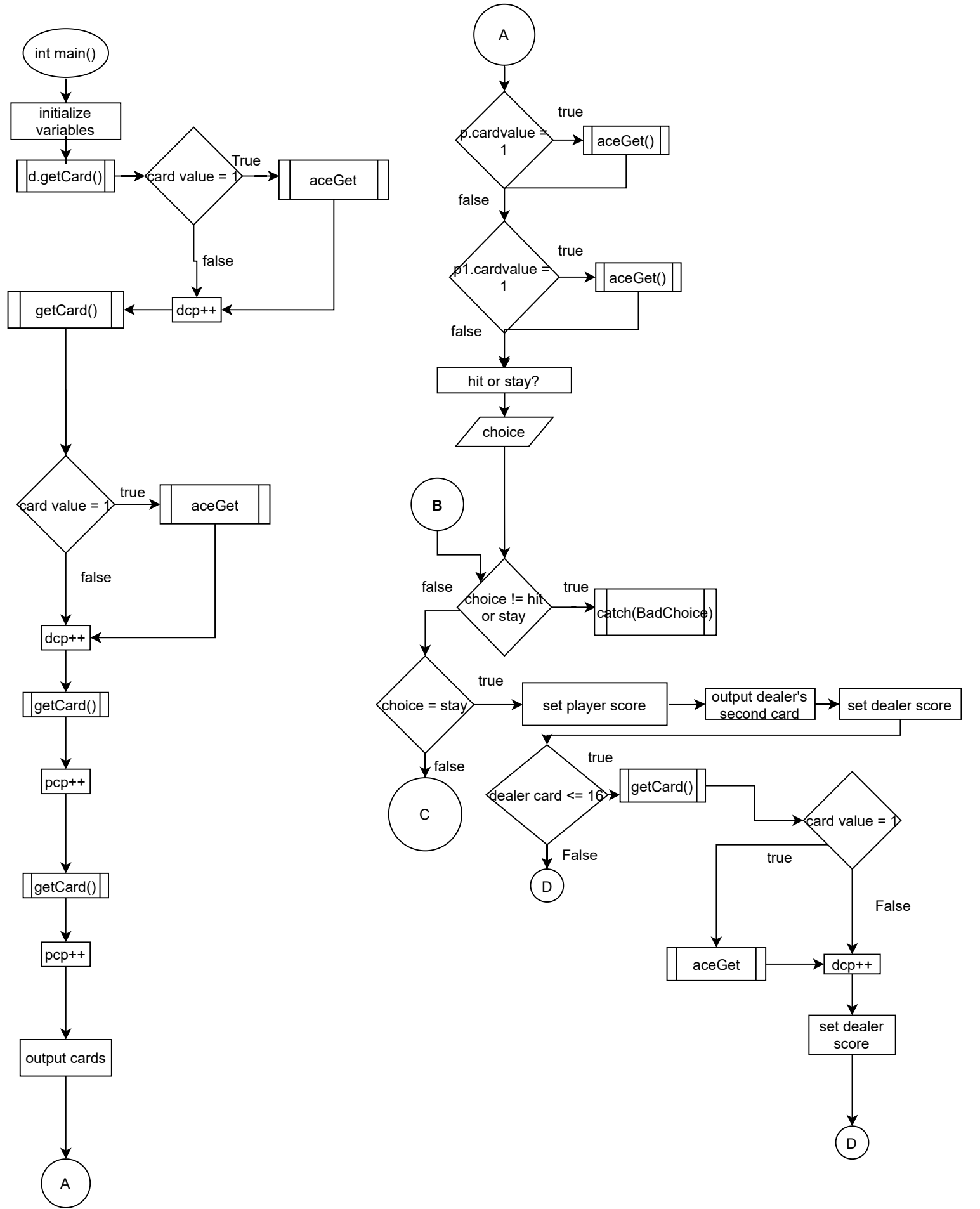
Source | History |      

