

Project 2

<Blackjack (21)>



June 13, 2025

Name: Belarmino, J miguel

CIS-5 42565

**My Github link:**

<https://github.com/JMiguelB/BlackjackgameProject>

**Introduction**

Title: Blackjack(21) Final Version

This game project is an upgraded version of my previous simplified Blackjack game which was Project 1. In the original version, the game used only randomized number values to represent player and dealer hands. This final version introduces a full 52 card deck with suits and face values, giving the game a more realistic casino feel. Upon launch, the player enters their name, which allows the game to load and save past performance. The deck is built, sorted, and displayed before being shuffled. Each round deals two cards to the player and one to the dealer. The player can choose to hit or stand, while Aces dynamically adjust their value from 11 to 1 to avoid busting. The dealer follows a fixed rule, drawing until reaching at least 17. After both turns, the winner is determined based on total values. The game continues until the user opts out, with performance stats saved between sessions.

**Summary**

Project size: about 335 lines

The number of variables: about 25

The number of Library: 6

This project covers all the key concepts we’ve learned from class and goes well beyond the minimum 250 lines. The code is organized into functions, easy to follow, and includes clear comments throughout. I tried to make it as modular and clean as possible. For future upgrades, I’d like to expand the game to support multiple players add a betting feature to make it more interactive and maybe switch to using classes or structs for a more object oriented design. It would also be interesting to keep a full game history by logging more detailed info into files.

I picked this card game because it’s one of my favorite games to play. It made the project more engaging to me.

**Project 1**

This 1st Version took me about 35 hours and still counting, from brainstorming, researching and sketching ideas on paper to writing and putting all the codes together. Even though there were many struggles along the way, solving each problem felt really rewarding especially when I got it to work on how I want it to work. Throughout the process, I kept thinking of new ideas and features to add, which leave me unsatisfied for some parts but overall, this was a great learning experience. I was able to reflect on what I’ve learned so far and even learned some more along the way. This game still has more potential and I look forward to make some improvements on it.

**V.2.1**

This Version of my project 2 took significant time and effort, I spent most of my time turning big chunks of code into cleaner, reusable functions. It wasn’t always easy, some parts broke along the way, but it was worth it. Organizing things into functions made the code easier to understand and way less messy. I learned a lot just from figuring out how to make each part work on its own while still fitting into the bigger picture. This version feels like a real improvement, and it’s nice to see everything running smoother because of it.

**V.2.2**

This version of my Blackjack project added a full deck using suits and face cards, making the game feel a lot more realistic. I built the deck by combining face values with suits, assigned card values (like Jack, Queen, King = 10, Ace = 11), and added a shuffle function so the cards aren’t in the same order every time. One challenge was making sure cards are drawn correctly using topCard++, and reshuffling when the deck gets low. It was satisfying to see the gameplay feel more like a real Blackjack table. Adding the ability to draw from a full, randomized deck definitely leveled up the experience and brought the project one step closer to feeling complete.

**V.2.3**

In this version of my Blackjack project, I focused on improving gameplay by correctly handling Aces. Aces now start with a value of 11 but can change to 1 if a player or dealer is about to bust, just like in a real Blackjack game. I added a forAces function to track the number of Aces and adjust their values dynamically during play. It was tricky at first to manage both the player’s and dealer’s Aces separately, but once I set up ace counters for each and passed them through the game, it worked smoothly. This fix made the game more realistic and balanced, preventing players or the dealer from losing unfairly when they draw multiple Aces. Version 3 definitely made the gameplay smarter and more strategic.

**V.2.4 (FINAL VERSION)**

For the final version of my Blackjack project, I focused on finishing all remaining tasks and cleaning up the structure of the game. One of the main changes I made was fixing the use of global constants. Based on the project checklist, I learned that only physical constants or conversions should be global, so I moved the game-related constants into the main function. I also added sorting and display features to show the deck both before and after shuffling, which made the game look more organized and complete. I made sure to go through the checklist one last time to confirm that everything was in place, including win tracking and saving progress. This version feels like a solid, finished product.

**Pseudo Code**

*Initialize Libraries*

*Initialize Function Prototypes*

*Execution begins here at main*

Set *random number generator*

*Declare Variables*

*Face values, suits, and corresponding numerical values*

*//Map/Process the Inputs -> Outputs While read file*

*Build the deck function*

*sort the deck function*

*Show deck function*

*shuffle deck function*

*Show deck function*

*Check saved game by name function*

*Start main game loop*

*Do while loop*

*restart deck if less than 10 cards*

*Initialize Variables*

*both starts at the total value of 0*

*User's two initial card*

*dealer's initial card*

*User's turn functions*

*Dealer/computer's turn if player hasn't busted function*

*counts the rounds*

*calculate of win rate*

*Ask Useer if they want to play again*

*While loop play*

*Clean up memory and files*

*Save game function*

*Exit the Program*

*Check saved game by name function*

*Prompt the user to enter their name*

*Input file streams*

*Attempt to load previous progress*

*read data*

*if found resume round*

*if not new gane*

*User's turn functions*

*Do while loop*

*User's decision loop for hit or stand*

*If hit new total*

*For Aces function 11/1*

*if its over 21 game over*

*exit loop*

*if player hits 21*

*increment user's win count*

*exit loop*

*while loop*

*Dealer/computer's turn if player hasn't busted*

*Dealer's hidden second card*

*For Aces function 11/1*

*While loop*

*Dealer hits until 17 or more*

*For Aces function 11/1*

*If goes over max value*

*increment user's win count*

*else if*

*dealer has more than the user*

*else*

*increment user's win count*

*Save game function*

*Output file streams/save*

*Save game results to file*

*Build the deck function*

*Index for filling the deck*

*Loop through 4 suits*

*Combine*

*assign card value*

*shuffle deck function*

*random from deck size*

*swap strings*

*swap values*

*For Aces function 11/1*

*convert an Ace from 11 to 1*

*one less Ace counted as 11*

*if true*

*swap the values and card names function*

*Swap integer values*

*swap card strings*

*bubble sort but custom function*

*do while*

*pass the face of the first card*

*pass the face of the 2nd card*

*push Aces value 11 to the end*

*otherwise, sort numerically*

*display function*

*for loop*

*show cards*

*help swap the Ace at the end custom sort function*

*while loop*

*Stop at the first space*

*Return*

**Screenshots of input/output**

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.