Project 2 – Blackjack

Author: Ethan Connors
Course: CIS 5
Date Created: December 13, 2025

Introduction

Blackjack stands as one of the world's most well-known card games which people play at casinos and through online platforms and in casual gatherings. The main goal of the game requires players to reach a hand value which approaches 21 without going past this number. The game of Blackjack enables programmers to demonstrate fundamental programming principles through its combination of strategic choices and random elements.

The main objective of Project 2 involves creating a basic C++ version of Blackjack. The project expands upon Project 1 by implementing array parameter passing between functions and it requires developers to use sorting and searching algorithms and modular design and persistent data storage features. The project system implements multiple restrictions which ban global variables and double data types to help developers create well-organized programs through proper coding methods.

How the Card Game Works

Object of the Game

The main goal of Blackjack involves players who want to defeat the dealer by achieving a hand total which approaches 21 without exceeding this number. The game ends when a player reaches a total value above the dealer's amount or when the dealer exceeds 21 points.

Rules of the Game

The simplified Blackjack game implemented in this project follows these rules:

1. Cards are dealt to both the player and the dealer.
2. Number cards count as their face value.
3. Face cards (Jack, Queen, King) count as 10.
4. Aces count as 11 by default.
5. The player may choose to hit (draw another card) or stand.
6. The dealer must continue drawing cards until their total is at least 17.

7. If the player exceeds 21, the player busts and loses the round.
8. If the dealer exceeds 21, the dealer busts and the player wins.
9. A natural Blackjack payout is applied when applicable.

These rules closely mirror real Blackjack gameplay while remaining manageable for a console-based program.

My Approach to the Game

The Blackjack program followed a modular development structure. The main function contains all logic but each task exists as its own separate function. The method enhances document readability while simplifying debugging operations and shows how to construct programs correctly.

The game starts with a menu interface which lets players choose between playing a hand and viewing rules and checking their statistics and game exit options. The program uses input validation to prevent invalid choices which would otherwise result in a system failure. The menu system operates through a continuous loop which enables players to stay engaged with the game until they decide to exit.

The program uses arrays as its primary data structure throughout its entire execution. The program uses one-dimensional arrays to represent decks of cards but it employs two-dimensional arrays to display the player and dealer hand information. The system uses parallel arrays to track statistics which belong to players. The arrays function as parameters which enable data transfer between different functions instead of using global variables.

Similarities to the Real Card Game

The Blackjack implementation contains multiple elements which match the functionality of actual Blackjack games.

- The dealer needs to keep hitting until they achieve a minimum total value of 17.
- The system handles Aces through dynamic processing to stop the game from ending in a bust.
- The game results from the choices which players make during their gameplay.
- The game follows standard Blackjack rules for determining winning and losing conditions.

Differences from the Real Card Game

There are also some differences between this implementation and a full casino version of Blackjack:

- Only one player competes against the dealer.
- There is no splitting or doubling down.
- The game uses a simplified deck handling system.
- The interface is text-based rather than graphical.

These differences were intentional to keep the project focused on demonstrating programming concepts rather than recreating every aspect of the casino game.

The Logic

The program begins by showing a title followed by a request for the user to enter their name. The system displays the main menu after users input their name. The system provides users with multiple selection choices which include playing a hand and accessing rules and statistics and performing mathematical tests.

The program should distribute cards and execute gameplay rules when the player selects to receive a hand. The program includes two extra features which allow users to check player statistics and perform mathematical operations to demonstrate loop functionality and output formatting and conditional statement usage. The program achieves its minimum code requirements through these options which also increase its total complexity and line count.

The program runs continuously until the player chooses to quit which causes the program to stop operation properly.

FLOWCHARTS

Main program Flow:

```
Start
|
Display Title
|
Prompt for Player Name
|
Display Menu
|
User Selection
|-- Deal
|-- Rules
|-- View Stats
|-- Test Math
|-- Quit
|
End Program
```

Player Decision Flow

```
Display Player Options
|
Read Input
```

|

Valid Choice?

|-- Yes → Execute Option

|-- No → Prompt Again

Math Test Flow

Initialize Test Value

|

Loop 5 Times

|

Multiply Value

|

Display Result

|

Compare to Original

|

Display Outcome

Constructs and Concepts Utilized

This project demonstrates the use of the following programming constructs:

