BAHRIA UNIVERSITY KARACHI CAMPUS

DEPARTMENT OF COMPUTER SCIENCE

Bahria University
Discovering Knowledge

COMPUTER PRORAMMING (CSL-113)

BS(IT)-1(A)

ROCK, PAPER SCISSORS GAME

Submitted by:

ESHA ASHFAQ (02-235232-002)

FAIZA ALTAF (02-235232-036)

HADIQA MEHMOOD (02-235232-007)

Submitted To:

Engr. Gul Saba

Submitted On:

Date: 10-JAN-2024

Table of Contents

ACKNOWLEDEMENT:

The Rock, Paper, Scissors C++ code provided here showcases a well-structured and functional implementation of the classic game. We have effectively utilized standard input/output and random number generation to create an engaging user experience. The use of switch statements for displaying choices and determining the winner adds clarity to the code. Additionally, the incorporation of a do-while loop allows the game to be replayed based on user preference. Overall, this code serves as a solid example for beginners learning C++ programming and game development.

ABSTARCT:

The C++ code provided offers a concise and functional implementation of the classic Rock, Paper, Scissors game, demonstrating key principles of programming such as user input, random number generation, and control flow structures. The code begins by seeding the random number generator with the current time, ensuring a more unpredictable distribution of values for the computer's choices. The user is prompted to input their selection (Rock, Paper, or Scissors) represented by numerical values 1, 2, or 3, respectively. The computer's choice is generated randomly using the modulo operator to limit the range of possible outcomes.

The program then displays the user and computer choices, providing clear and user-friendly output through switch statements. The comparison of choices follows, determining the winner based on the rules of Rock, Paper, Scissors. A tie is declared if both choices are identical, otherwise, the winner is determined by specific combinations of choices. The code gracefully handles invalid user inputs, informing the user of their mistake and prompting them to enter a valid choice.

The implementation incorporates a do-while loop to enable repeated gameplay based on the user's preference. After each round, the user is prompted to decide whether they want to play again by entering 'y' for yes or 'n' for no. The loop continues as long as the user opts for another round, enhancing the overall user engagement.

This code serves as an excellent learning resource for beginners exploring C++ programming and game development. It demonstrates fundamental concepts such as conditional statements, loops, and random number generation in a context that is both practical and enjoyable. The clear

organization and concise syntax contribute to the code's readability and understanding. Overall, this Rock, Paper, Scissors implementation stands as a valuable example for educational purposes, illustrating key programming techniques in a fun and interactive manner.

INTRODUCTION:

Rock, Paper, Scissors is a classic hand game that has transcended generations, cultures, and borders. Originating in China over two thousand years ago, this simple yet engaging game has found its way into various aspects of popular culture. In its essence, Rock, Paper, Scissors is a game of chance and strategy, where three elements – rock, paper, and scissors – interact in a cyclic manner, determining the winner of each round based on their unique relationships.

This report explores the implementation of the Rock, Paper, Scissors game using the C++ programming language. As we delve into the details of our C++ code, we will uncover the logic and structure required to create a functional and enjoyable rendition of this timeless game. From user input to randomization and outcome determination, our C++ implementation aims to capture the essence of Rock, Paper, Scissors while providing insights into programming fundamentals. Whether you are a novice programmer seeking to enhance your skills or simply a fan of the game, this report serves as a comprehensive guide to understanding the intricacies of coding Rock, Paper.

PROBLEM STATEMENT:

Develop a C++ program that implements the classic Rock, Paper, Scissors game. The program should prompt the user to input their choice (Rock, Paper, or Scissors) and generate a random choice for the computer. Display both choices, determine the winner based on the game rules, and inform the user of the outcome. Allow the user to play multiple rounds by providing an option to continue playing after each round. The program should handle invalid inputs gracefully and provide a user-friendly interface throughout the gaming experience

EXISTING SYSTEM:

Rock, Paper, Scissors games are a popular and simple form of entertainment that has been implemented in various programming languages. While the basic structure of the game remains consistent across implementations, differences arise in terms of language-specific syntax, user

interface design, and additional features. Here, we compare the provided C++ implementation with potential existing systems:

Graphical User Interface (GUI) Implementations: Some versions of Rock, Paper, Scissors are built with graphical interfaces, introducing visual elements to enhance user experience. Such implementations may use buttons or interactive elements to represent choices, providing a more visually appealing interaction compared to a console-based approach.

Web-Based Implementations: Online versions of Rock, Paper, Scissors exist as web applications. These implementations often include additional features such as multiplayer options, score tracking, and animations. The web environment allows for a more dynamic and interactive experience compared to a console-based application.

