**Project #2**

**BlackJack**

****

**Javier Paz**

**CIS 5**

**July 30, 2023**

**Rules for the Blackjack V5 game:**

1. Objective:

The objective of Blackjack is to get a hand value as close to 21 as possible without exceeding it. Players compete against the dealer, and the hand closest to 21 wins.

2. Card Values:

- Numbered cards (2 to 10): Their face value is their point value.

- Face cards (Jacks, Queens, Kings): They are worth 10 points each.

- Ace: Can be worth either 1 or 11 points, depending on which value benefits the hand more.

3. Game Setup:

- The game starts with the user choosing the number of players (1 to 8) participating in the game.

- Each player begins with a balance of $1000.

-The initial bet must be between 0 and the maximum bet (MAX\_BET).

4. Gameplay:

- Each player is dealt two initial cards.

- Players have the option to "split" their hand if they are dealt two cards of the same rank.

- Players take turns to decide whether to "hit" (receive an additional card) or "stand" (keep their current hand).

- Players can "double down" on their initial hand, receiving one more card and doubling their bet.

- Players who exceed 21 points "bust" and lose the round.

5. Dealer's Turn:

- After all players have completed their turns, the dealer plays their hand.

- The dealer must hit until they reach a minimum point threshold (DLR\_MIN = 17).

- If the dealer busts (exceeds 21 points), all remaining players win.

6. Winning and Losing:

- If a player's hand is closer to 21 than the dealer's hand without exceeding 21, the player wins.

- If a player's hand ties with the dealer's hand, it's a push (a tie), and the player's bet is returned.

- If the dealer's hand is closer to 21 or both the dealer and the player bust, the player loses.

7. Betting:

- Players place bets at the beginning of each round.

- Bets must be within the range of MIN\_BET ($10) and MAX\_BET ($100).

8. Double Down:

- Players can double down only on their initial hand.

- If a player doubles down and wins, their winnings are doubled.

9. Display:

- Cards are displayed in a graphical representation.

- The leaderboard shows the names of players and their current balances.

10. Loop and Exit:

- After each round, players are asked if they want to play again (y/n).

- If players choose not to play again, the game ends with a goodbye message.

**Summary of Blackjack Game:**

The "Blackjack" is a simplified version of the classic Blackjack card game. The game is played between the players and the dealer, and the objective is to get a hand total as close to 21 points as possible without exceeding it.

**Summary:**

Project size: 328 lines

Number of variables: 30 variables

This C++ code, titled "BlackJack V5," is a text-based implementation of the popular card game Blackjack. The program allows multiple players to participate and plays against a computer-controlled dealer. The main function of the program includes a game loop that enables players to play multiple rounds until they decide to quit.

The program starts by displaying a welcome message and prompts the user to enter the number of players (ranging from 1 to 8). Each player is then asked to input their name, and their initial balance is set to $1000.

The gameplay follows standard Blackjack rules, where each player is dealt two random cards at the beginning. The face cards (Jack, Queen, King) have a point value of 10, and Aces can be worth either 1 or 11 points, depending on the player's preference.

Players are then presented with their hands, and they have the option to "hit" (receive additional cards), "stand" (end their turn), or "double down" (double their bet and receive one more card). The game automatically checks if a player has a pair of the same rank in their initial hand and allows them to "split" their hand into two separate hands.

After players have completed their turns, it is the dealer's turn. The dealer must hit until their hand's total points reach or exceed 17.

At the end of each round, the program displays the leaderboard, showing the names of all players and their respective balances. Players are also prompted to play again, and the game continues until they choose to quit.

The code uses functions to perform various tasks, such as getting random card values, displaying the graphical representation of cards, displaying card ranks and suits, and displaying the leaderboard.

Overall, "BlackJack V5" offers an interactive and engaging Blackjack experience for multiple players, allowing them to enjoy the excitement of the classic card game in a command-line environment.

