A Project Report

BSCY Card Picking Game Using Random Numbers 2025

on

Card Picking Game Using Random Numbers



by

Mohsin Rasul BCY243024

Rana Rameez Mushtaq BCY243023

A Project Report submitted to the

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

in partial fulfillment of the requirements for the degree of

BACHELORS OF SCIENCE IN CYBER SECURITY

Faculty of Engineering

Capital University of Science & Technology,

Islamabad

January, 2025

Copyright © 2025 by CUST Student

All rights reserved. Reproduction in whole or in part in any form requires the prior written permission of Mohsin Rasul BCY243024, Rana Rameez Mushtaq BCY243023 or designated representative.

# DECLARATION

It is declared that this is an original piece of our own work, except where otherwise acknowledged in text and references. This work has not been submitted in any form for another degree or diploma at any university or other institution for tertiary education and shall not be submitted by us in future for obtaining any degree from this or any other University or Institution.

Mohsin Rasul  
Reg. No. BCY243024

Rana Rameez Mushtaq

Reg. No. BCY243023

January, 2025

**Table of Contents**

[DECLARATION iii](#_Toc187755166)

[PROBLEM: v](#_Toc187755167)

[METHODOLOGIES USED: v](#_Toc187755168)

[CODE: v](#_Toc187755169)

[RESULTS: xiii](#_Toc187755170)

[REFERENCES: xiii](#_Toc187755171)

# PROBLEM:

The purpose of this project was to create a fun and interactive card game, Blackjack, using C++. The game is designed for either player vs. computer or player vs. player mode. The main challenge was to create an efficient game flow that accurately simulates drawing cards, tracks scores, identifies winners, and manages ties or busts. Another objective was to save game results in a properly formatted file for record-keeping.

# METHODOLOGIES USED:

1. **Deck Initialization**: A full deck of 52 cards was represented in the program, with suits, ranks, and values assigned to each card.

2. **Randomization:** The deck was shuffled at the beginning of the game to ensure randomness in the card-picking process.

3. **Game Flow Control:**

Players (or computer) took turns to pick cards and decide whether to draw or stop.

If a player’s score exceeded 21, they automatically lost the game.

4. **File Handling:** The game results were saved in a structured format, with details about each game and the players' outcomes (Win, Lose, Tie).

5. **Input Validation:** All user inputs were validated to prevent errors like selecting an already-used card or choosing a number outside the valid range.

# CODE:

#include <iostream>

#include <string>

#include <ctime>

#include <cstdlib>

#include <fstream>

using namespace std;

const int totalCards = 52;

const int maxHand = 11;

struct Card {

string suit;

string rank;

int value;

};

struct Player {

string name;

Card hand[maxHand];

int handCount = 0;

int totalPoints = 0;

bool stopped = false;

};

void initDeck(Card deck[]) {

string suits[] = { "Hearts", "Diamonds", "Clubs", "Spades" };

string ranks[] = { "2", "3", "4", "5", "6", "7", "8", "9", "10", "Jack", "Queen", "King", "Ace" };

int values[] = { 2, 3, 4, 5, 6, 7, 8, 9, 10, 10, 10, 10, 11 };

int index = 0;

for (int i = 0; i < 4; i++) {

for (int j = 0; j < 13; j++) {

deck[index] = { suits[i], ranks[j], values[j] };

index++;

}

}

}

void shuffleDeck(Card deck[]) {

srand(time(0));

for (int i = totalCards - 1; i > 0; i--) {

int randIndex = rand() % (i + 1);

Card temp = deck[i];

deck[i] = deck[randIndex];

deck[randIndex] = temp;

}

}

void displayCard(const Card& card) {

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

cout << "| " << card.rank << string(11 - card.rank.length(), ' ') << " |" << endl;

cout << "| |" << endl;

cout << "| |" << endl;

cout << "| " << card.suit[0] << " |" << endl;

cout << "| |" << endl;

cout << "| |" << endl;

cout << "| " << string(11 - card.rank.length(), ' ') << card.rank << " |" << endl;

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

}

void drawCard(Player& p, Card deck[], bool usedCards[], bool isComp = false) {

int cardIndex;

if (isComp) {

while (true) {

cardIndex = rand() % totalCards;

if (!usedCards[cardIndex]) {

break;

}

}

cout << "Computer picked a card." << endl;

} else {

while (true) {

cout << p.name << ", choose a card number (1-52): ";

cin >> cardIndex;

if (cardIndex < 1 || cardIndex > 52 || usedCards[cardIndex - 1]) {

cout << "Invalid choice or card already used. Try again!" << endl;

