PROJECT
BASIC CALCULATOR

PREPARED BY

Sana Fatima

PREPARED TO

A&D Tech

sana.faati@gmail.com

ad.techinnov25@gmail.com

TABLE OF CONTENTS

Problem Statement

Solution Overview

Challenges and Resolution

Instructions

Problem Statement

Create a command-line calculator that performs basic arithmetic operations.

1. Define functions for each arithmetic operation:

- addition(a, b) returns the sum of a and b.

- subtraction(a, b) returns the difference between a and b.
- multiplication(a, b) returns the product of a and b.
- division(a, b) returns the quotient of a and b. Ensure you handle
- division by zero by returning a message like "Division by zero is not allowed."

2. Write main function that:

- Displays a menu for the user to choose an operation.
- Asks the user for two numbers.
- Calls the appropriate function based on the user's choice and displays the result.
- Allows the user to perform another operation or exit.

3. Testing Your Calculator:

- Test each function to make sure they handle various inputs correctly.
- Ensure your program gracefully handles invalid inputs like non-numeric values.

SOLUTION OVER VIEW

To solve this problem, I took a step-by-step approach:

1.Define the requirements:

Determine what operations the calculator should perform (e.g., addition, subtraction, multiplication, division).

2.Understood the problem:

I read and analyzed the task to identify the issue.

3.Choose a programming language and write a code:

Select a language to implement the calculator (e.g., C++).

4.Refine and optimize:

Improve the code as needed (e.g., add more operations, improve error handling, optimize performance)

5.Identified the root cause:

I determined that the problem was with input validation and handling invalid inputs.

6.Handled invalid inputs:

I added checks to handle cases where alphabets or special characters are entered instead of numbers.

7.Added error messages:

I displayed error messages to inform the user about invalid inputs.

8.Tested the solution:

I tested the code with various inputs to ensure it worked correctly.

By taking a methodical approach, I was able to identify and fix the issues in the code, ensuring that it now handles invalid inputs and performs calculations correctly.

CHALLENGES AND RESOLUTIONS

Challenges:

1. Input Validation: Ensuring that user input is valid and numeric.
2. Error Handling: Gracefully handling errors and informing the user.
3. Code Structure: Organizing the code to be efficient and easy to understand.
4. Calculation Logic: Ensuring accurate calculations for different operations.

Resolutions:

1. Input Validation: Added checks to ensure input is numeric using cin and stringstream.
2. Error Handling: Displayed error messages to inform the user about invalid inputs.
3. Code Structure: Organized the code using functions and loops for better readability and maintainability.
4. Calculation Logic: Implemented correct calculation logic for each operation.

By addressing these challenges and implementing effective solutions, the code now:

- Validates user input to prevent errors

- Handles errors gracefully and informs the user

- Is organized and easy to understand

- Performs calculations accurately

The code is now more robust, reliable, and user-friendly.

INSTRUCTIONS

By following these instructions, you should be able to set up and use the project/application successfully.Setting Up the Project

1. Include the required libraries:
   - iostream
   - string
   - sstream
2. Create a new C++ project:
   - Open IDE (Integrated Development Environment) e.g (Dev c++) and create a new C++ project.
   - Name the project (e.g., "BasicCalculator").
3. Add the source code:
   - Write the source code
   - Make sure to include all the necessary files (e.g., main.cpp).
4. Compile and build the project:
   - Compile the code using your IDE's built-in compiler.
   - Build the project to create an executable file.

Using the Application

1. Run the application:
   - Open a terminal or command prompt and navigate to the directory where the executable file is located.
   - Run the application.
2. Choose an operation:
   - The application will display a menu with four options (Addition, Subtraction, Multiplication, Division).

    - Enter the number of the operation you want to perform (e.g., 1 for Addition).
3. Enter numbers:
    - The application will prompt you to enter two numbers.
4. Get the result:
    - The application will display the result of the operation.
5. Repeat or quit:
    - You can repeat the process by choosing another operation or quit by entering q.

Tips and Variations

- Input validation: The application includes input validation to ensure that only numeric values are accepted. If you enter an invalid input, the application will display an error message and prompt you to try again.
- Calculation logic: The application uses the correct calculation logic for each operation. For example, division by zero is handled gracefully, and the application will display an error message instead of crashing.
- Code organization: The code is organized using functions and statements for better readability and maintainability.