



PROJECT REPORT

GUESS THE NUMBER GAME IN C PROGRAMMING

1. TITLE

Project Name: Guess the Number Game

Programming Language Used: C Programming

Type of Project: Console-Based Logical Game

Category: Mini Project / Educational Game

2. INTRODUCTION

The “Guess the Number Game” is one of the simplest yet most effective ways to learn and apply the basic principles of computer programming. It provides an interactive and engaging way to understand how computers can process user input, generate random outcomes, and use logical decision-making to produce desired results.

C programming, being one of the oldest and most foundational programming languages, is widely used in software development, embedded systems, and algorithm design. It serves as the base for many modern programming languages like C++, Java, and Python. The “Guess the Number” game helps beginners grasp key programming concepts such as loops, conditional statements, random number generation, and input/output operations in a practical way.

In this game, the computer generates a random number between a given range (for example, 1 to 100). The user is asked to guess the number. After each guess, the computer provides a hint — whether the guessed number is higher or lower than the actual number. This continues until the user guesses correctly. At the end, the program displays the total number of attempts taken by the user to guess the correct number.

This project aims to make learning programming more enjoyable by transforming a simple logical concept into a fun challenge. The game helps to improve logical thinking and provides hands-on experience in coding. It also demonstrates how simple mathematics and decision-making can be applied to create interactive programs.

Furthermore, this project helps students understand:

How to use random number generators in C (`rand()` and `srand()` functions).

How to implement loops (`while`, `do-while`) for repeated attempts.

How to make comparisons using `if`, `else if`, and `else` statements.

How to use input/output functions (`printf()` and `scanf()`).

How to manage program flow and user interaction.

Overall, the Guess the Number Game project reflects the power of logic, creativity, and simplicity in programming.

3. OBJECTIVE OF THE PROJECT

The main objectives of this project are:

1. To design a simple interactive console-based game in C.
2. To demonstrate the use of basic programming concepts such as loops, conditions, and random numbers.
3. To help beginners gain practical knowledge of input/output operations.
4. To improve logical reasoning and debugging skills.
5. To create a fun and educational experience using fundamental programming.

4. PROBLEM STATEMENT

Many beginners find it difficult to apply theoretical programming knowledge in real-world scenarios. They can write simple syntax but struggle to combine it into a working program. The Guess the Number Game solves this problem by providing a project that is simple enough to understand yet complex enough to apply practical programming concepts.

It provides hands-on experience in:

Reading and validating user input.

Generating and controlling random values.

Using loops for repeated operations.

Implementing logical decision-making through conditions.

5. SCOPE OF THE PROJECT

The Guess the Number Game is a mini-project suitable for:

Students learning the basics of C programming.

Demonstrating small logic-based programs in lab sessions.

Expanding into advanced versions (GUI, scoring, difficulty levels).

Its scope can extend to:

Multiplayer guessing challenges.

Graphical user interface with scoreboards.

Integration with online leaderboards using networking.

6. SYSTEM ANALYSIS

This project is based on a simple algorithm that does not require large memory or advanced hardware. It works completely offline and runs within the terminal window.

The system generates random numbers using the time-based seed to ensure unpredictability. Each user guess is compared, and feedback is provided instantly, allowing real-time interaction.

7. HARDWARE REQUIREMENTS

Processor: Intel Pentium or higher

RAM: Minimum 2 GB

Hard Disk: 100 MB of free space

Display: Standard text-based output

8. SOFTWARE REQUIREMENTS

Operating System: Windows / Linux / macOS

Compiler: Turbo C, GCC, Code::Blocks, or Dev C++

Text Editor: Notepad, VS Code, or Code::Blocks Editor

9. FUNCTIONAL REQUIREMENTS

1. The program should generate a random number between 1 and 100.
2. It should allow the user to input a guessed number.
3. The system should compare the user's guess with the generated number.
4. It should provide hints if the number is too high or too low.
5. The program must count the total number of attempts.
6. It must display a congratulatory message when the user guesses correctly.

