C++:

This language is used to create the logic for the game, manage the leaderboard, and handle user input and output.

Instruction and Ideas:

C++ Code Development:

Knowledge of the syntax and semantics of C++.

Knowledge of common input and output procedures.

Familiarity with control structures (conditionals, loops).

Use of functions to promote code reuse and modularity.

Project Overview:

The Number Guessing Game is a simple console-based game where the player attempts to guess a randomly generated number within a specified range. The game includes several features to enhance the user experience, such as difficulty levels, hint provision, score tracking, and replayability.

Features

1. **Difficulty Levels**:
    - **Easy**: Number range is 1-50, and the player has 10 attempts.
    - **Medium**: Number range is 1-100, and the player has 7 attempts.
    - **Hard**: Number range is 1-200, and the player has 5 attempts.

2. **Hint Feature**:
    - If the player has two attempts remaining and still hasn't guessed correctly, a hint is provided indicating whether the number is odd or even.

3. **Score Tracking**:

- The game tracks the number of attempts it takes for the player to guess the number correctly.
- The fewer attempts taken, the better the player's performance.

4. **Replay Option**:
   - After the game ends, the player is given the option to play again.

5. **Input Validation**:
   - Ensures that the player's guess is within the valid range and is a numerical value.
   - Handles invalid input gracefully without crashing.

How it Works:

1. **Initialization**:
   - The program seeds the random number generator to ensure different random numbers in each game session.
   - The player is greeted and prompted to select a difficulty level.
2. **Game Loop**:
   - Based on the chosen difficulty, the game sets the maximum number range and the number of trials.
   - The game starts, and the player is prompted to guess a number within the range.
   - After each guess, the game checks if the guess is correct, too high, or too low and provides feedback accordingly.
   - If the player guesses the number correctly, a congratulatory message is displayed along with the number of attempts taken.
   - If the player fails to guess the number within the allowed attempts, the game reveals the correct number.
   - A hint is provided if the player has two attempts remaining and hasn't guessed correctly.
3. **Replay Option**:
   - After the game concludes, the player is asked if they want to play again.
   - If the player chooses to play again, the game restarts; otherwise, the game ends with a thank-you message.

Code Structure:

• Header **Files**:

- `<iostream>`: For input and output operations.
- `<cstdlib>`: For random number generation.
- `<ctime>`: For seeding the random number generator.

- `<climits>`: For handling the maximum integer value in input validation.

- Main **Function**:
  - Initializes the game and handles the main loop for replay functionality.

- play **Game Function**:
  - Takes the maximum range and the number of trials as parameters.
  - Implements the game logic, including input validation, guessing mechanism, hint provision, and feedback.

CODE:

```
#include <iostream>

#include <cstdlib>

#include <ctime>

#include <climits>

using namespace std;


void playGame(int maxRange, int maxTrials);


int main() {

   srand(time(0));  // Seed the random number generator


   cout << "Welcome to the Random Number Guessing Game!" << endl;


   char playAgain = 'y';

   while (playAgain == 'y' || playAgain == 'Y') {
```

```
int difficulty;
cout << "Select difficulty level (1. Easy 2. Medium 3. Hard): ";
cin >> difficulty;

int maxRange, maxTrials;
switch(difficulty) {
    case 1:
        maxRange = 50;
        maxTrials = 10;
        break;
    case 2:
        maxRange = 100;
        maxTrials = 7;
        break;
    case 3:
        maxRange = 200;
        maxTrials = 5;
        break;
    default:
        cout << "Invalid choice. Defaulting to Medium difficulty." << endl;
        maxRange = 100;
        maxTrials = 7;
```

```
        }

        playGame(maxRange, maxTrials);

        cout << "Do you want to play again? (y/n): ";
        cin >> playAgain;
    }

    cout << "Thank you for playing!" << endl;
    return 0;
}

void playGame(int maxRange, int maxTrials) {
    int ranNum = rand() % maxRange + 1;
    bool win = false;
    int guess;
    int pTries = 0;

    cout << "Guess a number between 1 and " << maxRange << endl;

    while(!win && pTries < maxTrials) {
        cout << "Trial " << (pTries + 1) << " of " << maxTrials << ". Enter your guess:
";
```

```
        cin >> guess;


        if (cin.fail() || guess < 1 || guess > maxRange) {
            cin.clear(); // clear the error flags
            cin.ignore(INT_MAX, '\n'); // discard invalid input
            cout << "Invalid input. Please enter a number between 1 and " <<
maxRange << "." << endl;
            continue;
        }


        pTries++;


        if (guess == ranNum) {
            cout << "Congratulations! You guessed the number in " << pTries << "
tries." << endl;
            win = true;
        } else if (guess < ranNum) {
            cout << "Too low!" << endl;
        } else {
            cout << "Too high!" << endl;
        }


        if (!win && pTries == maxTrials - 2) {
```

```
        if (ranNum % 2 == 0) {
            cout << "Hint: The number is even." << endl;
        } else {
            cout << "Hint: The number is odd." << endl;
        }
    }
  }

  if (!win) {
    cout << "You failed to guess the number. The correct number was " << ranNum << "." << endl;
  }
}
```

Output:

Screenshot: →

Conclusion:

This project provides a fun and educational way to practice C++ programming, especially focusing on control structures, user input validation, and basic game logic.