TRIBHUVAN UNIVERSITY INSTITUTE OF

ENGINEERING THAPATHALI CAMPUS

Proposal On

Scientific Calculator

Submitted By:

Nosin C. Adhikari (THA081BEI026)

Oasis Paudel (THA081BEI027)

Sujana Shrestha (THA081BEI045)

Sumit Paudel (THA081BEI046)

Submitted To:

Department of Electronics and Computer Engineering
Thapathali Campus
Kathmandu, Nepal

February,2025

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to everyone who contributed to the successful completion of our C programming project, "Calculator". This project would not have been possible without the guidance, support, and encouragement of many individuals.

First and foremost, we extend our heartfelt thanks to our C Programming instructor, Prajwal Pakka, for providing us with the opportunity to work on this project and for their invaluable guidance throughout the process. Their expertise and constructive feedback helped us overcome challenges and improve our understanding of programming concepts. We are also thankful to our colleagues and peers for their constant support, motivation, and collaborative spirit. Their suggestions and insights played a significant role in refining our project.

Lastly, we would like to acknowledge the online resources, textbooks, and tutorials that served as references during the development of this project. These resources were instrumental in enhancing our knowledge and skills in C programming. This project has been a great learning experience, and we are grateful to everyone who contributed directly or indirectly to its completion.

Nosin C. Adhikari (THA081BEI027)

Oasis Paudel (THA081BEI027)

Sujana Shrestha (THA081BEI045)

Sumit Paudel (THA081BEI046)

ABSTRACT

The purpose of this project is to develop a comprehensive calculator application when using C programming languages that support a wide range of mathematical operations commonly used in real life scenarios. The calculator will handle basic arithmetic operations (also subtraction, multiplication and partition), logarithmic functions, trigonometric functions and vector operations. By integrating these features, the calculator will give users the opportunity to perform complex calculations in areas such as engineering, science and finance. This project will serve as a practical performance of advanced programming techniques in C, including the use of functions, matrices, loops and mathematical libraries, while meeting the need for a versatile tool that can calculate the real world. This project will serve as an opportunity to apply core programming concepts in C, such as loops, conditionals, and input/output handling, while creating a practical tool for everyday use.

Keyword: C programming, Mathematical libraries, I/O handling

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATION

GUI	Graphical User Interface
CLI	Command Line Interface
IDE	Integrated Development Environment
GCC	GNU Compiler Collection
MSVC	Microsoft Visual C++

1. INTRODUCTION

1.1 Background

Calculators are one of the most widely used tools for performing mathematical operations, ranging from simple arithmetic to complex scientific computations. They are indispensable in fields such as education, engineering, finance, and research. While modern calculators come in various forms, including physical devices and software applications, building a calculator from scratch using a programming language like C offers a unique opportunity to understand the foundational principles of computation and software development [1].

The C programming language is a powerful and efficient language known for its low-level system access and portability. It is widely used in system programming, embedded systems, and performance-critical applications [2]. By developing a calculator in C, we can explore core programming concepts such as input/output handling, control structures, functions, and error management. This project not only reinforces fundamental programming skills but also provides a practical foundation for tackling more complex software challenges in the future.

1.2 Motivation

The motivation behind this project stems from the need to apply theoretical knowledge of programming into practical, real-world applications. By building a calculator, we can reinforce our understanding of key programming constructs and logic. Additionally, this project serves as an excellent opportunity to enhance problem-solving skills, learn debugging techniques, and gain confidence in writing efficient and structured code. The calculator project is not only a stepping stone for beginners such as ourselves but also a way to explore the potential of C programming in creating useful tools.

1.3 Problem Definition

Despite the availability of numerous calculators in the market, creating a custom calculator from scratch allows us to understand the underlying logic and algorithms involved in performing basic arithmetic operations. The problem we aim to address is the development of a user-friendly, command-line-based calculator that can handle addition, subtraction, multiplication, and division. The challenge lies in ensuring the program is robust, handles errors gracefully (such as division by zero), and provides accurate results for user inputs.