Mobile Applications: Rock, Paper, Scissors games are available as mobile applications on various platforms. These versions leverage touch interfaces and may include features like sound effects, customizable avatars, and social sharing options, distinguishing them from console-based or web implementations.

The provided C++ implementation distinguishes itself by offering a basic yet functional representation of the game, suitable for educational purposes. It adheres to fundamental programming principles, providing a clear and concise example for beginners. While lacking the graphical and interactive features of more advanced implementations, it serves as a solid foundation for learning key programming concepts in the context of a classic game.

OBJECTIVES AND GOALS:

The primary objective is to introduce fundamental concepts of C++ programming, including user input, random number generation, conditional statements, and loop structures. By prompting the user to input their choice and generating a random choice for the computer, the code illustrates the basics of interactive program design.

Another goal is to demonstrate the use of control flow structures such as switch statements for efficient handling of multiple cases. The clear organization of the code enhances readability and helps learners understand how to structure decision-making processes in a program.

The incorporation of a do-while loop facilitates a repeated gaming experience, allowing users to play multiple rounds based on their preference. This feature reinforces the understanding of loop structures and user-driven program flow.

Overall, the code seeks to offer a foundational example that is both instructive and enjoyable, allowing beginners to grasp key programming concepts within the context of a classic and familiar game. By achieving these objectives, the code provides a stepping stone for learners to build upon as they progress in their understanding of C++ programming.

PROJECT SCOPE:

The scope of this project involves implementing a Rock, Paper, Scissors game using the C++ programming language. The primary objectives include:

User Interaction: Create a user-friendly interface for players to input their choices (rock, paper, or scissors) within the console application.

Randomization: Implement a mechanism to randomly generate the computer's choice to ensure an element of chance in the game.

Game Logic: Develop the underlying logic to determine the winner of each round based on the interactions between the user's and computer's choices.

Score Tracking: Implement a scoring system to keep track of the user's and computer's wins, losses, and ties throughout the course of multiple rounds.

Replay ability: Allow players the option to replay the game and experience the thrill of Rock, Paper, Scissors repeatedly.

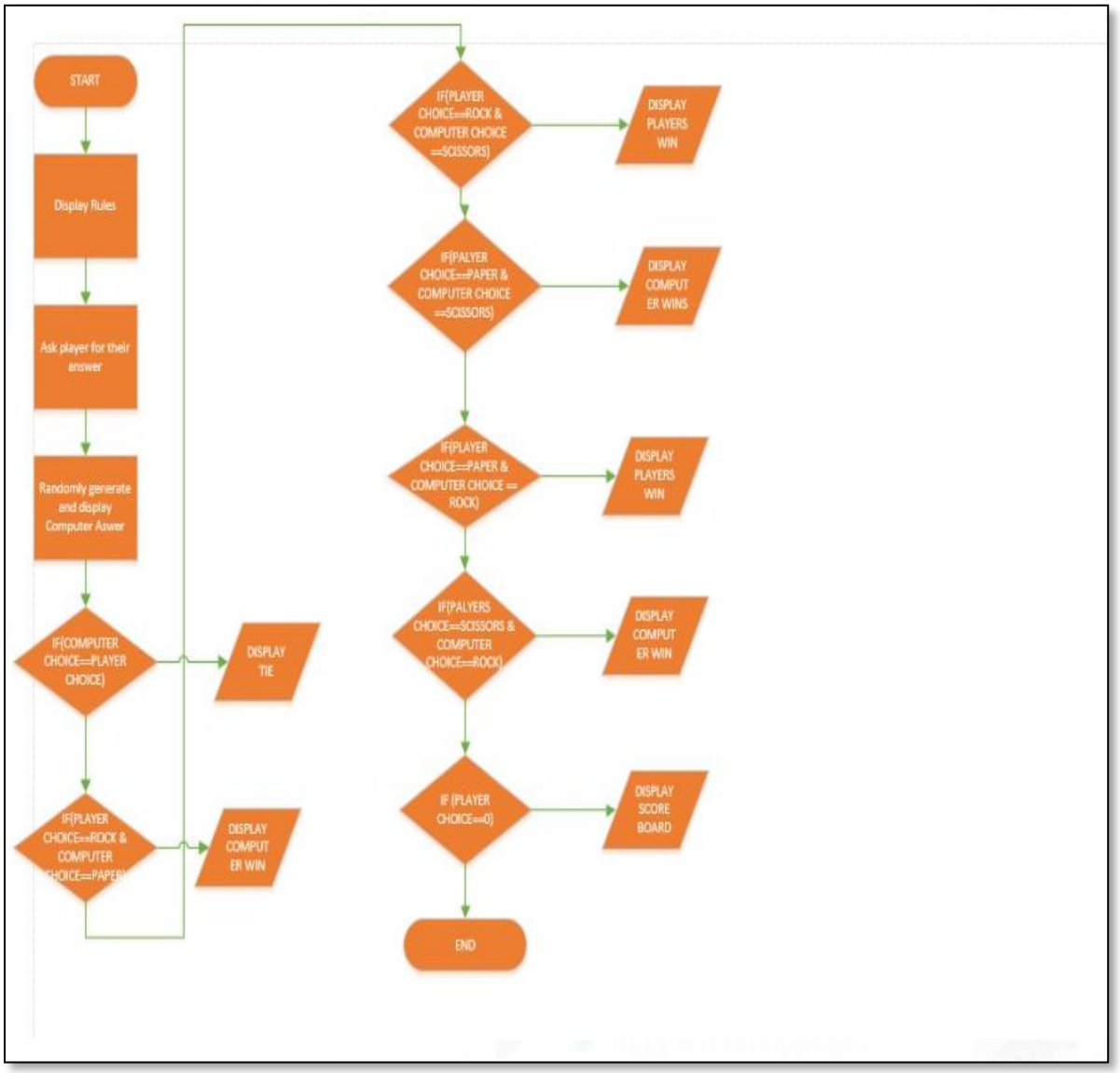
Code Structure: Maintain a well-organized and modular code structure, adhering to best practices in C++ programming for readability and maintainability.

