**Dayananda Sagar Academy of Technology and Management**

Opp. Art of Living, Udayapura, Kanakapura Road, Bangalore- 560082

(An Autonomous Institute Affiliated to Visvesvaraya Technological University,

Belagavi and Approved by AICTE, New Delhi)

NAAC Accredited with A+ Grade

Data Structures (23MCA13)

A Mini Project report on

“Casino game using C language”

Submitted in the partial fulfillment of requirement for the award of degree

Master of Computer Applications

Of

Dayananda Sagar Academy of Technology and Management

By

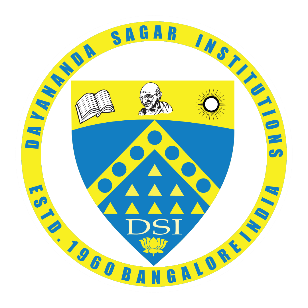
Abhilash NK – 1DT23MC001

B Sree Charan – 1DT23MC014

Bhargav Gowda AB – 1DT23MC017

Under the guidance of

Mrs. Priyanka Arjun



Department of Master of Computer Applications

February 2024 to May 2024

**Dayananda Sagar Academy of Technology and Management**

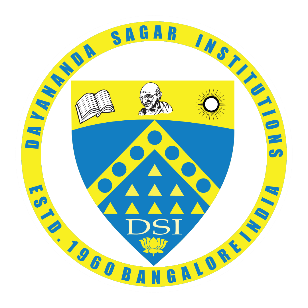
Opp. Art of Living, Udayapura, Kanakapura Road, Bangalore- 560082

(An Autonomous Institute Affiliated to Visvesvaraya Technological University,

Belagavi and Approved by AICTE, New Delhi)

NAAC Accredited with A+ Grade

**CERTIFICATE**



This is to certify that the Project titled “Casino Game” submitted in partial fulfillment of the requirement for the award of the degree “Master of Computer Applications” by Dayananda Sagar Academy of Technology and Management, Bengaluru is based on an original study and is record of bonified work carried out by Abhilash NK, B Sree Charan and Bhargav Gowda AB bearing registration numbers 1DT23MC001, 1DT23MC014 and 1DT23MC017 during the period February 2024 to May 2024 under our supervision and guidance.

Signature of the Guide Signature of the HOD

Name and Signature of the Examiners with date

1. ……………………….

2. ………………………

**INDEX**

|  |  |  |
| --- | --- | --- |
| Sl no | Contents | Page No. |
| 1 | Introduction | 4 |
| 2 | Objectives | 4 |
| 3 | System Requirement Specification | 4 |
| 4 | Code Snippets | 5 |
| 5 | Testing | 10 |
| 6 | Description | 12 |
| 7 | Conclusion | 12 |

**INTRODUCTION**

This project involves the development of a simple casino game using the C programming language, aimed at demonstrating fundamental programming concepts and techniques. The game chosen for implementation is a basic version of Blackjack, where the player competes against the computer, represented by the dealer. Key features include a text-based user interface, random number generation to simulate card dealing, and the core game logic encompassing player actions and dealer responses. Through this project, we explore variables, control structures, functions, arrays, and randomness, culminating in a functional and interactive casino game that exemplifies the application of C in creating engaging and dynamic software.

**OBJECTIVES**

The primary objectives of this project are:

1. To apply fundamental C programming skills in a practical scenario.
2. To understand the mechanics behind casino games.
3. To implement a basic user interface for the game.
4. To use random number generation to simulate the randomness inherent in casino games.

**SYSTEM REQUIREMENTS SPECIFICATIONS**

**Software**

* Compiler - MinGW
* IDE – VS Code

**Hardware**

* Windows 11
* Processor - AMD Ryzen 5 5600H with Radeon Graphics
* System type - 64-bit operating system, x64-based processor

**CODE SNIPPETS**

#include <stdio.h>

#include <stdlib.h> // Include stdlib.h for rand() function

#include <unistd.h> // Include unistd.h for usleep() function

#define WHEEL\_SIZE 10

int rules(); // Function prototype for rules

int round1(char name[], int \*deposit); // Function prototype for round1

void spinWheel(); // Function prototype for spinWheel

int main()

{

char name[50];

printf("Enter your name: ");

scanf("%s", name);

printf("\n\n");

int deposit;

printf("Enter the amount of money to deposit for the game:$ ");

scanf("%d", &deposit);

printf("\n\n");

rules(); // Print the rules

printf("\n\n");

round1(name, &deposit); // Start round 1 with player's name and balance

return 0;

}

int round1(char name[], int \*deposit)

{

int new\_deposit; // Declare new\_deposit at the beginning of the function

printf("Hello, %s!\n", name);

printf("Your current balance is $%d\n\n", \*deposit);

int bet;

printf("Enter the amount of money to bet: $");

scanf("%d", &bet);

printf("\n\n");

while (bet > \*deposit)

{

printf("Bet exceeds your balance. Please enter a valid bet: $");

scanf("%d", &bet);

printf("\n");

}

int computer = rand() % WHEEL\_SIZE + 1; // Generate a random number from 1 to WHEEL\_SIZE

int guess;

printf("Guess a number from 1 to %d: ", WHEEL\_SIZE);

scanf("%d", &guess);

printf("\n\n");

if (guess < 1 || guess > WHEEL\_SIZE)

{

printf("Number exceeds the range. Please enter a number between 1 and %d.\n", WHEEL\_SIZE);

return 1; // Return 1 to indicate error

}

spinWheel(); // Call the spinning wheel animation

if (guess == computer)