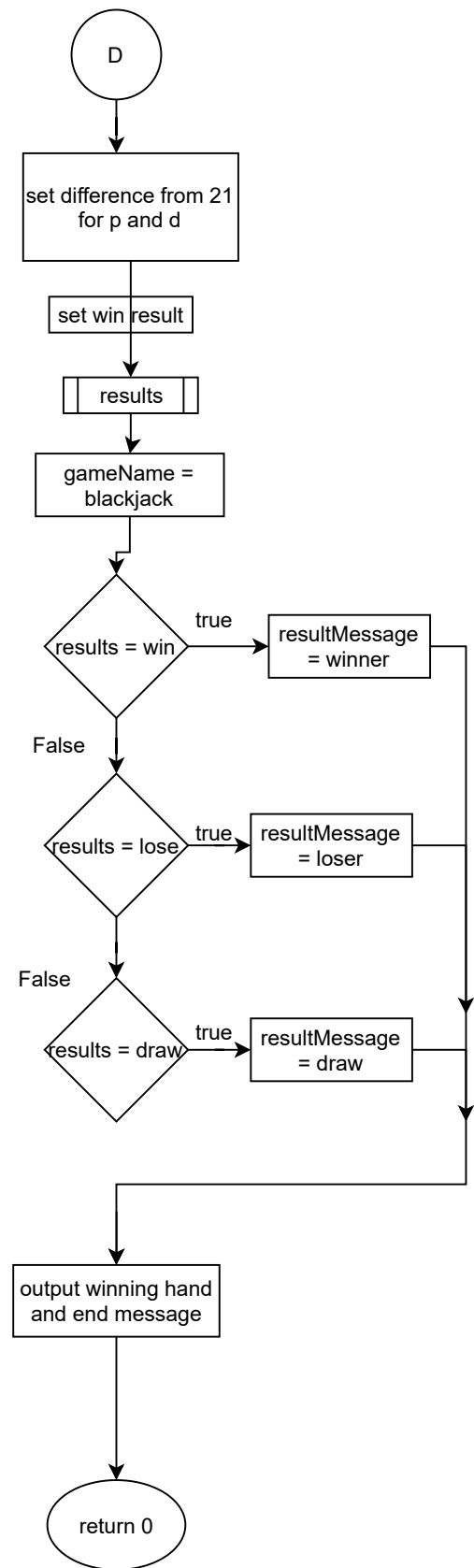
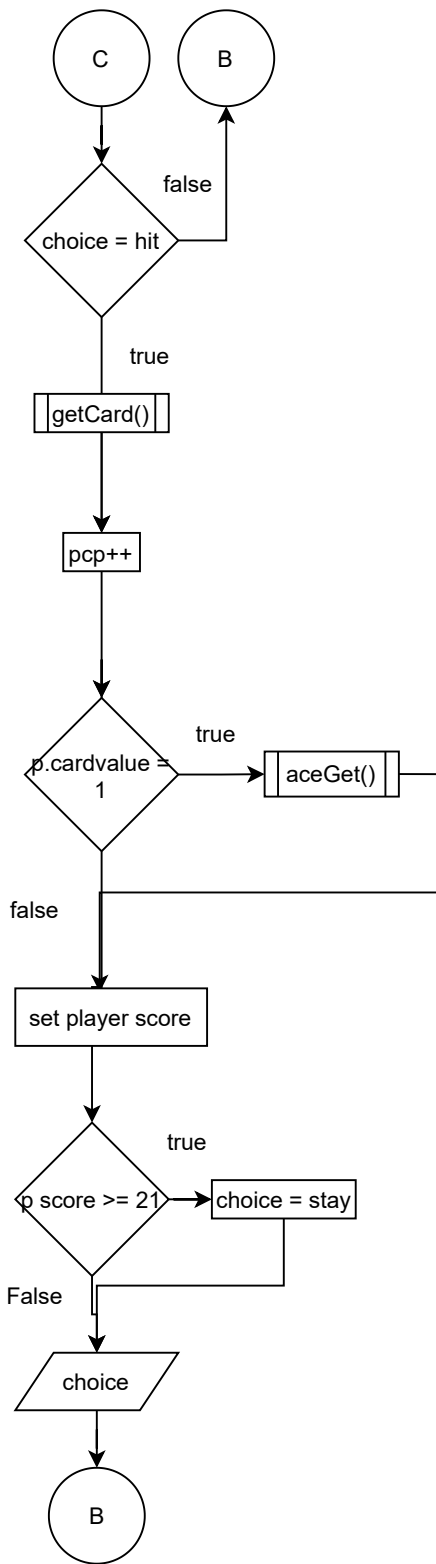
1 draw