Documentation: Provide comprehensive documentation, including comments within the code, to enhance understanding for both developers and users.

Error Handling: Implement error-handling mechanisms to handle unexpected inputs or scenarios gracefully, ensuring a robust user experience.

The project aims to serve as an educational tool for individuals interested in learning C++ programming, offering a practical application of fundamental concepts such as user input, randomization, conditional statements, loops, and modular code design. Additionally, it provides an opportunity to explore basic game development principles within the context of a simple yet engaging game.

WORKFLOW:



OVERVIEW:

This C++ Rock Paper Scissors game lets you test your luck and strategy against a computer opponent. Each round, you choose Rock, Paper, or Scissors, and the computer throws out its own random pick. The program then reveals your choices, declares the winner based on the classic rules, and lets you play again or quit. It's a simple but fun way to practice basic programming concepts like user input, conditional statements, and random number generation.

The program clearly displays chosen gestures, announces round results, and provides prompts for additional rounds. Users can play as many rounds as they want, making it a great way to practice variables, user input/output, conditional statements, random number generation, and looping structures.

TOOLS AND TECHNOLOGIES:

The tools and technologies used in this project are given below:

Programming Language:

The programming language that we have used is C++.

Libraries:

<iostream>: For input/output operations (displaying prompts, receiving user input, and printing game messages).

<cstdlib>: Provides the rand() function for generating random numbers (used for the computer's choices).

<ctime>: Includes the time() function for seeding the random number generator (ensuring different sequences of random choices).

Development Environment: Any C++ compiler or IDE (e.g., Visual Studio Code, DEV C etc.)

PROJECT FATURES:

The main functional and non-functional requirements are given below:

Functional Requirements:

- User can choose Rock, Paper, or Scissors.
- Computer randomly selects Rock, Paper, or Scissors.
- Game logic determines the winner based on standard game rules.
- Results of each round (user's choice, computer's choice, winner) are displayed.
- User has the option to play again or quit.

Non-Functional Requirements:

- Simple and intuitive text-based interface.
- Clear and concise output messages.
- Efficient code execution.

