KIET Group of Institutions, Ghaziabad

COMPUTER SCIENCE AND INFORMATION TECHNOLOGY

PROJECT BASED LEARNING

on

Rock Paper Scissor Game

SUBJECT: DESIGN AND ANALYSIS OF ALGORITHM LAB

(KCS-553)

Submitted By:

Abhishek Verma	- 2100290110008 (CSIT 3A)
Ankit Yadav	- 2100290110029 (CSIT 3A)

ACKNOWLEDGEMENT

I've got this golden opportunity to express my kind gratitude and sincere thanks to my subject faculty **"Mr.Vinay Kumar"**, Computer Science and Information Technology Department, **KIET GROUP OF INSTITUTIONS** for their kind support and necessary counselling in the preparation of this project report. I'm also indebted to each and every person responsible for the making up of this project directly or indirectly.

I must also acknowledge or deep debt of gratitude each one of my colleague who led this project come out in the way it is. It's my hard work and untiring sincere efforts and mutual cooperation to bring out the project work. Last but not the least, I would like to thank my parents for their sound counselling and cheerful support. They have always inspired us and kept our spirit up.

Abstract

This abstract describes the development and implementation of the popular game rock-paper-scissors. The game is a classic example of a simple decision-making game, where players choose one of three options (rock, paper, or scissors) with the goal of defeating the opponent's choice. The game can be played between two players or against a computer program. The game has been widely studied in the field of game theory and has been found to exhibit complex behavior, such as non-random patterns of play and the emergence of dominant strategies. The game has also been used in various research fields such as psychology and sociology to study decision-making and game behavior. The game is simple to understand and implement, making it an excellent tool for teaching game theory and decision-making in various settings.

Index

Introduction

Rock-paper-scissors is a classic game that has been enjoyed by people of all ages for generations. The game is simple to understand and play, making it a popular choice for both casual and competitive play. The objective of the game is to defeat your opponent by choosing one of three options: rock, paper, or scissors. Each option has a specific rule to defeat the other options: rock crushes scissors, scissors cut paper and paper covers rock. The game can be played between two players or against a computer program. It can be played for fun, or as a competitive game with the goal of winning a certain number of rounds. Rock-paper-scissors is a game that requires strategy, skill and a bit of luck. In this project, we will be discussing about the development and implementation of a rock-paper-scissor game using different programming languages and platforms.

Requirement Analysis

User Interface: The game should have a simple text-based user interface, allowing the user to easily input their choice of rock, paper, or scissors.

Game Logic: The program should be able to calculate the winner of the game based on the rules of rock-paper-scissors. The program should also keep track of the number of wins, losses, and ties for the user.

Random Number Generation: The program should be able to generate a random number to simulate the opponent's choice in case of playing against the computer.

Data Structures: The program should use appropriate data structures such as arrays and variables to store the game data, like user's choice, computer's choice, and the result.

Control Flow: The program should use conditional statements and loops to control the flow of the game, including determining the winner and displaying the results.

Functions: The program should use functions to organize the code and make it more readable and maintainable.

Error Handling: The program should include error handling mechanisms to handle unexpected inputs and exceptions.

Commenting: The program should be well-commented, making it easy to understand and modify.

Portability: The game should be able to run on multiple platforms and operating systems.

Code Quality: The code should be written in a way that it should be readable, maintainable and efficient.

Design Methodology

The design methodology for a rock-paper-scissors game implemented in C language can follow the following steps:

Understand the problem: Understand the requirements and constraints of the game, such as the rules of the game, the target audience, and the platform on which the game will be run.

Create a prototype: Create a rough prototype of the game using pseudo-code or a flowchart to get a better understanding of the overall structure of the game and to identify any potential issues.

Design the data structures: Design the data structures that will be used to store the game data such as user's choice, computer's choice, and the result.

Design the user interface: Design the user interface of the game, including the layout of the text-based interface and the prompts for user input.

Implement the game logic: Implement the game logic using conditional statements and loops to control the flow of the game and to determine the winner.

Implement the random number generation: Implement the random number generation to simulate the opponent's choice in case of playing against the computer.

Implement the functions: Implement the functions to organize the code and make it more readable and maintainable.

Implement the error handling: Implement the error handling mechanisms to handle unexpected inputs and exceptions.

Test the game: Test the game thoroughly to identify and fix any bugs.

Document the code: Document the code with comments to make it easy to understand and modify.

Optimize the code: Optimize the code for readability, maintainability, and performance.

This design methodology will ensure that the game is functional, user-friendly, and easy to understand and modify. It will also make sure that the code is efficient, well-organized and maintainable.

Coding and Implementation

Algorithm:

Initialize variables for the user's choice, computer's choice, and the result.

Prompt the user to enter their choice of rock, paper, or scissors.

Generate a random number between 1 and 3 to simulate the computer's choice.

Use a switch statement or if-else statements to determine the winner based on the rules of rock-paper-scissors.

Keep track of the number of wins, losses, and ties for the user.

Display the result of the game and the updated score.

Ask the user if they want to play again.