1.4 Objectives

The primary objectives of this project are as follows:

a. Develop a Functional Calculator: Create a calculator that can perform basic arithmetic operations, including addition, subtraction, multiplication, and division.
b. User-Friendly Interface: Design a simple and intuitive interface that allows users to input numbers and select operations easily.

c. Error Handling: Implement mechanisms to handle invalid inputs and edge cases, such as division by zero or non-numeric inputs.

d. Code Efficiency: Write clean, efficient, and well-structured code that adheres to best practices in C programming.

e. Learning and Application: Use this project as a platform to apply and reinforce programming concepts learned in the classroom, such as functions, loops, and conditional statements.

By achieving these objectives, we aim to create a reliable and efficient calculator that demonstrates the power and versatility of the C programming language.

2. LITERATURE REVIEW

The scientific calculator is a vital computational tool widely used in education, engineering, and scientific research. Unlike basic calculators, scientific calculators offer advanced functionalities such as trigonometric, logarithmic, and statistical computations. This literature review explores the historical development, underlying computational theories, and recent advancements in scientific calculators, with a focus on their implementation in the C programming language.

2.1 Historical Development

Early scientific calculators were introduced as standalone electronic devices capable of performing complex mathematical computations beyond basic arithmetic. Models like the HP-35 and Casio fx series provided engineers, scientists, and students with handheld tools to compute logarithmic, trigonometric, and exponential functions efficiently. These calculators utilized microprocessors and programmed logic circuits to enhance their mathematical capabilities [3]. Electronic scientific calculators revolutionized mathematical problem-solving in academic, engineering, and research fields by enabling quick, accurate calculations without manual computation. Early models were expensive, had small display screens, and lacked programmability. Additionally, their reliance on physical hardware limited their ability to evolve with emerging technologies. The evolution of scientific calculators can be traced back to the early 1970s with the introduction of electronic pocket calculators. Hewlett-Packard (HP) introduced the first handheld scientific calculator, the HP-35, in 1972, which revolutionized mathematical computation. Scientific calculators rely on various numerical methods and algorithms to perform complex calculations efficiently. Key computational theories include:

- Floating-Point Arithmetic: Enables representation of real numbers with high precision, following IEEE 754 standards.

- Trigonometric and Logarithmic Computations: Implemented using Taylor series, CORDIC (Coordinate Rotation Digital Computer) algorithm, and iterative approximation methods.

- Algebraic and Statistical Functions: Utilize matrix operations, polynomial evaluations, and regression analysis for statistical computations.

2.2 Software-Based Scientific Calculators

With the advancement of computing technologies, scientific calculators transitioned into software applications, available on computers, mobile devices, and online platforms. Examples include Windows Calculator and third-party applications that offer enhanced functionality beyond traditional hardware calculators. Software-based scientific calculators provide extensive mathematical capabilities, allowing users to perform a wide range of computations conveniently. They support complex functions like symbolic algebra, graphing, and statistical analysis. Some applications require internet access, consume more system resources, and may include unnecessary features, leading to a lack of optimization for basic users.[4]

2.3 Command-Line-Based Scientific Calculators in C

Command-line-based scientific calculators, developed using the C programming language, provide a lightweight, efficient alternative to GUI-based solutions. These calculators use standard libraries like math.h to perform trigonometric, logarithmic, and statistical functions.

CLI-based calculators are widely used in programming, engineering, and scientific computing, offering high precision and performance with minimal system resource consumption. The lack of a graphical interface makes them less accessible for users unfamiliar with command-line environments. They also require manual input, which may slow down usability compared to GUI-based options.[5] Implementing a scientific calculator in C requires the use of standard mathematical libraries such as <math.h>, which provides functions for trigonometry, logarithms, and exponentiation. Key aspects of implementation include:

- **User Interface Design:** Command-line or graphical interface for user input and display of results.

- **Function Handling:** Using switch-case or function pointers for efficient operation execution.

- **Error Handling:** Managing domain errors (e.g., logarithm of a negative number) and division by zero.