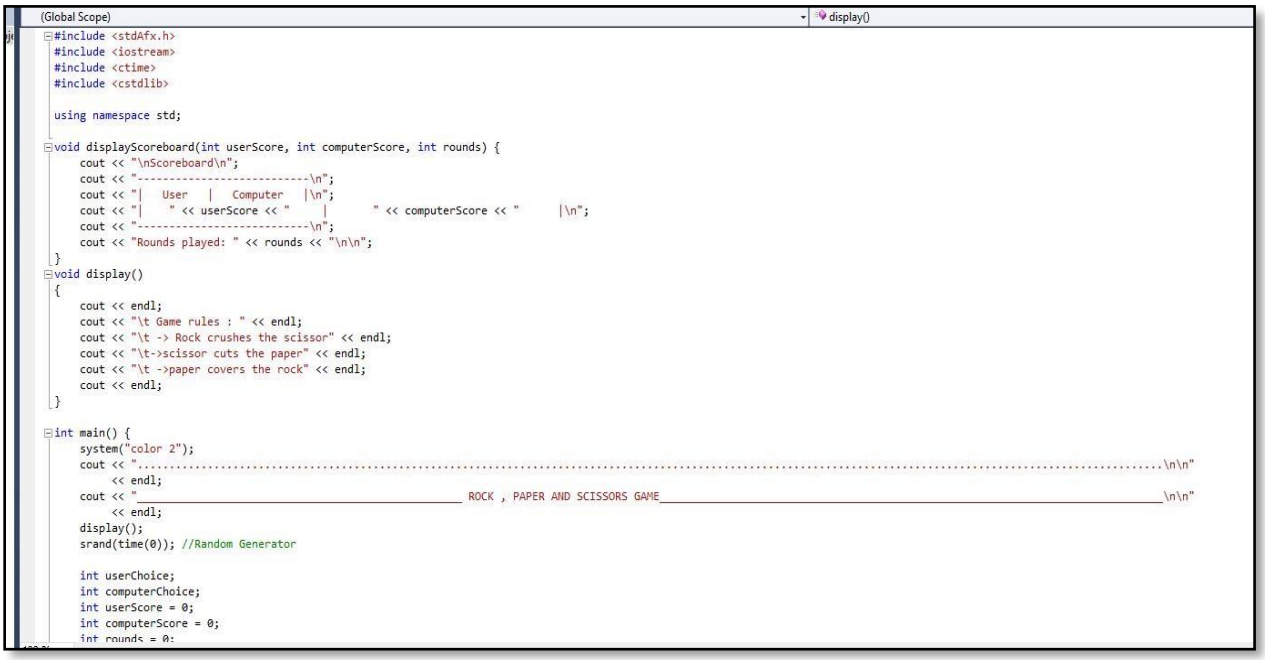
IMPLEMENTED CONCEPTS:

Following are the concepts that are used in this project:

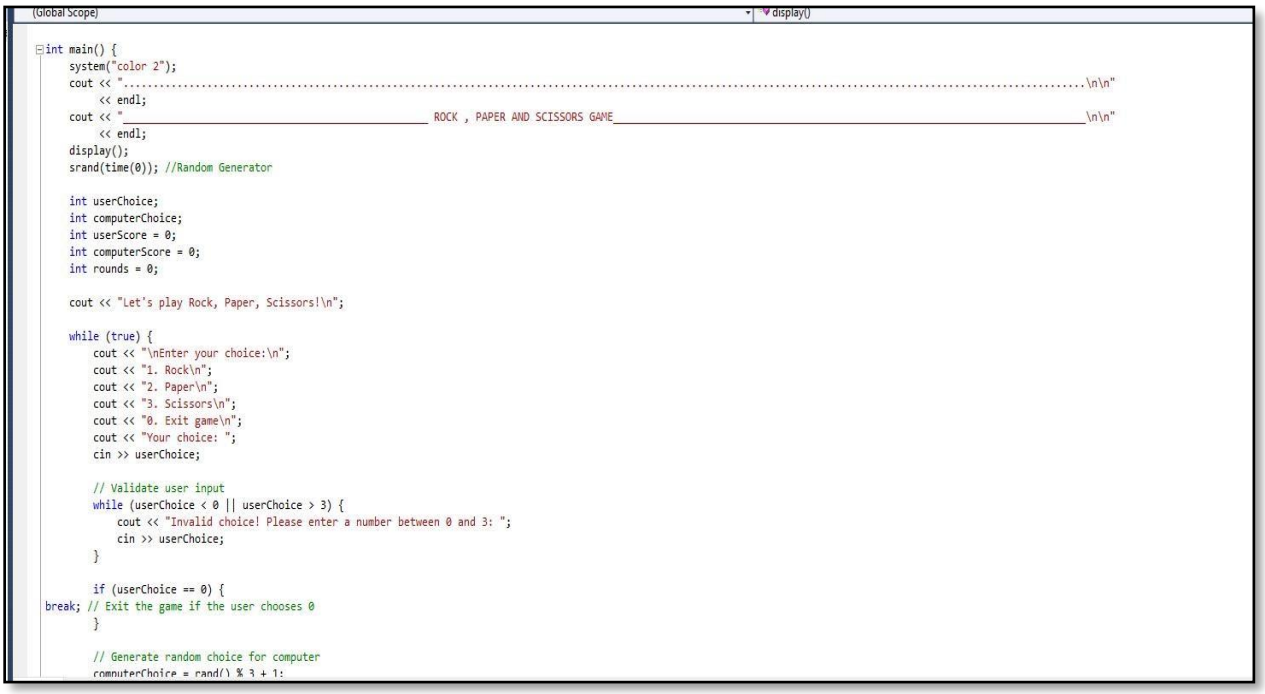
- **Variables:** Used to store user choices, computer choices, game state, and potential scores.
- **Data types:** Appropriate data types (e.g., integers, characters) are used for storing and manipulating game data.
- **User input and output**: The cin and cout objects are used to interact with the user, receiving input and displaying output.
- **Conditional statements:** if statements and their variants are employed to implement game logic, determine winners, and manage game flow.
- **Random number:** The rand() function is utilized to generate the computer's random **choices.**
- **Loop structures:** A do-while loop is used to repeat the game process until the user chooses to quit.