Containers	File	Line
List	hand.h	16
Map	main.cpp	359
Stack	blackjack_imp	346
Queue	main.cpp	415
Iterators		
Input	hand.h	71
Output	hand.h	57
Bidirectional	main.cpp	398
Random Access	main.cpp	364
Algorithms		
Count	main.cpp	449
Random_shuffle	blackjack_imp	347

Variable Type	Name	Line	NOTE: LINE IS IN MAIN UNLESS OTHER FILE IS STATED			
int	cardValue	card.h, 12				
	cip	hand.h, 12				
	score	hand.h, 14				
	df21	hand.h, 18				
	aceResult	player.h, 9				
	chk	43				
	dcp	46				
	pcp	48				
	aceChoice	50				
	ac	player.cpp, 8				
string	cardName	card.h, 11				
	suit	card.h, 13				
	choice	41				
enum	gameResult	20				
gameResult	winc	20	<- in blackjack_implementation.cpp			
bool	res	hand.cpp: 16, 29, 42				
Card	hold	77	<- in blackjack_implementation.cpp			
	hold	hand.cpp, 7				
Hand	Dealer	42				
player	p1	40				
list<Card>	cards	hand.h, 16				
vector<Card>	deck	76	<- in blackjack_implementation.cpp			
	gameName	362				
stack<Card>	deck2	346	<- in blackjack_implementation.cpp			
	deck	54				
map<int, char>	resultMessage	359				
queue<Card>	winningHand	415, 426				
iterator	it	hand.h: 27, 34, 42, 49, 57, 64, 71, 81, 92				
	i	364				

Note: I use abbreviations for variable names





Version 11 Pseudocode

Main

{

Set random number seed

Define classes and variables

Draw dealer cards

If an ace is drawn, make its value 11 if it doesn't bust the dealer

Increment number of cards in play

Draw player cards

Increment number of cards in play

Output dealer's first card and player's two cards

If the player draws a(n) ace(s) gets their choice for the aces value(only allows 1 or 11)

Gets the player's choice to hit or stay with input validation

If stay is chosen,

Calculates player and dealer card total

Reveals dealer's second card

If dealer's card total is ≤ 16 ,

draws cards, iterates cards in play

repeats until a total of >16 is achieved

If dealer draws ace, sets it to 11 if it doesn't bust the dealer

Calculates dealer card total

Output dealer's cards

```
If hit is chosen, loops until stay is chosen
    Calculates player and dealer card total
    Draw card and iterate cards in play

    Display card drawn
    If ace is drawn, get choice for ace value(only allows 1 or 11)
    Add card value to player card total

    If player card total >= 21, set choice to stay
    If player card total < 21, get choice to hit or stay(with input validation)

    Reveals dealer's second card
    If dealer's card total is <= 16,
        draws cards, iterates cards in play, and checks
        repeats until a total of >16 is achieved
        If dealer draws ace, sets it to 11 if it doesn't bust the dealer
        Calculates dealer card total
        Output dealer's cards

Find difference of dealer and player total from 21
Find and output game results based on difference

Output the game results

Define map and vector to hold result message and game name
If win result is win, message is winner
If win result is lose, message is loser
If win result is draw, message is draw

Write the result message to a binary file

Outputs queue of winning hand
Output the number of cards used in the game
Output exit message

Free up the used memory and exit the program

}
```

Card
-cardName: string -cardvalue; int -suit: string
+getCardValue(): int +getCardName(): string +getSuit(): string +setCardValue(n: int): void +setCardName(n: string): void +setCardSuit(s: string): void

Hand
-cip: int -score: int -cards: list<Card>
+Hand(): +~Hand(): +Hand(object: Hand&): +getCardsVal(pos: int): int +getCardsNam(pos: int):string +getCardsSu(pos: int): string +setCardsVal(pos: int, val: int): void +setCardsNam(pos: int, s: string): void +setCardsSu(pos: int, s: string): void +drawCard(pos: int): void +aceGet(pos: int): virtual void +setScore(pos: int): void +getScore(): int +setdf21(): void +getdf21(): int +cippp(): void +getCip(): int +operator < (const Hand&): bool +operator > (const Hand&): bool +operator == (const Hand&): bool

BadPos

BadChoice

Player
-aceResult: int
results(int, int, int, int): void aceGet(): void getAceRes(): int