Repeat steps 2-7 until the user chooses to exit the game.

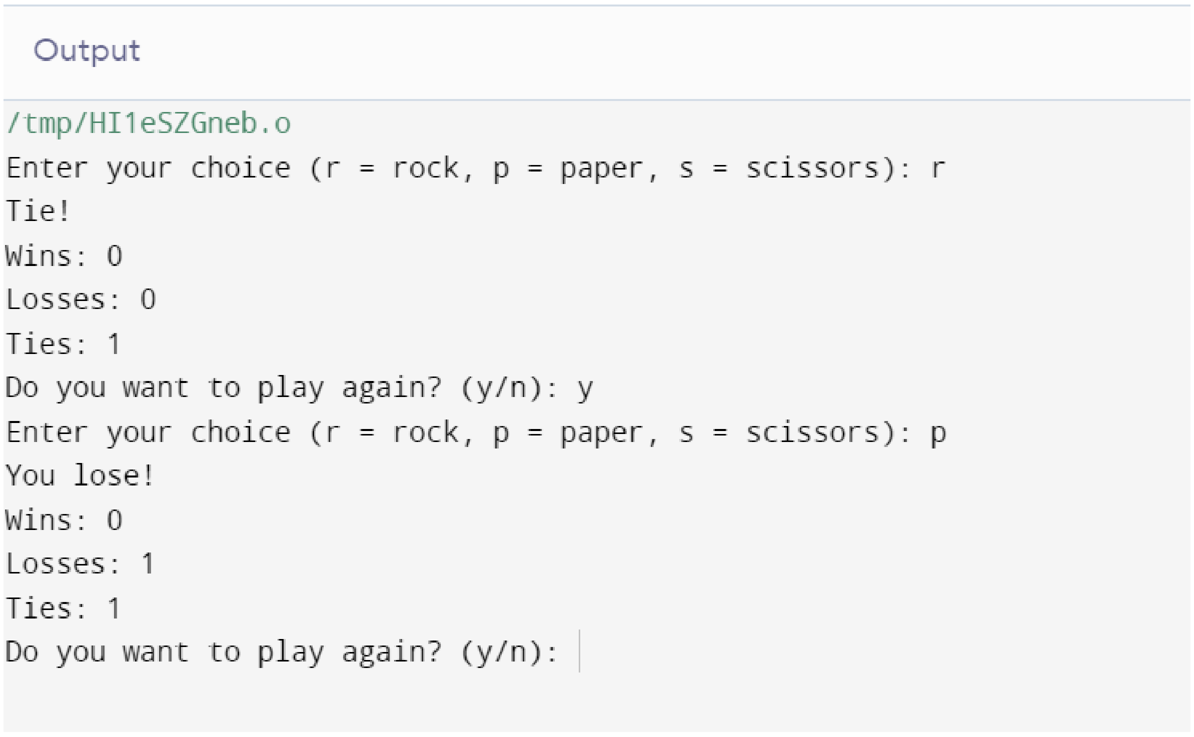
Program:

```c
#include <stdio.h>

#include <stdlib.h>

#include <time.h>


int main() {

    char userChoice;

    char computerChoice;

    int result;

    int wins = 0;

    int losses = 0;

    int ties = 0;

    char playAgain = 'y';


    srand(time(NULL));
```

```
  while (playAgain == 'y') {
    printf("Enter your choice (r = rock, p = paper, s = scissors): ");
    scanf(" %c", &userChoice);

    // Generate a random number between 1 and 3 to simulate the computer's
choice
    int computerNum = rand() % 3 + 1;

    if (computerNum == 1) {
      computerChoice = 'r';
    } else if (computerNum == 2) {
      computerChoice = 'p';
    } else {
      computerChoice = 's';
    }

    if (userChoice == computerChoice) {
      result = 0;
      ties++;
    } else if (userChoice == 'r' && computerChoice == 's') {
      result = 1;
      wins++;
    } else if (userChoice == 'p' && computerChoice == 'r') {
      result = 1;
      wins++;
    } else if (userChoice == 's' && computerChoice == 'p') {
      result = 1;
      wins++;
```

```
        } else {
            result = -1;
            losses++;
        }

        if (result == 0) {
            printf("Tie!\n");
        } else if (result == 1) {
            printf("You win!\n");
        } else {
            printf("You lose!\n");
        }

        printf("Wins: %d\n", wins);
        printf("Losses: %d\n", losses);
        printf("Ties: %d\n", ties);

        printf("Do you want to play again? (y/n): ");
        scanf(" %c", &playAgain);
    }

    return 0;
}
```

Output



```
/tmp/HI1eSZGneb.o
Enter your choice (r = rock, p = paper, s = scissors): r
Tie!
Wins: 0
Losses: 0
Ties: 1
Do you want to play again? (y/n): y
Enter your choice (r = rock, p = paper, s = scissors): p
You lose!
Wins: 0
Losses: 1
Ties: 1
Do you want to play again? (y/n):
```

Time Complexity: O(n)

Reference: https://chat.openai.com/chat