

Yunnan University Software College

C++ Course

**School of Software, Yunnan University**

**Name: ANANDA RABIDAS MUGDHO**

**ID: 20233120010**

## ****Table of Contents****

* 1.Introduction
* 2.Game Design
* - Core Components
* - Key Features
* 3.Game Rules
* - Objective
* - gameplay Mechanics
* - Scoring System
* 4.Implementation details
* - Random Number Generation
* - Input Validation
* - Game Loop Logic
* - Statistics Tracking
* 5.Technical Considerations
* - Strengths
* - Limitations
* - Potential Improvements
* 6.Conclusion

## ****1. Introduction****

The **Number Guessing Game** is a simple yet engaging console-based game implemented in C++. The program generates a random number between 1 and 100, and the player must guess the correct number within the fewest attempts possible. The game provides feedback (too high/too low) and adjusts the guessing range dynamically. It also tracks player statistics across multiple rounds.

This report provides a **structured breakdown** of the game’s:

* **Design philosophy**
* **Rules and mechanics**
* **Technical implementation**
* **Strengths, limitations, and future improvements**

## ****2. Game Design****

### ****2.1 Core Components****

| **Component** | **Description** |
| --- | --- |
| **Random Number Generator** | Uses rand() seeded with system time (srand(time(0))). |
| **User Interface (UI)** | Text-based console with prompts and feedback. |
| **Game Logic** | Validates input, adjusts ranges, and checks guesses. |
| **Statistics Tracker** | Records games played, wins, and attempts. |

### ****2.2 Key Features****

| **Feature** | **Description** |
| --- | --- |
| **Dynamic Range Adjustment** | Updates the valid guess range (e.g., 1–100 → 50–100 after a low guess). |
| **Input Validation** | Ensures numeric input within the current range. |
| **Multi-Game Support** | Tracks cumulative stats across sessions. |

## ****3. Game Rules****

### ****3.1 Objective****

* Guess the secret number between **1 and 100** in the fewest attempts.

### ****3.2 Gameplay Mechanics****

### The game generates a random number.

* Player guesses a number.
* System responds with:
* **Correct!** (Game ends)

⬆ **Too high!** (Adjusts max range)

⬇ **Too low!** (Adjusts min range)

### ****3.3 Scoring System****

* **1 point** per correct guess.
* Attempts are recorded for each game.

## ****Implementation Details****

### ****Random Number Generation****

int generateRandomNumber() {

return rand() % 100 + 1; // 1–100

}

* **Improvement Suggestion**: Use C++11’s <random> for better randomness.

### ****Input Validation****

while (!validInput) {

if (cin >> guess) {

if (guess >= minRange && guess <= maxRange) validInput = true;

else cout << "Enter a number within " << minRange << "-" << maxRange << "!\n";

} else {

cin.clear(); // Reset error flags

cin.ignore(10000, '\n'); // Discard bad input

}

}

* **Why?** Prevents crashes from non-numeric input.

### ****4.3 Game Loop Logic****

if (guess == randomNumber) {

cout << "Correct! Attempts: " << attempts << endl;

totalScore++;

} else if (guess < randomNumber) {

cout << "Too low!\n";

minRange = guess + 1; // Narrow range

} else {

cout << "Too high!\n";

maxRange = guess - 1;

}

* **Dynamic Adjustment**: Guides the player efficiently.

### ****4.4 Statistics Tracking****

void showStatistics(int games, int score, int attempts) {

cout << "\*\*\*\*\*\*\*\*\*\* Stats \*\*\*\*\*\*\*\*\*\*\n";

cout << "Total Games: " << games << endl;

cout << "Total Wins: " << score << endl;

cout << "Last Attempts: " << attempts << endl;

cin.ignore(); // Pauses for user

}

* **Displays**: Session and cumulative performance.

## ****5. Technical Considerations****

### ****5.1 Strengths****

* **Simple Code Structure** – Easy to read and modify.
* **Robust Input Handling** – Prevents crashes from invalid inputs.

### ****5.2 Limitations****

❌ **No Data Persistence** – Stats reset when the program closes.  
❌ **Basic RNG** – rand() has predictability issues.

### ****5.3 Potential Improvements****

| **Feature** | **Benefit** |
| --- | --- |
| **Save/Load Stats** | Use file I/O (fstream) to retain history. |
| **Difficulty Levels** | Adjust range (e.g., 1–50 for Easy, 1–200 for Hard). |
| **Visual Feedback** | Add ASCII art or colors for engagement. |

## ****6. Conclusion****

This project demonstrates:

* **Core C++ concepts** (loops, conditionals, I/O).
* **User-friendly design** (input validation, dynamic feedback).
* **Expandability** (easy to add features like high-score tracking).

**Future Work**: Integrate <random>, add a GUI, or implement multiplayer.