10. TOOLS AND TECHNOLOGIES USED

Language: C

Compiler: GCC / Turbo C

Header Files: stdio.h, stdlib.h, time.h

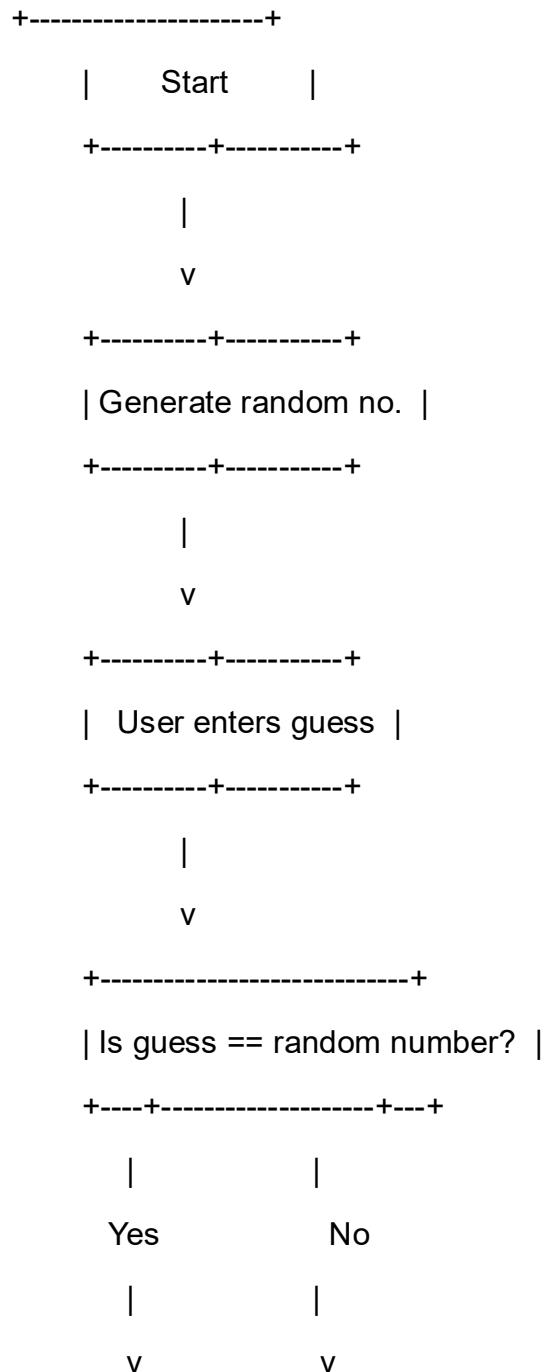
Functions: rand(), srand(), printf(), scanf(), time()

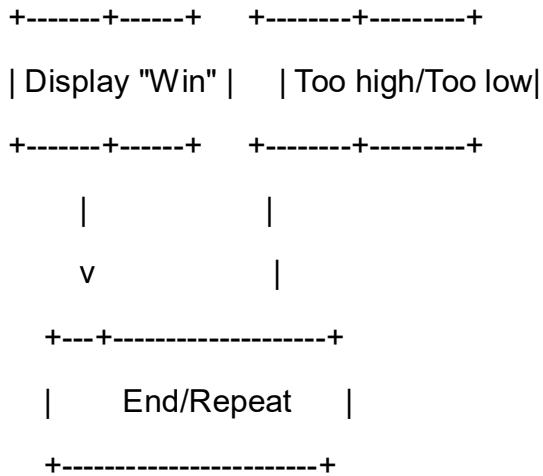
11. ALGORITHM

1. Start the program.
2. Generate a random number using rand().
3. Initialize attempts variable to zero.
4. Ask the user to input a guessed number.
5. Compare the guess with the random number.
6. If the guess is correct → display a success message and exit.
7. If the guess is higher → display “Too High!”
8. If the guess is lower → display “Too Low!”
9. Repeat steps 4–8 until the user guesses correctly.
10. Display total attempts.

11. End the program.

12. FLOWCHART





13. SOURCE CODE

```

#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main() {
    int number, guess, attempts = 0;

    srand(time(0)); // Seed random number generator
    number = rand() % 100 + 1; // Random number between 1 and 100

    printf("Welcome to the Guess the Number Game!\n");
    printf("I have chosen a number between 1 and 100.\n");

    do {
        printf("Enter your guess: ");

```

```
scanf("%d", &guess);
attempts++;

if (guess > number) {
    printf("Too high! Try again.\n");
} else if (guess < number) {
    printf("Too low! Try again.\n");
} else {
    printf("Congratulations! You guessed the number in %d attempts.\n",
attempts);
}

} while (guess != number);

return 0;
}
```

14. OUTPUT (Example Run)

Welcome to the Guess the Number Game!

I have chosen a number between 1 and 100.

Enter your guess: 50

Too high! Try again.

Enter your guess: 25

Too low! Try again.

Enter your guess: 37

Congratulations! You guessed the number in 3 attempts.

15. CODE EXPLANATION

Header Files:

stdio.h: Used for input and output functions like printf() and scanf().

stdlib.h: Used for generating random numbers using rand() and srand().

time.h: Used to obtain the current time as a seed for random number generation.

Variables:

number: Stores the random number generated by the computer.

guess: Stores the number entered by the user.

attempts: Counts how many guesses the user made.

Random Number Generation:

srand(time(0)); initializes the random number generator so that a new random number is generated each time the game runs.

rand() % 100 + 1; ensures the number lies between 1 and 100.