**FlowChart**

A diagram of a game

Description automatically generated

A screenshot of a diagram

Description automatically generated





**Pseudo Code:**

1. Start

2. Function getRandom():

a. Generate a random integer between 2 and 14 (inclusive).

b. If the generated value is greater than 10:

i. Return 10.

c. Otherwise, return the generated value.

d. End Function

3. Function displayCard(int cardValue):

a. Display the graphical representation of the card based on the cardValue:

i. If cardValue is between 2 and 9:

- Display the card value as a digit.

ii. If cardValue is 10:

- Display "10".

iii. If cardValue is 11:

- Display "J".

iv. If cardValue is 12:

- Display "Q".

v. If cardValue is 13:

- Display "K".

vi. If cardValue is 14:

- Display "A".

b. End Function

4. Function displayCardInfo(int cardValue):

a. Display the rank and suit of the card based on the cardValue:

i. If cardValue is between 2 and 10:

- Display the card value as it is.

ii. If cardValue is 11:

- Display "Jack".

iii. If cardValue is 12:

- Display "Queen".

iv. If cardValue is 13:

- Display "King".

v. If cardValue is 14:

- Display "Ace".

vi. Generate a random number between 0 and 3 to represent the suit of the card:

- 0: Hearts

- 1: Diamonds

- 2: Clubs

- 3: Spades

vii. Display the suit based on the generated random number.

b. End Function

5. Function displayLeaderboard(string playerNames[], float playerBalances[], int numPlrs):

a. Display the leaderboard with the player names and their corresponding balances.

b. End Function

6. Function main():

a. Initialize constants:

- HAND\_SIZE = 2

- DLR\_MIN = 17

- MAX\_BET = 100.0

- MIN\_BET = 10.0

b. Seed the random number generator using the current time.

c. Display "Welcome to Blackjack!".

d. Get the number of players (numPlrs) from the user:

- Prompt the user to enter the number of players between 1 and 8.

- Repeat until a valid input is provided.

e. Initialize arrays to store player names, balances, and hands:

- playerNames: string array of size 8 (maximum number of players).

- playerBalances: float array of size 8 (maximum number of players).

- playerHands: 2D int array of size 8x2 (maximum number of players and hand size).

f. Start the game loop:

- Initialize playAgain to 'y' or 'Y'.

g. Inside the game loop:

i. Prompt each player for their name and initial bet:

- Use cin.ignore() to consume the newline character left in the stream after cin >> numPlrs.

- Repeat for each player.

- Validate the initial bet to be between 0 and MAX\_BET.

- Deduct the initial bet from the player's balance.

ii. Initialize the round for each player:

- Generate random cards for each player's hand.

- Check if the player can split the hand and ask if they want to split.

- If the player chooses to split, create a second hand and update points accordingly.

iii. For each split hand and each player:

- Display the current player's turn or split hand.

- Display the cards and points for the player's hand.

- Ask the player to hit, stand, or double down until they stand or bust.

- If the player doubles down, deal one more card and update the points.

- If the player busts, ask for a new bet.

iv. Dealer's turn:

- Generate random cards for the dealer's hand.

- Display the dealer's cards and points.

- The dealer hits until reaching the minimum points (DLR\_MIN).

v. Determine the winners and update player balances:

- Compare the player's and dealer's points to determine the winner.

- If the player wins, ask for a new bet and update the balance.

- Apply the double down feature on player's winnings if applicable.

vi. Display the leaderboard.

vii. Ask if the players want to play again.

h. End the game loop when the players choose not to play again.

i. Display "Thanks for playing Blackjack! Goodbye!".

7. End

**GamePlay Picture (Sample I/O)**

Winning Hand:

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

Losing Hand:

A screenshot of a computer

Description automatically generated

**Game Code**

/\*

\* Author: Javier Paz

\* Created on July 25th, 2023 10:00 pm

\* Purpose: BlackJack V5

\*/

#include <iostream>

#include <iomanip>

#include <cmath>

#include <cstdlib>

#include <ctime>

#include <fstream>

#include <string>

using namespace std;

// Function to get a random card value (2 to 11, K, Q, J set to 10)

int getRandom() {

int cardValue = rand() % 13 + 2;

if (cardValue > 10) {

return 10;

}

return cardValue;

}

// Function to display the graphical representation of cards

void displayCard(int cardValue) {

cout << " \_\_\_\_\_ " << endl;

cout << "| |" << endl;

cout << "| ";

if (cardValue >= 2 && cardValue <= 9)

cout << static\_cast<char>('0' + cardValue);

else if (cardValue == 10)

cout << "10";

else if (cardValue == 11)

cout << "J";

else if (cardValue == 12)

cout << "Q";

else if (cardValue == 13)

cout << "K";

else if (cardValue == 14)

cout << "A";

cout << " |" << endl;

cout << "| |" << endl;

cout << " ----- " << endl;

