PROJECT REPORT

Code Debugging Simulator

A C++ Console-Based Learning Tool

Submitted by: Aayan Amir

Code Debugging Simulator - Project Report

1. Introduction

The Code Debugging Simulator is a C++ console-based application designed to help users practice identifying and fixing bugs in code snippets. The program presents users with buggy code segments across three difficulty levels (Beginner, Intermediate, and Advanced) and allows them to input corrections. The system checks their solutions and provides feedback.

Objectives:

- Improve users' debugging skills in C++.
- Provide an interactive learning experience.
- Track user performance (attempts and correct fixes).

2. Features

Three Difficulty Levels:

- Beginner: Simple syntax errors (missing semicolons, incorrect quotes, etc.).
- Intermediate: Logical errors (off-by-one loops, incorrect conditions).
- Advanced: Memory issues (pointers, infinite loops).

Interactive Debugging:

- Users input fixes line-by-line.
- Option to clear input (==again) and restart.
- Automatic comparison with correct solutions.

Visual Feedback:

- Console color changes (red for incorrect, green for correct).
- Displays correct solution if the user's fix is wrong.

Performance Tracking:

- Tracks total attempts and correct fixes.
- Displays final score upon exit.

Code Debugging Simulator - Project Report

3. Implementation Details

Data Structures Used:

- Arrays of Strings: Stores buggy and fixed code snippets.

  string begbug[], begsol[], interbug[], intersol[], advbug[], advsol[];

Key Functions:

- cleanans(string& str): Removes whitespace and newlines for comparison.
- menu(): Displays the main menu with difficulty options.
- playGame(int level, int& attempts, int& correct): Selects a random buggy code snippet and takes user input for fixes.
- main(): Manages the game loop.

External Libraries:

- <windows.h>: Used for console color change etc.
- <cstdlib>: Provides rand() and srand() for random selection.

4. Code Structure

Code Snippet Storage:

- Beginner Level: Simple mistakes like missing semicolons, wrong quotes, missing return statements.
- Intermediate Level: Logical errors like incorrect conditions, array out-of-bounds.
- Advanced Level: Memory issues, infinite loops, uninitialized references.

User Input Handling:

- Full code cleaning process to remove spaces, new lines etc from the user's answer so that only the code can be checked for errors
- Uses getline() for multi-line input.
- Supports ==again to reset input.
- Terminates input with 'done'.

Scoring Mechanism:

- Increments attempts and correct counters.
- Displays final score upon exit

Code Debugging Simulator - Project Report

5. Sample Output

Example Gameplay (Beginner Level):

==== Code Debugging Simulator ====

1. Beginner Level

2. Intermediate Level

3. Advanced Level

4. Exit

Choose an option: 1

Fix the following code snippet:

```
int main() {
  cout << 'Hello World';
  return 0;
}
```

Enter your fix (type 'done' to finish):

```
int main() {
  cout << "Hello World";
  return 0;
}
done
```

Congratulations! You fixed the bug!

Final Score Display:

Thank you for playing! Your score:

Total attempts: 5

Correct attempts: 3

Code Debugging Simulator - Project Report

6. Limitations & Future Improvements

Limitations:

- No syntax validation before comparing solutions.
- Limited to predefined code snippets.
- No difficulty progression tracking.

Possible Enhancements:

- Dynamic Code Generation.
- Syntax Highlighting.
- Time-Based Challenges.
- User Profiles.
- More Languages (Python, Java, etc.).

7. Conclusion

The Code Debugging Simulator is a simple yet effective tool for practicing C++ debugging. It helps users identify common mistakes and reinforces best practices. With further improvements, it could become a comprehensive learning platform for programmers of all levels.

Submitted by: [Your Name]

Date: April 12, 2025

GitHub Repository: [If applicable]