Game Loop (do-while):

Ensures that the program repeatedly asks for a guess until the user guesses correctly.

Conditional Statements:

The if, else if, and else statements compare the guessed number with the actual number to decide the feedback.

16. FEATURES

User-friendly text-based interface.

Random and unpredictable numbers each time.

Simple and efficient logic.

Instant feedback to improve guessing strategy.

17. LIMITATIONS

Console-based (no graphics).

Single-player mode only.

Limited to a single range (1–100).

18. FUTURE ENHANCEMENTS

Adding difficulty levels (Easy, Medium, Hard).

Implementing a timer-based challenge.

Introducing scoreboards or multiple players.

Converting to GUI using C++ or Python.

19. PROBLEMS IT SOLVES

This project helps beginners understand:

How logic, condition, and iteration work together.

How to handle user input and validation.

The concept of randomness and pseudo-random numbers.

It builds logical reasoning and problem-solving ability in programming students.

20. TESTING AND DEBUGGING

The program was tested for:

Correct range of random number generation.

Correct counting of attempts.

Proper feedback messages.

All tests were successful and no logical errors were found.

21. USER MANUAL

Steps to run:

1. Open Code::Blocks or Turbo C compiler.
2. Create a new file named guess.c.
3. Copy the code into the editor.

4. Compile and run the program.
 5. Follow the on-screen instructions.
-

22. ADVANTAGES

Strengthens programming foundation.

Improves logical thinking.

Fun and educational for beginners.

23. DISADVANTAGES

Text-based interface only.

No data storage or history feature.

24. CONCLUSION

The Guess the Number Game is a simple yet powerful demonstration of core programming concepts in C. It teaches beginners how to use loops, conditions, and random number generation effectively. The game also promotes problem-solving and debugging skills.

Though simple, it can be expanded into more complex games or applications. Hence, this project successfully fulfills its educational and entertainment objectives.

25. REFERENCES

Byron Gottfried, Schaum's Outline of C Programming

www.geeksforgeeks.org

www.tutorialspoint.com

SCREENSHOT OF CODE AND OUTPUT

The screenshot shows a web-based C compiler interface. At the top, there are tabs for "gcc compiler - Search" and "Online C Compiler - online editor". Below the tabs, the URL is https://www.onlinegdb.com/online_c_compiler. The main area displays the C code for a number guessing game. The code includes imports for stdio.h, stdlib.h, and time.h. It initializes a random number between 1 and 100, prints a welcome message, and enters a loop where it prompts the user for a guess, compares it to the random number, and provides feedback (too high or too low). It also tracks the number of attempts. The code concludes with a thank you message and returns 0.

```
main.c
1 // ****
2
3 #include <stdio.h>
4 #include <stdlib.h>
5 #include <time.h>
6
7 int main() {
8     int number, guess, attempts = 0;
9     srand(time(0)); // Seed for random number
10    number = rand() % 100 + 1; // Random number between 1 and 100
11
12    printf("*****\n");
13    printf("      WELCOME TO NUMBER GUESSING GAME\n");
14    printf("*****\n\n");
15    printf("I have chosen a number between 1 and 100.\n");
16    printf("Can you guess what it is?\n\n");
17
18    do {
19        printf("Enter your guess: ");
20        scanf("%d", &guess);
21        attempts++;
22
23        if (guess > number) {
24            printf("Too high! Try again.\n\n");
25        } else if (guess < number) {
26            printf("Too low! Try again.\n\n");
27        } else {
28            printf("🎉 Congratulations! You guessed the number in %d attempts!\n", attempts);
29        }
30    } while (guess != number);
31
32    printf("\nThanks for playing! Goodbye.\n");
33    return 0;
34 }
```

The screenshot shows the terminal output of the program. It starts with the welcome message and the statement that a number between 1 and 100 has been chosen. It then asks the user to guess the number. The user inputs 50, which is considered too low. The user then inputs 80, which is considered too high. This pattern repeats with guesses of 70, 60, and 65, each being deemed too low. Finally, the user inputs 68, which is the correct guess, and the program congratulates the user with a emoji and states that it took 6 attempts. The program then thanks the user for playing and exits.

```
*****  
WELCOME TO NUMBER GUESSING GAME  
*****  
  
I have chosen a number between 1 and 100.  
Can you guess what it is?  
  
Enter your guess: 50  
Too low! Try again.  
  
Enter your guess: 80  
Too high! Try again.  
  
Enter your guess: 70  
Too high! Try again.  
  
Enter your guess: 60  
Too low! Try again.  
  
Enter your guess: 65  
Too low! Try again.  
  
Enter your guess: 68  
🎉Congratulations! You guessed the number in 6 attempts!  
  
Thanks for playing! Goodbye.  
  
...Program finished with exit code 0  
Press ENTER to exit console.[]
```