Simple Calculator perform

+ , - , * , /, Using Classes in C++

1. Introduction

In this project, I developed a fully functional Simple Calculator program using C++ that demonstrates the concept of object-oriented programming (OOP). The calculator performs basic arithmetic operations such as addition, subtraction, multiplication, and division. The project was designed to help understand how classes, objects, and functions interact together to create a structured program. It also provided hands-on experience in using encapsulation, class methods, and error handling.

2. Objectives

The main objectives of this project were to:
1. Understand and apply the concept of classes and objects in C++.
2. Create user-defined functions to perform basic arithmetic operations.
3. Implement input validation and error handling (especially for division by zero).
4. Design a simple and user-friendly calculator that can take input and display results accurately.

3. Problem Statement

The problem was to design a calculator program that could perform four basic operations on two user-input numbers. The program should allow users to enter two numbers, choose the desired operation, and display the correct output. The solution needed to ensure error handling for invalid inputs or division by zero.

4. Methodology

The project followed a simple yet systematic approach:
• A class named 'Calculator' was defined with private data members num1 and num2.
• Member functions add + , subtract - , multiply * , and divide / were created to perform operations.
• User inputs were collected using standard input (cin) and stored in the Calculator object.
• The program displayed a menu to let users select an operation, then executed the corresponding function.
• Division by zero was handled using an if-condition that displayed an error message instead of crashing.

5. Implementation (Code Explanation)

The implementation involved defining a Calculator class and writing the main() function to control program flow. Each operation was defined as a separate function for modularity and clarity. Below is the complete C++ code:

```
#include <iostream>
using namespace std;

// Define a class for the calculator
class Calculator {
private:
    double num1, num2;  // Variables to store two numbers

public:
    // Function to set the values of num1 and num2
    void setValues(double n1, double n2) {
        num1 = n1;
        num2 = n2;
    }

    // Function to perform addition
    double add() {
        return num1 + num2;
    }

    // Function to perform subtraction
    double subtract() {
        return num1 - num2;
    }

    // Function to perform multiplication
    double multiply() {
        return num1 * num2;
    }

    // Function to perform division
    double divide() {
        if (num2 == 0) {
            cout << "Error: Division by zero is not allowed." << endl;
            return 0;
        } else {
            return num1 / num2;
        }
    }
```

```cpp
};

int main() {
    Calculator calc;  // Create an object of the Calculator class
    double num1, num2;
    int choice;

    cout << "Simple Calculator\n";
    cout << "Enter two numbers:\n";
    cout << "Number 1: ";
    cin >> num1;
    cout << "Number 2: ";
    cin >> num2;

    // Set the values in the Calculator object
    calc.setValues(num1, num2);

    // Display the operation menu
    cout << "\nChoose operation to perform:\n";
    cout << "1. Add (+)\n";
    cout << "2. Subtract (-)\n";
    cout << "3. Multiply (*)\n";
    cout << "4. Divide (/)\n";
    cout << "Enter your choice (1-4): ";
    cin >> choice;

    // Perform the chosen operation
    switch(choice) {
        case 1:
            cout << "Result: " << calc.add() << endl;
            break;
        case 2:
            cout << "Result: " << calc.subtract() << endl;
            break;
        case 3:
            cout << "Result: " << calc.multiply() << endl;
            break;
        case 4:
            cout << "Result: " << calc.divide() << endl;
            break;
        default:
            cout << "Invalid choice!" << endl;
    }

    return 0;
}
```

7. Challenges and Solutions

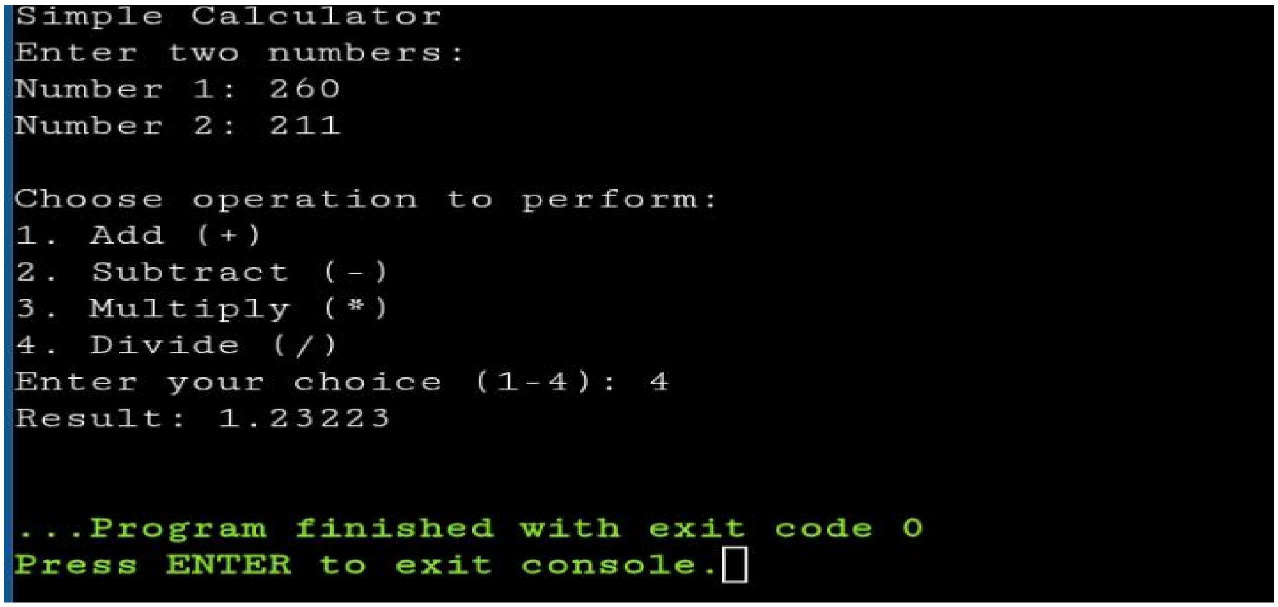
One of the main challenges was handling division by zero, which could cause runtime errors. This was resolved by adding a condition to check if the second number is zero before performing the division. Another challenge was ensuring modularity and reusability of code, which was achieved by using separate functions for each operation.

8. Future Enhancements

To improve the project further, I plan to:
• Add advanced mathematical functions (like power, square root, modulus, etc.).
• Create a graphical interface using C++ GUI libraries.
• Allow continuous calculations without restarting the program.
• Save calculation history to a file for later review.

Output and Results

When the program is executed, it prompts the user to input two numbers and choose an operation. Based on the user's choice, the correct mathematical result is displayed. The output is clean and informative, and the error message is shown properly when attempting to divide by zero.



```
Simple Calculator
Enter two numbers:
Number 1: 260
Number 2: 211

Choose operation to perform:
1. Add (+)
2. Subtract (-)
3. Multiply (*)
4. Divide (/)
Enter your choice (1-4): 4
Result: 1.23223


...Program finished with exit code 0
Press ENTER to exit console.
```

This project successfully demonstrated the concept of object-oriented programming in C++. By implementing a calculator using classes and methods, I gained practical understanding of class structures, encapsulation, and data abstraction. The project enhanced my problem-solving skills and taught me the importance of writing clean, maintainable code