}

// Function to display the rank and suit of a card

void displayCardInfo(int cardValue) {

cout << "Rank: ";

if (cardValue >= 2 && cardValue <= 10)

cout << cardValue;

else if (cardValue == 11)

cout << "Jack";

else if (cardValue == 12)

cout << "Queen";

else if (cardValue == 13)

cout << "King";

else if (cardValue == 14)

cout << "Ace";

cout << ", Suit: ";

int suit = rand() % 4;

switch (suit) {

case 0:

cout << "Hearts";

break;

case 1:

cout << "Diamonds";

break;

case 2:

cout << "Clubs";

break;

case 3:

cout << "Spades";

break;

default:

break;

}

cout << endl;

}

// Function to display the leaderboard

void displayLeaderboard(string playerNames[], float playerBalances[], int numPlrs) {

cout << "\n-------- Leaderboard --------" << endl;

cout << "Player\t\tBalance" << endl;

cout << "----------------------------" << endl;

for (int i = 0; i < numPlrs; ++i) {

cout << setw(10) << playerNames[i] << "\t$" << playerBalances[i] << endl;

}

cout << "----------------------------" << endl;

}

int main() {

const int HAND\_SIZE = 2;

const int DLR\_MIN = 17;

const float MAX\_BET = 100.0;

const float MIN\_BET = 10.0;

// Seed the random number generator

srand(static\_cast<unsigned int>(time(nullptr)));

cout << "Welcome to Blackjack!" << endl;

// Get the number of players from the user

int numPlrs;

do {

cout << "Enter the number of players (1-8): ";

cin >> numPlrs;

} while (numPlrs < 1 || numPlrs > 8);

// Player names, balances, and hands arrays

string playerNames[8]; // Assuming the maximum number of players is 8

float playerBalances[8]; // Assuming the maximum number of players is 8

int playerHands[8][HAND\_SIZE]; // Assuming the maximum number of players is 8

// Game loop

char playAgain;

do {

// Player names input and write to file

for (int i = 0; i < numPlrs; ++i) {

cin.ignore(); // To consume the newline character left in the stream after cin >> numPlrs;

cout << "Enter the name of Player " << i + 1 << ": ";

getline(cin, playerNames[i]);

playerBalances[i] = 1000.0; // Initialize all players with $1000

// Ask for bet initialization after each round for each player

float initialBet;

while (true) {

cout << "Enter your initial bet (0-" << static\_cast<int>(MAX\_BET) << "): ";

cin >> initialBet;

if (initialBet >= 0 && initialBet <= MAX\_BET) {

playerBalances[i] -= initialBet; // Deduct the initial bet from the player's balance

break;

}

cout << "Invalid bet. Please try again." << endl;

}

}

// Initialize the round for each player

for (int i = 0; i < numPlrs; ++i) {

int plrPoints = 0;

int splitHand[2] = {0};

int splitPoints[2] = {0};

int numSplits = 0;

bool canSplit = false;

for (int j = 0; j < HAND\_SIZE; ++j) {

// Deal initial cards to the player

playerHands[i][j] = getRandom();

plrPoints += playerHands[i][j];

// Check if the player can split the hand

if (j == 1 && playerHands[i][0] == playerHands[i][1]) {

canSplit = true;

}

}

if (canSplit) {

char splitChoice;

cout << "Do you want to split your hand? (y/n): ";

cin >> splitChoice;

if (splitChoice == 'y' || splitChoice == 'Y') {

numSplits = 1;

splitHand[0] = playerHands[i][0];

splitHand[1] = getRandom();

plrPoints -= splitHand[0];

plrPoints += splitHand[1];

splitPoints[0] = splitHand[0];

splitPoints[1] = splitHand[1];

}

}

for (int splitIndex = 0; splitIndex <= numSplits; ++splitIndex) {

if (splitIndex == 1) {

cout << "\n" << playerNames[i] << "'s split hand:" << endl;

plrPoints = splitPoints[1];

} else {

cout << "\n" << playerNames[i] << "'s turn:" << endl;

}

for (int j = 0; j < HAND\_SIZE; ++j) {

if (splitIndex == 1 && j == 0) {

continue; // Skip displaying the first card of the second hand during the split

}

cout << "Card " << j + 1 << ":\n";

displayCard((splitIndex == 1) ? splitHand[j] : playerHands[i][j]);

displayCardInfo((splitIndex == 1) ? splitHand[j] : playerHands[i][j]);

}

cout << "Points: " << setw(7) << ((plrPoints > 21) ? "Bust!" : to\_string(plrPoints)) << endl;

