**Scientific Calculator**

**Submitted by**

**Name of the Students: AQUASA AZIZ**

**Enrolment Number:** 12022002018003

**Section:** F

**Class Roll Number:** 26  
**Stream:** CSBS

**Subject:** Programming for Problem-Solving using C

**Subject Code:** ESC103(Pr)

**Department:** Basic Science and Humanities

Under the supervision of

**Swarnendu Ghosh**

**Academic Year: 2022-26**

PROJECT REPORT SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE FIRST SEMESTER



**DEPARTMENT OF BASIC SCIENCE AND HUMANITIES**

**INSTITUTE OF ENGINEERING AND MANAGEMENT, KOLKATA**



**CERTIFICATE OF RECOMMENDATION**

We hereby recommend that the project prepared under our supervision by AQUASA AZIZ**,** entitled Scientific Calculator accepted in partial fulfillment of the requirements for the degree of partial fulfillment of the first semester.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Head of the Department Project Supervisor

Basic Sciences and Humanities

IEM, Kolkata

# Introduction

In this project, we have created a scientific calculator using the C programming language. The calculator has all the basic functions such as addition, subtraction, multiplication, and division, along with more advanced functions like exponentiation, square root, sine, cosine, tangent, and logarithm. The user interface is designed to be user-friendly, with clear instructions and prompts. The calculator also has error-handling mechanisms to prevent erroneous inputs from causing a program crash.

**Project Scope:**

This project's scope is to create a scientific calculator using the C programming language. The calculator should have all the basic functions such as addition, subtraction, multiplication, and division, along with more advanced functions like exponentiation, square root, sine, cosine, tangent, and logarithm. The user interface should be designed to be user-friendly, with clear instructions and prompts. The calculator should also have error-handling mechanisms to prevent any erroneous inputs from causing a program crash.

**Program Design:**

The program design for our scientific calculator involves several steps. These are:

1. User interface design: The user interface design should be clear and simple, with easy-to-understand instructions and prompts. The user should be able to easily select the desired operation using a menu system.
2. Function design: The program should have several functions to handle the various operations, such as addition, subtraction, multiplication, division, exponentiation, square root, sine, cosine, tangent, and logarithm. These functions should take input from the user, perform the necessary calculations, and output the result.
3. Error handling: The program should have error handling mechanisms to prevent any erroneous inputs from causing a program crash. The program should display appropriate error messages and prompt the user to enter valid input.

**Program Implementation:**

The program implementation for our scientific calculator involves several steps. These are:

1. User interface implementation: The user interface should be implemented using simple text prompts and menus. The user should be able to easily navigate through the menus and select the desired operation.
2. Function implementation: The various functions for the calculator, such as addition, subtraction, multiplication, division, exponentiation, square root, sine, cosine, tangent, and logarithm should be implemented using the appropriate mathematical formulas and programming constructs.
3. Error handling implementation: The program should have error handling mechanisms in place to prevent any erroneous inputs from causing a program crash. The program should display appropriate error messages and prompt the user to enter valid input.

**Variables:**

* input: a character array that stores the user input expression
* token: a pointer to a character array used to split the input into tokens
* num1: a double variable used to store the first number in the expression
* num2: a double variable used to store the second number in the expression
* result: a double variable used to store the result of the operation

**Functions:**

* add(double x, double y)**: returns the sum of** x **and** y
* subtract(double x, double y)**: returns the difference between** x **and** y
* multiply(double x, double y)**: returns the product of** x **and** y
* divide(double x, double y)**: returns the quotient of** x **divided by** y
* power(double x, double y)**: returns** x **raised to the power of** y
* square\_root(double x)**: returns the square root of** x
* sine(double x)**: returns the sine of** x
* cosine(double x)**: returns the cosine of** x
* tangent(double x)**: returns the tangent of** x
* logarithm(double x)**: returns the base 10 logarithm of** x
* natural\_logarithm(double x)**: returns the natural logarithm of** x

**Note that all functions are of type** double**, as they all return a double value.**

**Files:**

This program consists of a single C file, which contains the code for a simple command-line calculator. The file likely has a .c extension and could be named anything the programmer chose, such as calculator.c

**Datasets:**

The program does not use any external datasets or files. All necessary information is contained within the code, including the available operations and their corresponding function definitions. The user input is provided via the command-line interface during program execution.

**Features:**

Our calculator program has several features that make it user-friendly and efficient. Some of them are as follows:

1. Clear and simple user interface: Our calculator has a clear and simple user interface that is easy to understand and use.
2. Basic and advanced functions: Our calculator has both basic and advanced functions that make it useful for a wide range of users.
3. Error handling: Our calculator has error handling mechanisms that prevent any erroneous inputs from causing a program crash.
4. Memory function: Our calculator has a memory function that allows users to store numbers for later use.
5. Easy to modify and extend: Our calculator is easy to modify and extend with additional functions and features

# Programs

Provide the C programs of the various modules.

#include <stdio.h>

#include <math.h>

#include <string.h>

// Define the available operations

double add(double x, double y) { return x + y; }

double subtract(double x, double y) { return x - y; }

double multiply(double x, double y) { return x \* y; }

double divide(double x, double y) { return x / y; }

double power(double x, double y) { return pow(x, y); }

double square\_root(double x) { return sqrt(x); }

double sine(double x) { return sin(x); }

double cosine(double x) { return cos(x); }

double tangent(double x) { return tan(x); }

double logarithm(double x) { return log10(x); }

double natural\_logarithm(double x) { return log(x); }

// Main calculator loop

int main() {

// Define variables

char input[100];

char \*token;

double num1, num2, result;

// Loop until user enters "quit"

while(1) {

// Get user input

printf("Enter an expression or 'quit' to exit: ");

fgets(input, 100, stdin);

input[strcspn(input, "\n")] = 0; // Remove trailing newline character

// Check for exit command

if (strcmp(input, "quit") == 0) {

break;

}

// Split the user input into tokens

token = strtok(input, " ");

// Convert the first token to a number

num1 = atof(token);

// Process the tokens

while (token != NULL) {

// Get the next token

token = strtok(NULL, " ");

// Check if the token is an operator

if (token != NULL) {

// Convert the next token to a number

num2 = atof(token);

// Perform the operation

if (strcmp(token, "+") == 0) {

result = add(num1, num2);

} else if (strcmp(token, "-") == 0) {

result = subtract(num1, num2);

} else if (strcmp(token, "\*") == 0) {

result = multiply(num1, num2);

} else if (strcmp(token, "/") == 0) {

result = divide(num1, num2);

} else if (strcmp(token, "^") == 0) {

result = power(num1, num2);

} else if (strcmp(token, "sqrt") == 0) {

result = square\_root(num1);

} else if (strcmp(token, "sin") == 0) {

result = sine(num1);

} else if (strcmp(token, "cos") == 0) {

result = cosine(num1);

} else if (strcmp(token, "tan") == 0) {

result = tangent(num1);

} else if (strcmp(token, "log") == 0) {

result = logarithm(num1);

} else if (strcmp(token, "ln") == 0) {

result = natural\_logarithm(num1);

}

// Set the result as the first number for the next operation

num1 = result;

}

}

// Print the result

printf("Result: %lf\n", result);

}

return 0;

}

# Outputs