3. PROPOSED SYSTEM ARCHITECTURE

The different blocks of the system architecture are explained below:

3.1 Block Diagram or System Architecture

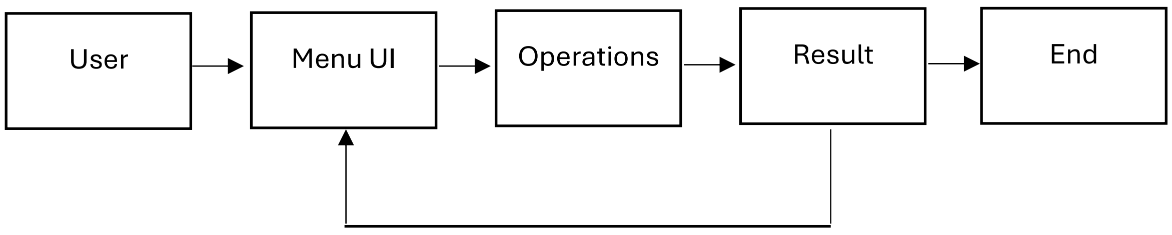

Fig 3.1 Top Level Flowchart

3.2 Parts of Program

3.2.1 User Interface

The interface of the scientific calculator includes a menu-driven structure where users can select various functions. The interface provides options such as performing calculations and exiting the program.

3.2.1.1 Calculation Module

The core of the system is the calculation module, which processes mathematical functions including basic arithmetic, trigonometry, logarithms, and exponential calculations. It is implemented using C standard libraries like math.h.

3.2.1.2 Error Handling Mechanism

Error handling ensures that invalid operations (e.g., division by zero, invalid input) are properly managed, displaying appropriate messages and preventing crashes.

3.2.1.3 Exit

The exit function safely terminates the program, ensuring that any stored data is preserved or cleared as necessary.

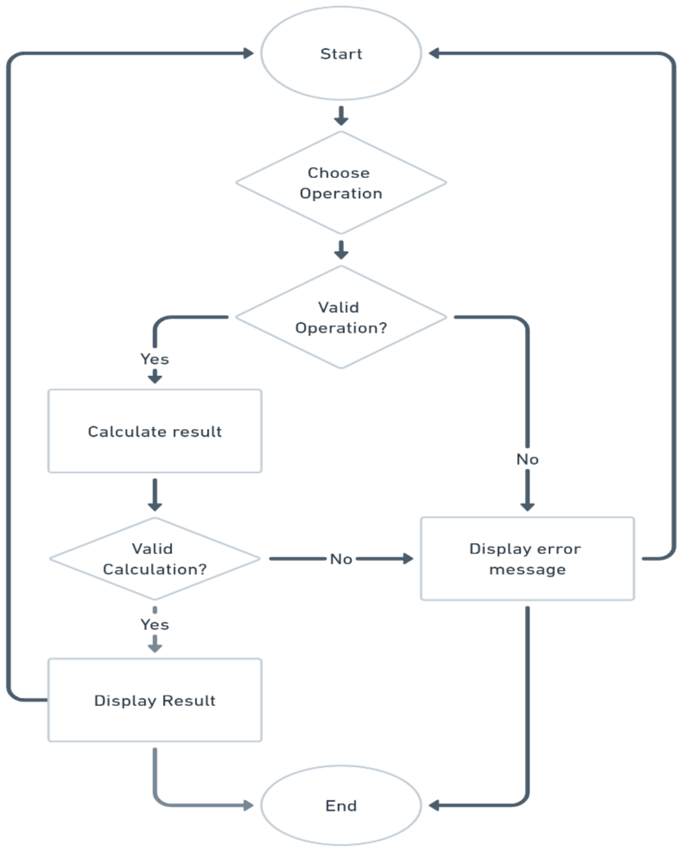

Fig 3.2 Mechanism of Scientific Calculator

3.3 Tools and Environment

The scientific calculator is developed using the C programming language, utilizing compilers such as GCC or MSVC. Development environments include:

Code::Blocks / Dev-C++ for Windows, **GCC and Terminal** for Linux, **Visual Studio Code** with integrated debugging tools

4. METHODOLOGY

The various methodologies that will be used in the development of the scientific calculator include different header files, functions, conditional statements, loops, arrays which are explained below.