// Ask the player to hit, stand, or double down until they stand or bust

bool doubleDown = false; // Initialize doubleDown variable

while (plrPoints < 21) {

char choice;

cout << "Do you want to hit (h), stand (s), or double down (d)? ";

cin >> choice;

switch (choice) {

case 'h':

// Deal another card to the player

playerHands[i][HAND\_SIZE - 1] = getRandom();

plrPoints += playerHands[i][HAND\_SIZE - 1];

for (int j = 0; j < HAND\_SIZE; ++j) {

cout << "Card " << j + 1 << ":\n";

displayCard(playerHands[i][j]);

displayCardInfo(playerHands[i][j]);

}

cout << "Points: " << setw(7) << ((plrPoints > 21) ? "Bust!" : to\_string(plrPoints)) << endl;

break;

case 's':

// Player stands

break;

case 'd':

// Double down

if (HAND\_SIZE == 2) {

playerHands[i][1] = getRandom();

plrPoints += playerHands[i][1];

doubleDown = true;

} else {

cout << "You can only double down on your initial hand!" << endl;

}

for (int j = 0; j < HAND\_SIZE; ++j) {

cout << "Card " << j + 1 << ":\n";

displayCard(playerHands[i][j]);

displayCardInfo(playerHands[i][j]);

}

cout << "Points: " << setw(7) << ((plrPoints > 21) ? "Bust!" : to\_string(plrPoints)) << endl;

break;

default:

cout << "Invalid choice. Please try again." << endl;

}

if (choice == 's' || doubleDown) {

// Player stands or has double down

break;

}

}

if (plrPoints > 21) {

cout << "Bust! You lose." << endl;

float bet;

while (true) {

cout << "Enter your bet (0-" << static\_cast<int>(MAX\_BET) << "): ";

cin >> bet;

if (bet >= 0 && bet <= MAX\_BET) {

break;

}

cout << "Invalid bet. Please try again." << endl;

}

} else {

// Dealer's turn

int dlrPoints = 0;

int dlrHand[HAND\_SIZE];

dlrHand[0] = getRandom();

dlrPoints += dlrHand[0];

cout << "\nDealer's turn:" << endl;

cout << "Card 1:\n";

displayCard(dlrHand[0]);

displayCardInfo(dlrHand[0]);

cout << "Points: " << setw(7) << dlrPoints << endl;

// Dealer hits until reaching the minimum points

while (dlrPoints < DLR\_MIN) {

dlrHand[HAND\_SIZE - 1] = getRandom();

dlrPoints += dlrHand[HAND\_SIZE - 1];

cout << "Card " << HAND\_SIZE << ":\n";

displayCard(dlrHand[HAND\_SIZE - 1]);

displayCardInfo(dlrHand[HAND\_SIZE - 1]);

cout << "Points: " << setw(7) << dlrPoints << endl;

}

float bet;

bool playerWins = false;

if (dlrPoints > 21 || plrPoints > dlrPoints) {

cout << "Congratulations! You win." << endl;

playerWins = true;

while (true) {

cout << "Enter your bet (0-" << static\_cast<int>(MAX\_BET) << "): ";

cin >> bet;

if (bet >= 0 && bet <= MAX\_BET) {

break;

}

cout << "Invalid bet. Please try again." << endl;

}

} else if (plrPoints == dlrPoints) {

cout << "It's a tie!" << endl;

} else {

cout << "You lose." << endl;

while (true) {

cout << "Enter your bet (0-" << static\_cast<int>(MAX\_BET) << "): ";

cin >> bet;

if (bet >= 0 && bet <= MAX\_BET) {

break;

}

cout << "Invalid bet. Please try again." << endl;

}

}

// Apply the double down feature on player's winnings

if (doubleDown && playerWins) {

bet \*= 2;

cout << "Congratulations! Your winnings are doubled due to double down!" << endl;

}

// Update the player's balance based on the bet outcome

playerBalances[i] += (playerWins ? bet : -bet);

cout << "Current Balance: $" << playerBalances[i] << endl;

}

}

}

// Display the leaderboard

displayLeaderboard(playerNames, playerBalances, numPlrs);

// Ask if the players want to play again

cout << "\nDo you want to play again (y/n)? ";

cin >> playAgain;

} while (playAgain == 'y' || playAgain == 'Y');

cout << "Thanks for playing Blackjack! Goodbye!" << endl;

return 0;

}