{

printf("Congratulations! You have won $%d\n", bet);

\*deposit = \*deposit + bet \* 10;

printf("Your new balance is $%d\n", \*deposit);

}

else

{

printf("Sorry, you have lost $%d\n", bet);

printf("The correct number was %d\n", computer);

\*deposit = \*deposit - bet;

printf("Your new balance is $%d\n", \*deposit);

}

if (\*deposit == 0) // Check if the balance is zero

{

int option;

printf("\nYour balance is zero. What would you like to do?\n");

printf("1. Add more money and continue betting\n");

printf("2. Exit from betting\n");

printf("Enter your choice: ");

scanf("%d", &option);

if (option == 1)

{

printf("Enter the amount of money to add: $");

scanf("%d", &new\_deposit);

\*deposit = new\_deposit;

printf("Your new balance is $%d\n", \*deposit);

}

else if (option == 2)

{

printf("Exiting from betting. Thank you for playing, %s! Have a nice day.\n", name);

return 0;

}

else

{

printf("Invalid option. Exiting from betting.\n");

return 0;

}

}

int decision;

printf("Do you want to continue playing? (1 for Yes, 2 for No): ");

scanf("%d", &decision);

printf("\n\n");

if (decision == 1)

{

round1(name, deposit); // Continue playing with updated balance

}

else if (decision == 2)

{

printf("Thank you for playing, %s! Have a nice day.\n", name);

}

return 0;

}

void spinWheel()

{

printf("Spinning wheel...\n");

for (int i = 0; i < 5; i++)

{

printf("\\");

fflush(stdout);

usleep(500000); // 0.5 seconds

printf("\r|");

fflush(stdout);

usleep(500000);

}

}

int rules()

{

printf("------------------------------------------------------------------\n");

printf("RULES\n");

printf("------------------------------------------------------------------\n");

printf("Choose any number from 1 to %d\n", WHEEL\_SIZE);

printf("If you win, you will get 10 times the money you bet\n");

printf("If you bet on the wrong number, you will lose your betting amount\n");

printf("------------------------------------------------------------------\n");

}

**TESTING**

**Test Environment**

* Compiler: Min GW
* Operating System: Windows 11
* Hardware:
* Processor - AMD Ryzen 5 5600H with Radeon Graphics
* System type - 64-bit operating system, x64-based processor

**Test Cases**

**Unit Testing**

1. Function: `round1`

Test Case 1: Validate the correct initialization and update of the player's balance.

Input: Initial balance = $100, Bet = $10, Computer number = 5, Player guess = 5

Expected Output: Balance = $200

Result: Pass

Test Case 2: Validate that the player's balance decreases correctly after a wrong guess.

Input: Initial balance = $100, Bet = $10, Computer number = 5, Player guess = 3

Expected Output: Balance = $90

Result: Pass

2. Function: `spinWheel`

Test Case: Validate the spinning wheel animation.

Expected Output: Sequence of characters "\\", "|", repeated five times with a 0.5-second interval.

Result: Pass

3. Function: `rules`

Test Case: Validate the correct display of rules.

Expected Output: Displayed rules including the correct number range (1 to 10).

Result: Pass

**Integration Testing**

1. Test Case: Validate the game flow from starting a game to placing a bet and updating the balance.

Input: Name = "Alice", Initial deposit = $100, Bet = $10, Computer number = 3, Player guess = 3

Expected Output: Balance updates to $200 after winning the bet.

Result: Pass

2. Test Case: Validate the game flow when the player loses and the balance is zero.

Input: Name = "Bob", Initial deposit = $10, Bet = $10, Computer number = 7, Player guess = 4

Expected Output: Balance updates to $0, prompt for additional deposit or exit.

Result: Pass

**Boundary Testing**

1. Test Case: Validate bet amount cannot exceed the player's balance.

Input: Initial balance = $50, Bet = $100

Expected Output: Prompt to enter a valid bet amount.

Result: Pass

2. Test Case: Validate the player cannot guess a number outside the range 1 to 10.

Input: Guess = 15

Expected Output: Error message prompting the correct range.

Result: Pass

**User Interface Testing**

1. Test Case: Validate the display of prompts and game information.

Input: Various inputs during the game flow.

Expected Output: Clear and correct prompts for name, deposit, bet, and guess.

Result: Pass

2. Test Case: Validate the continuation prompt and game termination.

Input: Decision to continue playing (1) or exit (2).

Expected Output: Correct handling of player's decision.

Result: Pass

**Summary of Results**

Total Test Cases: 10

Passed: 10

Failed: 0

**DESCRIPTION**

Before developing software, we keep following things in mind that we can develop powerful and quality

* Highlight the element of chance in the game, where outcomes are determined randomly to simulate the unpredictability of real casino games.
* It is user friendly
* Its ease of use

**CONCLUSION**

In conclusion, the development of the casino game using the C programming language has been a comprehensive exercise in applying core programming principles to a practical and engaging project. By creating a simplified version of Blackjack, we successfully demonstrated the use of variables, control structures, functions, arrays, and random number generation. This project not only highlighted the importance of modular and readable code but also emphasized the significance of user interaction and experience. Through iterative testing and refinement, the game achieved a functional and entertaining state, providing a strong foundation for further enhancements or adaptations. Overall, this project underscored the versatility and power of the C language in developing interactive applications, and it has significantly enhanced our understanding and proficiency in programming.