CODE:

The code of this project is:



```
(Global Scope)                                                    display()
#include <stdAfx.h>
#include <iostream>
#include <ctime>
#include <cstdlib>

using namespace std;

void displayScoreboard(int userScore, int computerScore, int rounds) {
    cout << "\nScoreboard\n";
    cout << "--------------------------\n";
    cout << "|   User   |   Computer   |\n";
    cout << "|    " << userScore << "     |       " << computerScore << "      |\n";
    cout << "--------------------------\n";
    cout << "Rounds played: " << rounds << "\n\n";
}
void display()
{
    cout << endl;
    cout << "\t Game rules : " << endl;
    cout << "\t -> Rock crushes the scissor" << endl;
    cout << "\t->scissor cuts the paper" << endl;
    cout << "\t ->paper covers the rock" << endl;
    cout << endl;
}

int main() {
    system("color 2");
    cout << "..........\n\n"
         << endl;
    cout << "_____ ROCK , PAPER AND SCISSORS GAME_____\n\n"
         << endl;
    display();
    srand(time(0)); //Random Generator

    int userChoice;
    int computerChoice;
    int userScore = 0;
    int computerScore = 0;
    int rounds = 0;
```



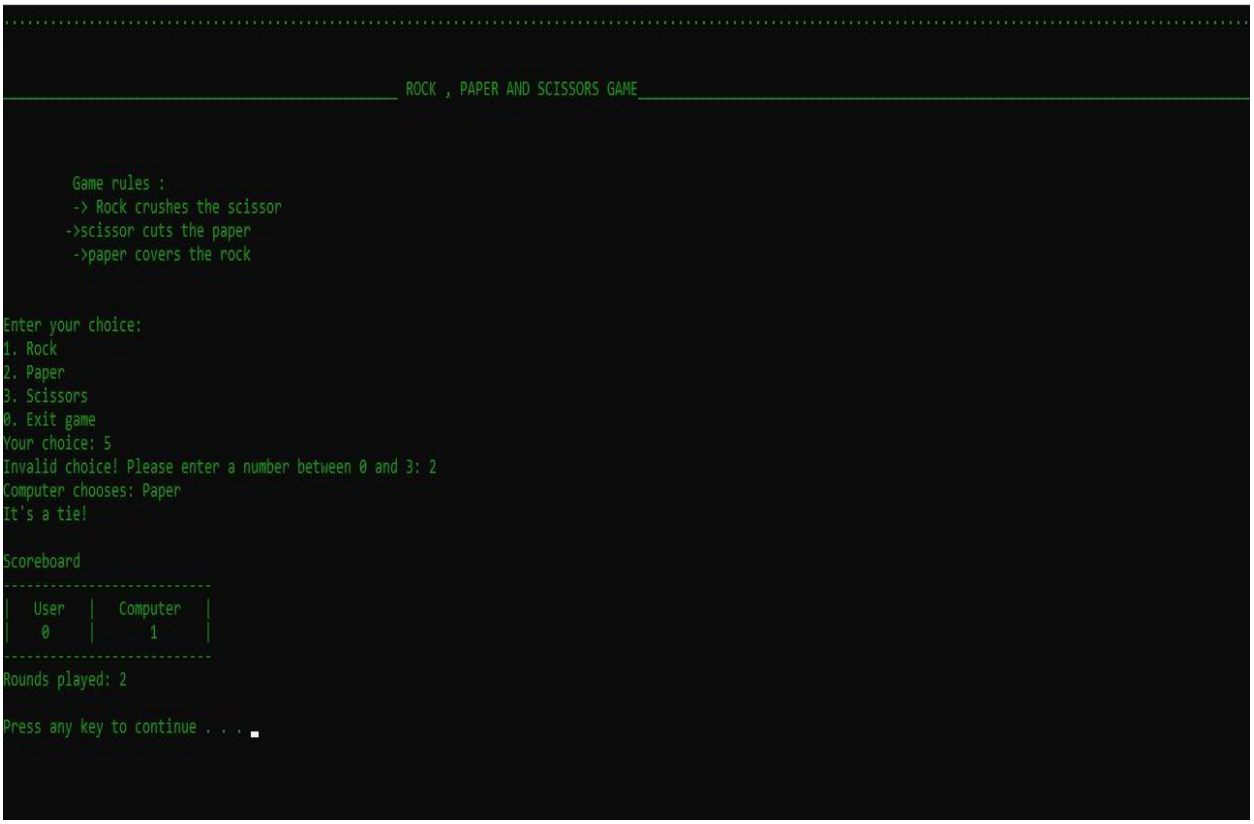
```
(Global Scope)                                                    display()
int main() {
    system("color 2");
    cout << "..........\n\n"
         << endl;
    cout << "_____ ROCK , PAPER AND SCISSORS GAME_____\n\n"
         << endl;
    display();
    srand(time(0)); //Random Generator

    int userChoice;
    int computerChoice;
    int userScore = 0;
    int computerScore = 0;
    int rounds = 0;

    cout << "Let's play Rock, Paper, Scissors!\n";

    while (true) {
        cout << "\nEnter your choice:\n";
        cout << "1. Rock\n";
        cout << "2. Paper\n";
        cout << "3. Scissors\n";
        cout << "0. Exit game\n";
        cout << "Your choice: ";
        cin >> userChoice;

        // Validate user input
        while (userChoice < 0 || userChoice > 3) {
            cout << "Invalid choice! Please enter a number between 0 and 3: ";
            cin >> userChoice;
        }

        if (userChoice == 0) {
break; // Exit the game if the user chooses 0
        }

        // Generate random choice for computer
        computerChoice = rand() % 3 + 1;
```