4.1 Header files

In C language a C header file is a text file that contains pieces of code written in the C programming language. The name of a header file, by convention, ends with the .h extension. It is inserted inside a program by coding the #include preprocessor directive. The program will use various header files in order to perform basic input output tasks and mathematical operations as well. The header files that will be used include:

- stdio.h
- conio.h
- math.h

4.2 Functions

Functions will be an essential part for the development of a scientific calculator in C. The use of functions helps to decrease the repeatability of code. Functions will allow us to define reusable blocks of code that can be used whenever a specific task needs to be performed. This will improve code readability as well as efficiency.

4.3 Conditional statements

Conditional statements will be employed throughout the program. It will serve for basic arithmetic operations, error handling and for advanced mathematical computations:

i. **Basic arithmetic operations**: Conditional statements will be used to determine the type of operation the user wants to perform (e.g., addition, subtraction, multiplication etc.). The program will then select the appropriate operation based on the input given by user.

ii. **Error handling**: Conditional statements will effectively handle the errors in the following cases:
   - Division: It will effectively prevent division by zero and display a divide by zero error message to the user.
   - Matrix: Handling the errors that may arise in matrix operations when the number of columns and rows don’t meet the requirement for the operation to be performed.

iii. **Advanced computations**: Conditional statements will be useful in advanced mathematical operations such as:
   - Quadratic equations: Determining the type of roots the equation will have i.e., identical, real or imaginary.
   - Matrix: Checking the matrix’s dimension for given operation and ensuring that the calculation is possible or not.

5. SCOPE AND APPLICATIONS

The calculator will support from fundamental operations, error handling and advanced computations. Specifically, the program will be able to perform matrix, vector operations, logarithmic operations efficiently all while ensuring proper error handling. However, the scope of the calculator is limited to the operation of only two numbers at a time as the calculator won’t have option for continuous arithmetic calculations. Equation solver will only be limited to third-degree polynomial equations and won’t have a option for higher degree operations.

The calculator will support broad range of applications making it a powerful and versatile tool for basic as well as advanced mathematical calculations. It will effectively perform all the fundamental operations i.e., addition, subtraction, multiplication and divisional on with advanced features like log , $e\ , a\ , \sqrt[2]{\ }, \sqrt[3]{\ }, \sqrt{x}, x!$, permutation and combination. It will have the option for trigonometric calculations and base conversion as well. Beyond these fundamental capabilities the calculator will also be able to perform Matrix operations, vector operations and equation solving. The combination of all these features will make this calculator a powerful tool for students, professional or anyone who needs to perform calculations with ease.

6. TIME ESTIMATION

	Feb 1-2	Feb 2-4	Feb 4-12	Feb 12-16	Feb 16-18	Feb 18-20
Project completion and review						▨
Final testing and debugging					▨	
Documentation and cod comments				▨		
Testing and debugging				▨		
Implementing advance features			▨			
Implementing basic operator			▨			
Designing calculator logic		▨				
Setting up development environment	▨					
Requirement Analysis and planning	▨					

Table 6.1: Time Estimation Gantt Chart

7. FEASIBILITY ANALYSIS

7.1Technical Feasibility

The calculator project is technically possible because it is based on programming languages, which is effective and widely used for such applications. Important factors include: Basic arithmetic operations (also subtraction, multiplication, division) can be used using basic C programming structures. Advanced features such as trigonometric calculations, logarithms and experiences can be handled using <math.h> library.The project requires only a basic compiler such as GCC, which is freely available.

7.2 Economic Feasibility

The project includes no major financial investments, as it is completely software -based. GCC, Code :: Block or Dev-C ++ can be developed using free and open source tools. No external licenses or third -party services are required, making it a cost -effective solution.

7.3 Schedule Feasibility

Implementation of a basic calculator with core arithmetic functions can be completed within a few days. Adding scientific functions and other enhancements may require additional time for development and testing. Adequate time should be allocated for debugging and testing to ensure accuracy and reliability.

REFERENCES

[1] Wikipedia, "Calculator" October 2011.

[2] W3schools,"C Introduction".

[3] Researchgate, " Scientific Calculators and the Skill of Efficient Computation" January 2012.

[4] Wikipedia, "Software Based Calculator" March 2015.

[5] Medium," Command-line calculator" September 2024