PROJECT DOCUMENT

Presented by Aruba Khan

PROBLEM STATEMENT

Write a C++ program that implements a number guessing game. The program should generate a random number between -100 and 50, inclusive, and allow the user to guess the number. The program should provide feedback if the guess is too high or too low and keep track of the number of attempts made by the user. The game should handle invalid input gracefully and should congratulate the user when they guess the correct number, displaying the total number of attempts made.

INSTRUCTIONS

- Initialize the Random Number Generator: The program starts by seeding the random number generator using the current time. This ensures that each time the program runs, a different random number is generated.
- Generate a Secret Number: The program generates a random number between -100 and 50. This will be the number the user has to guess.
- Welcome the Player: A welcome message is displayed to the player, explaining the range of the random number.
- User Input Loop:
- The program enters a loop where it repeatedly prompts the user to guess the number.
- It keeps track of the number of attempts made by the user.
- Input Validation:
- The program checks if the input is valid. If the user enters something that isn't a number, the program will notify them of the invalid input and prompt them to enter a valid number.
- Comparison and Feedback:
- If the user's guess is lower than the secret number, the program tells them the guess is too low.
- If the guess is higher than the secret number, the program tells them the guess is too high.
- If the guess matches the secret number, the program congratulates the user and displays the number of attempts made.
- End of Game: Once the user guesses the number correctly, the loop breaks, and the game ends.

CHALLENGES AND SOLUTION

- Generating a Random Number in a Specific Range:
- Challenge: Ensuring the random number falls between -100 and 50.
- Resolution: Use the formula rand() % 151 - 100 to generate a number in the desired range. This ensures the output falls between -100 and 50, inclusive.
- Seeding the Random Number Generator:
- Challenge: Without proper seeding, the random number generator produces the same sequence of numbers every time the program runs.
- Resolution: Use srand(time(nullptr)) to seed the random number generator with the current time, ensuring different sequences of random numbers for each program execution.
- Handling User Input:
- Challenge: Ensuring that the program can handle non-numeric input gracefully.
- Resolution: Check for input failures using cin.fail(), clear the error state with cin.clear(), and discard invalid input using cin.ignore(numeric_limits<std::streamsize>::max(), '\n'). This prevents the program from crashing and prompts the user to enter a valid number.

- Providing Feedback to the User:
- Challenge: Giving appropriate feedback based on whether the guess is too high, too low, or correct.
- Resolution: Use conditional statements (if, else if, else) to compare the user's guess with the secret number and provide corresponding feedback.
- Tracking the Number of Attempts:
- Challenge: Keeping track of how many guesses the user has made.
- Resolution: Initialize an attempts variable to 0 and increment it each time the user makes a guess. This allows the program to display the total number of attempts when the user correctly guesses the number.
- Exiting the Loop Upon Correct Guess:
- Challenge: Breaking out of the loop when the user guesses the correct number.
- Resolution: Use a break statement within the loop to exit when the user's guess matches the secret number.

LinkedIn Post Link:
https://www.linkedin.com/feed/update/urn:li:activity:7214011679758553090/

GitHub Post Link:
https://github.com/Arubakhan-22/Handling-Data-Types-and-Variables