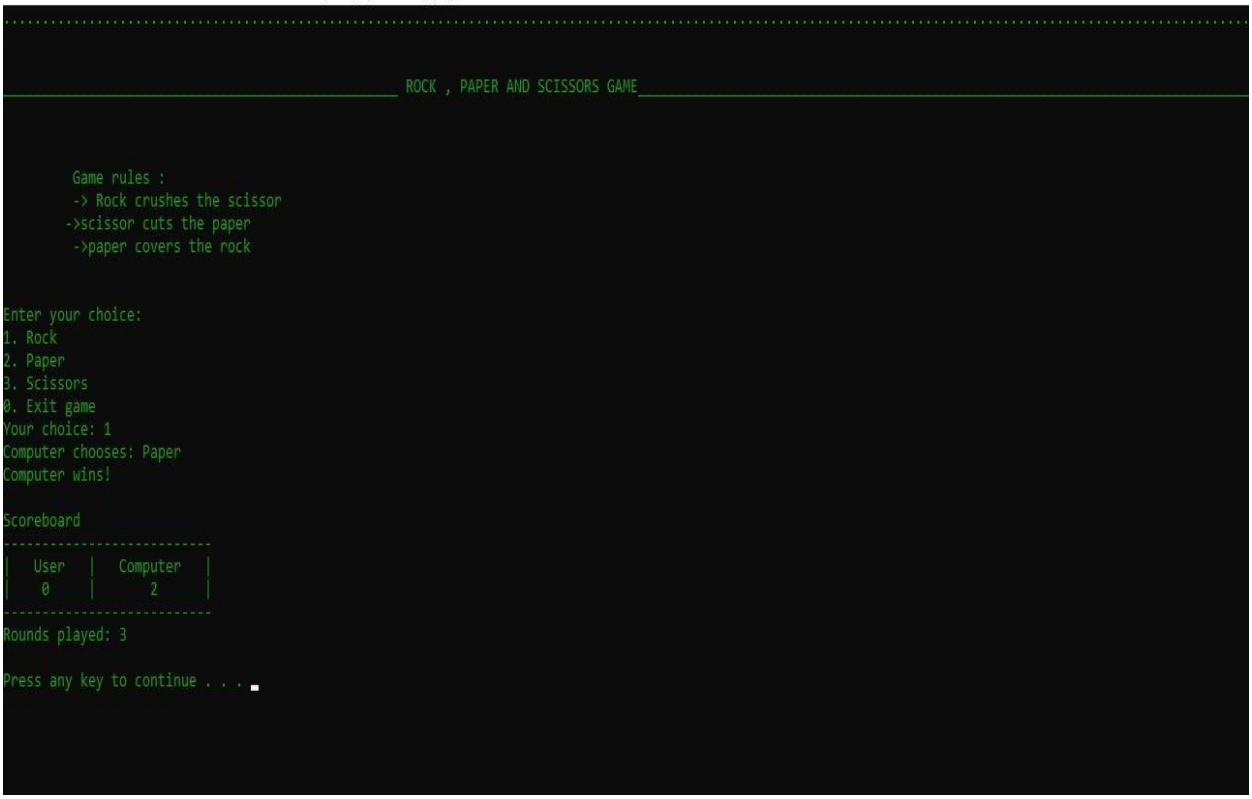
```
(Global Scope)                                                                                                    display()

        // Generate random choice for computer
        computerChoice = rand() % 3 + 1;

        cout << "Computer chooses: ";
        switch (computerChoice) {
            case 1:
                cout << "Rock\n";
                break;
            case 2:
                cout << "Paper\n";
                break;
            case 3:
                cout << "Scissors\n";
                break;
        }

        // Determine the winner
        if (userChoice == computerChoice) {
            cout << "It's a tie!\n";
        } else if ((userChoice == 1 && computerChoice == 3) ||
                   (userChoice == 2 && computerChoice == 1) ||
                   (userChoice == 3 && computerChoice == 2)) {
            cout << "You win!\n";
            userScore++;
        } else {
            cout << "Computer wins!\n";
            computerScore++;
        }

        rounds++;

        // Display current scoreboard
        displayScoreboard(userScore, computerScore, rounds);
        system("pause");
        system("cls");
        cout << "..........\n\n"
         << endl;
    cout << "                                                ROCK , PAPER AND SCISSORS GAME                                                \n\n"
00 %
```

```
        // Display current scoreboard
        displayScoreboard(userScore, computerScore, rounds);
        system("pause");
        system("cls");
        cout << "..........\n\n"
         << endl;
    cout << "_____ ROCK , PAPER AND SCISSORS GAME_____\n\n"
         << endl;
    display();

    }

    // Display final scoreboard after exiting the game
    displayScoreboard(userScore, computerScore, rounds);
    if (userScore>computerScore)
    {
        cout<<"User Won!!!!!!\n";
    }
    else if(userScore==computerScore)
    {
        cout<<"Its a Tie\n";
    }
    else
    {
        cout<<"Computer Won!!!!!\n";
        cout<<"You loss!!!!!!\n";
    }
    cout << "Thanks for playing!\n";
    system("pause");

return 0;
}
```