} else {

break;

}

}

cardIndex--;

}

usedCards[cardIndex] = true;

p.hand[p.handCount++] = deck[cardIndex];

p.totalPoints += deck[cardIndex].value;

if (p.totalPoints > 21) {

for (int i = 0; i < p.handCount; i++) {

if (p.hand[i].rank == "Ace" && p.hand[i].value == 11) {

p.hand[i].value = 1;

p.totalPoints -= 10;

break;

}

}

}

cout << p.name << " drew: " << endl;

displayCard(deck[cardIndex]);

cout << "Total points: " << p.totalPoints << endl;

}

void saveResult(string result) {

static int gameNum = 1;

ofstream outFile("game\_results.csv", ios::app);

if (outFile.is\_open()) {

outFile << "Game " << gameNum++ << endl;

outFile << result << endl;

outFile.close();

cout << "Result saved to game\_results.csv!" << endl;

} else {

cout << "Failed to save result!" << endl;

}

}

int main() {

Card deck[totalCards];

bool usedCards[totalCards] = { false };

Player p1, p2;

p1.name = "Player 1";

p2.name = "Player 2";

initDeck(deck);

shuffleDeck(deck);

cout << "Welcome to the Card Picking Game - Blackjack" << endl;

cout << "1 - Play with Computer" << endl;

cout << "2 - Play with Player" << endl;

cout << "3 - Exit" << endl;

int choice;

cin >> choice;

if (choice == 3) {

cout << "Goodbye!" << endl;

return 0;

}

while (true) {

p1.handCount = 0;

p2.handCount = 0;

p1.totalPoints = 0;

p2.totalPoints = 0;

p1.stopped = false;

p2.stopped = false;

cout << "Game started!" << endl;

while (!p1.stopped || !p2.stopped) {

if (!p1.stopped) {

drawCard(p1, deck, usedCards);

if (p1.totalPoints > 21) {

cout << p1.name << " busts! " << p2.name << " wins!" << endl;

saveResult(p1.name + " | Lose\n" + p2.name + " | Win");

break;

}

char choice;

cout << p1.name << ", do you want to draw another card? (y/n): ";

cin >> choice;

if (choice == 'n' || choice == 'N') {

p1.stopped = true;

}

}

if (!p2.stopped) {

if (choice == 1) {

drawCard(p2, deck, usedCards, true);

if (p2.totalPoints > 21) {

cout << p2.name << " busts! " << p1.name << " wins!" << endl;

saveResult(p1.name + " | Win\n" + p2.name + " | Lose");

break;

}

} else {

drawCard(p2, deck, usedCards);

if (p2.totalPoints > 21) {

cout << p2.name << " busts! " << p1.name << " wins!" << endl;

saveResult(p1.name + " | Win\n" + p2.name + " | Lose");

break;

}

char choice;

cout << p2.name << ", do you want to draw another card? (y/n): ";

cin >> choice;

if (choice == 'n' || choice == 'N') {

p2.stopped = true;

}

}

}

if (p1.stopped && p2.stopped) {

if (p1.totalPoints > p2.totalPoints) {

cout << p1.name << " wins!" << endl;

saveResult(p1.name + " | Win\n" + p2.name + " | Lose");

} else if (p1.totalPoints < p2.totalPoints) {

cout << p2.name << " wins!" << endl;

saveResult(p1.name + " | Lose\n" + p2.name + " | Win");

} else {

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

saveResult(p1.name + " | Tie\n" + p2.name + " | Tie");

}

break;

}

}

cout << "Play again?: ";

cin >> choice;

if (choice == 2) break;

}

cout << "Thank you for playing!" << endl;

return 0;

}

# RESULTS:

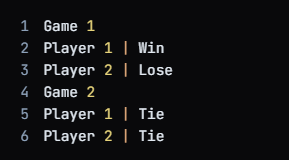
1. **Game Functionality:**

The game operates smoothly for both player vs. computer and player vs. player modes.

The game stops automatically when a player’s points exceed 21, declaring the other player as the winner.

2. **File Output Example:**

After the game, results are saved in a CSV file in the following format:



3. **User Experience:**

The game interface is interactive and provides clear prompts for the players.

The cards drawn by players are displayed visually, showing the suit and rank.

# REFERENCES:

1. C++ Documentation: For understanding arrays, structures, and file handling.

2. Online Tutorials: For learning randomization and implementing game logic.

3. Instructor and Group Members: For collaborative input and debugging.