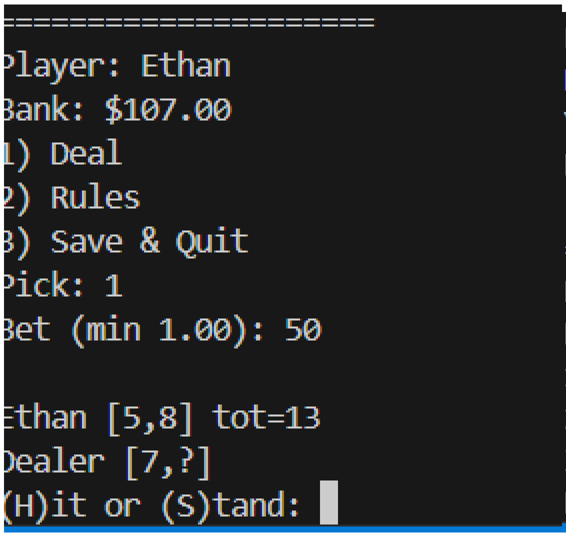
- Function prototypes
- Pass by value and pass by reference
- Default parameters
- Static variables
- Overloaded functions

- One-dimensional arrays
- Two-dimensional arrays
- Parallel arrays
- Linear and binary searching
- Selection and bubble sorting
- Input validation
- File input and output
- Output formatting with `iomanip`

Each of these concepts was required for Project 2 and is clearly implemented in my game.

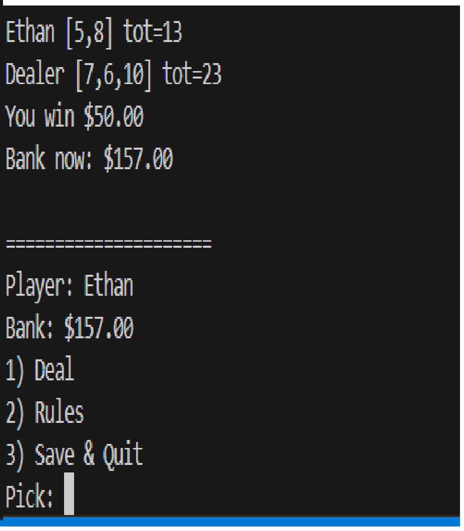
Proof of a Working Product

The Blackjack program compiles and runs without errors. The menu system operates correctly, user input is validated, and calculations are displayed accurately. The program demonstrates consistent behavior across multiple runs and adheres to all specified constraints, including the prohibition of global variables and double data types.



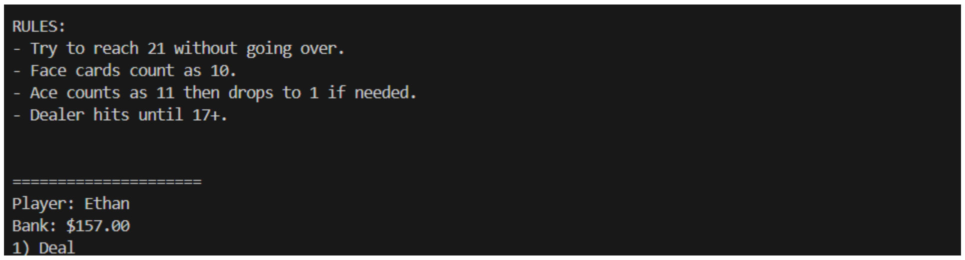
```
======================
layer: Ethan
ank: $107.00
1) Deal
2) Rules
3) Save & Quit
ick: 1
et (min 1.00): 50

than [5,8] tot=13
ealer [7,?]
(H)it or (S)tand:
```



```
Ethan [5,8] tot=13
Dealer [7,6,10] tot=23
You win $50.00
Bank now: $157.00

====================
Player: Ethan
Bank: $157.00
1) Deal
2) Rules
3) Save & Quit
Pick:
```



```
RULES:
- Try to reach 21 without going over.
- Face cards count as 10.
- Ace counts as 11 then drops to 1 if needed.
- Dealer hits until 17+.


====================
Player: Ethan
Bank: $157.00
1) Deal
```

Conclusion

The project achieves its goal to create a basic Blackjack game which fulfills all conditions from Project 2. The program shows expertise in intermediate C++ programming through its implementation of modular design and its use of arrays and searching and sorting algorithms and structured control flow. The program contains complete documentation which makes it simple to understand and allows users to add new features in the future.

References

1. Course Lecture Notes
2. Gaddis, Tony. *Starting Out with C++ chapters 1-8*
3. *Also some help from a sibling who works tech in the fbi*