OUTPUT:

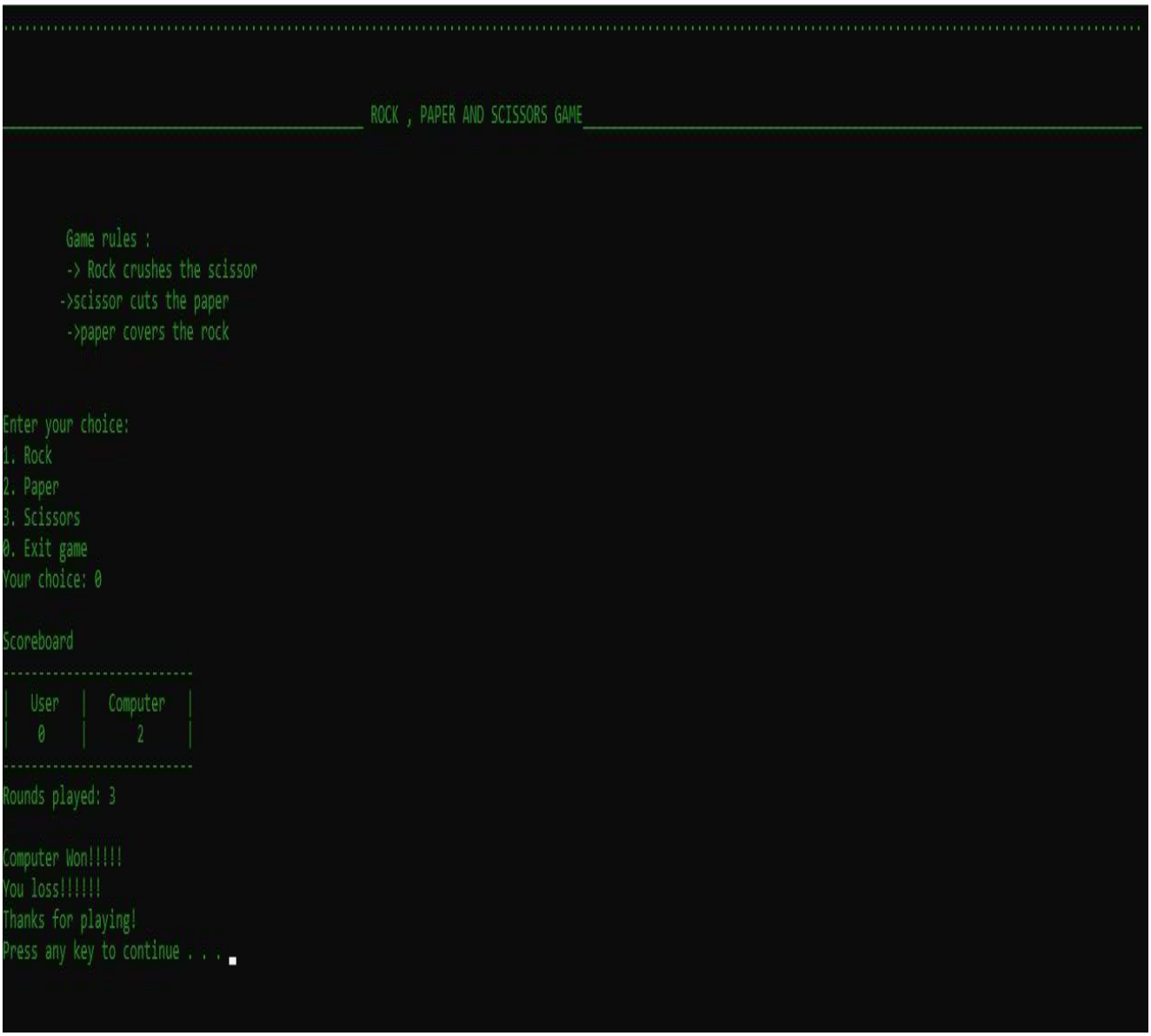
The output of this code is:


ROCK , PAPER AND SCISSORS GAME
Game rules :
-> Rock crushes the scissor
->scissor cuts the paper
->paper covers the rock
Enter your choice:
1. Rock
2. Paper
3. Scissors
0. Exit game
Your choice: 5
Invalid choice! Please enter a number between 0 and 3: 2
Computer chooses: Paper
It's a tie!
Scoreboard
| User | Computer |
| 0 | 1 |
Rounds played: 2
Press any key to continue . . .


ROCK , PAPER AND SCISSORS GAME
Game rules :
-> Rock crushes the scissor
->scissor cuts the paper
->paper covers the rock
Enter your choice:
1. Rock
2. Paper
3. Scissors
0. Exit game
Your choice: 1
Computer chooses: Paper
Computer wins!
Scoreboard
| User | Computer |
| 0 | 2 |
Rounds played: 3
Press any key to continue . . .

After exiting:

FINAL SCOREBOARD:



FUTURE WORK:

This Rock Paper Scissors game provides a solid foundation for further development and expansion. Here are some potential areas for future work:

Two-player mode: Implement functionality for two human players to compete against each other, adding a social element to the game.

AI-powered opponent: Introduce AI strategies for the computer opponent, progressively increasing its difficulty based on the player's performance.

Game variations: Include variations like "Rock Paper Scissors Lizard Spock" or other customized hand gestures to offer more gameplay options.

Graphical user interface (GUI): Develop a graphical interface with visual representations of hand gestures, scoreboards, and animated outcomes for a more engaging experience.

Sound effects and music: Integrate sound effects for chosen gestures and background music to enhance the atmosphere and user enjoyment.

Network play: Implement online or LAN multiplayer functionality to allow players to compete remotely with friends or other users.

Scorekeeping and statistics: Track individual wins, losses, and ties for both players and the computer, potentially showcasing leaderboards or rankings.

Customization options: Offer options to personalize the game, such as choosing different background themes, sound effects, or difficulty levels.

CONCLUSION:

This C++ Rock Paper Scissors game successfully presents a fun and interactive way to apply fundamental programming concepts learned in computer programming. It emphasizes user input, conditional statements, random number generation, and looping structures, providing a valuable practice platform for beginners. Beyond its educational value, the game offers engaging gameplay for anyone looking to enjoy a classic challenge. The project's potential for future work is vast,

offering opportunities to expand functionalities, enhance user experience, and integrate advanced programming techniques. By continuously improving and building upon this foundation, the Rock Paper Scissors game can evolve into a more engaging and versatile application, demonstrating the practical application of coding skills and promoting a love for programming.

REFERENCES:

- AN INTRODUCTION TO PROGRAMMING WITH C++……. **DIANE ZAK**
- STARTING OUT WITH >>> C++ FROM CONTROL STRUCTURES THROUGH OBJECTS……. **TONY GADDIS (EIGHTH EDITION)**
- PROGRAMMING WITH C++ ……AIKMAN SERIES.