

**TRIBHUVAN UNIVERSITY**

**Institute of Engineering**

Central Campus, Pulchowk

**Object-Oriented Programming Project Proposal:**

All-in-One Engineering Calculator

**Submitted To:**

Department of Electronics and Computer Engineering

**Submitted By:**

Pratik Luitel (073BEX426)

Rajil Bajracharya (073BEX430)

Stuti Kafle (073BEX445)

**Submitted On:**

11-03-018

**Acknowledgement**

First of all, we would like to thank our respected lecturer Dr. Basanta Joshi and lab teachers for providing required knowledge and guidance for the project. We would also like to thank our Electronics and Computer Department for providing this opportunity to express our theoretical knowledge into practical.

We would also like to thank our seniors for their prompt help, guidance and encouraging advice in the project. We are really grateful to them. Finally, we would like to thank our beloved friends for their help during the completion of the project.

**Abstract**

Contents

[1. Introduction: 4](#_Toc504074547)

[2. Objectives: 4](#_Toc504074548)

[3. Existing Systems: 4](#_Toc504074549)

[4. Proposed System: 5](#_Toc504074550)

[4.1 Description 5](#_Toc504074551)

[4.2 Block Diagram 6](#_Toc504074552)

[5. Methodology: 7](#_Toc504074553)

[5.1 Study of Source Material: 7](#_Toc504074554)

[5.2 Problem Analysis: 7](#_Toc504074555)

[5.3 Coding: 7](#_Toc504074556)

[5.4 Debugging and Testing: 7](#_Toc504074557)

[5.5 Documentation: 8](#_Toc504074558)

[6. Project Scope: 8](#_Toc504074559)

[7. Project Schedule: (estimated project duration: 26 days) 8](#_Toc504074560)

1. Introduction:

A calculator is a device, handheld or otherwise, that can perform mathematical operations automatically based upon inputs that may consist of letters, numbers, constants and/or symbols. This project is concerned with the design of a software-based calculator with advanced engineering capabilities along with the traditional capabilities of a common scientific calculator. These features include advanced matrix operations, coordinate transform, number system transform, differential equation solving, simultaneous equation solving up to 4 variables, among others.

1. Objectives:

* To develop and implement an all-in-one engineering calculator.
* To familiarize ourselves with various features of the C++ programming language.
* To understand the object-oriented approach to programming and problem solving.
* To get a first-hand experience on project development in a team.

1. Existing Systems:

Following are the most popular calculators in existence and in relevance to our current project:

* Scientific Calculator(Software-based)
* Graphing Calculator
* Programmable Calculator
* Specialized problem solving calculator such as Sudoku solver
* Formula Calculator
* Spreadsheet-based Calculator
* Non-numerical Calculators such as formula weight calculator

1. Methodology:

The execution of the above proposed project shall be carried out in the following major steps:

* 1. Study of Source Material:

In this phase, all available data was collected in relevance to this project and the necessary documentation was studied. Moreover, the necessary syntax, header files and programming techniques were also thoroughly studied and reviewed. In regard to the mathematical operations, necessary literature was also referred extensively

In this particular project, what we did for the source material was a thorough study of major C++ practices, as well as a study of the Object Oriented Approach to programming , where we thoroughly studied the literature given in the references of this report.

Literature review was also done for the method to analyze matrices as well as equations with up to four variables by using the matrix method, which was obtained from the basic study of mathematics literature.

In addition to that, much information was obtained from the internet as well.

* 1. Problem Analysis:

In this phase, the necessary logic for the development of the source code was devised. The related constraints were identified and the programming logic was tallied with the project objectives. Here, we broke the project down into basically two major parts – the console – where the user inputted a mathematical expression via his keyboard, and the Engg Mode, where we initially planned to include many features, including matrix modes and equations. The matrix mode- for analysis of matrices upto the order of 4\*4, and the equation mode –for solving linear equations of upto 4 variables were successfully completed later down the line, but unfortunately due to certain constraints, things such as derivatives and integration – which had been initially scheduled for completion were not included.

* 1. Coding:

In this phase, the actual coding of the program was and completed in a suitable IDE – which in this case was Code::Blocks – we chose it because it was relatively lightweight and fulfilled all the requirements we had the compiler was GNU GCC compiler. Most of the time of the project was allocated for this phase of the project as it determined the output of the project.

* 1. Debugging and Testing:

In this phase, the code developed in step 4.3 was checked for any error in the logic and any formatting issues or otherwise, and was corrected and modified. The program was then provided a carefully designed data set to see if it is able to produce the correct result in as fast a manner as is expected of it. Based on the results, the algorithm and hence, the code was also be improved upon. This is a perpetual phase, as the code was carefully analyzed as it was being written and the output was also checked for errors right on the fly. Also, the debugging phase will not be completed even with the completion of the project itself, as the project will be enhanced, adding new features as new ideas arise, which will cause even more complications on the way and new debugging methods will be applied. The testing will also need to be carried on for as long as the code addition and feature enhancement continues in the project.

1. Implementation

# **Block Diagram**

**Y**

**Input Module**

**Input Parsing Algorithm**

**Output Module**

**Operation Module Selection**

**Operation Module Execution**

**More Input?**

**N**

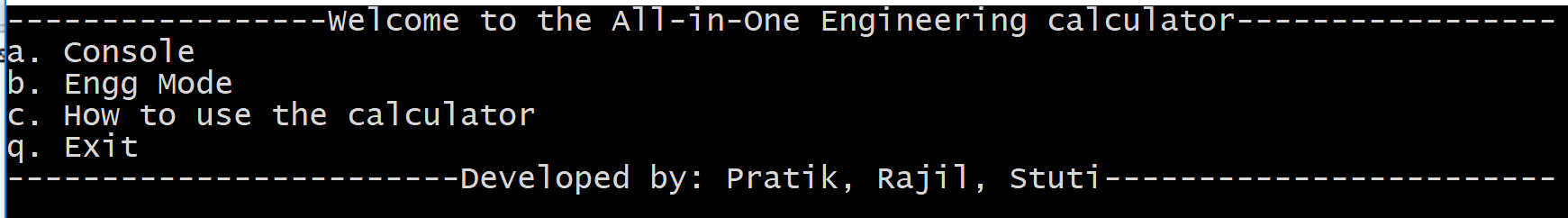
Figure 1 Project Block Diagram (for Console)

**(INSERT THE MATRIX RA EQUATION KO FLOWCHART HERE)**

**Working Principle:**

In this project, we divide the working of it into several parts. The first part of it, which is entered once the program is started, is known as the main menu of the program, where various options can be accessed.

1. **Main Menu**

 Figure 2: The Main Menu.

You might observe that we implemented the whole program as a console application, without using any kind of graphics library. This is because we focused more on the actual OOP aspect of the program and tried to run it without actually downloading any external libraries, with whatever tools we had available from the original C++ itself. This also allowed us headroom to put graphics in it in the future as an additional feature.   
The main menu has four options and enters each mode as the key corresponding to it is entered via the keyboard. This was relatively simple to implement. We made a class called wallOfText that had functions to display all menu texts, and take input from the user, and instantiated an object ‘main’ of this class and accessed the particular function via the main() function itself. This menu() function in the wallOfText class had the code to display the text and also the code to read input from the user and call the respective menu() functions from objects instantiated for each of the menu options. i.e. console and engg classes were made for console and engg mode respectively, whose objects were instantiated as private data members of the wallOfText class. The option ‘c’ displayed instructions via the instructions() function in the same class, that also asked for an input and if the input was ‘q’, it came back to the main menu. This theme of ‘q’ quitting to the previous menu is repeated all over the program.

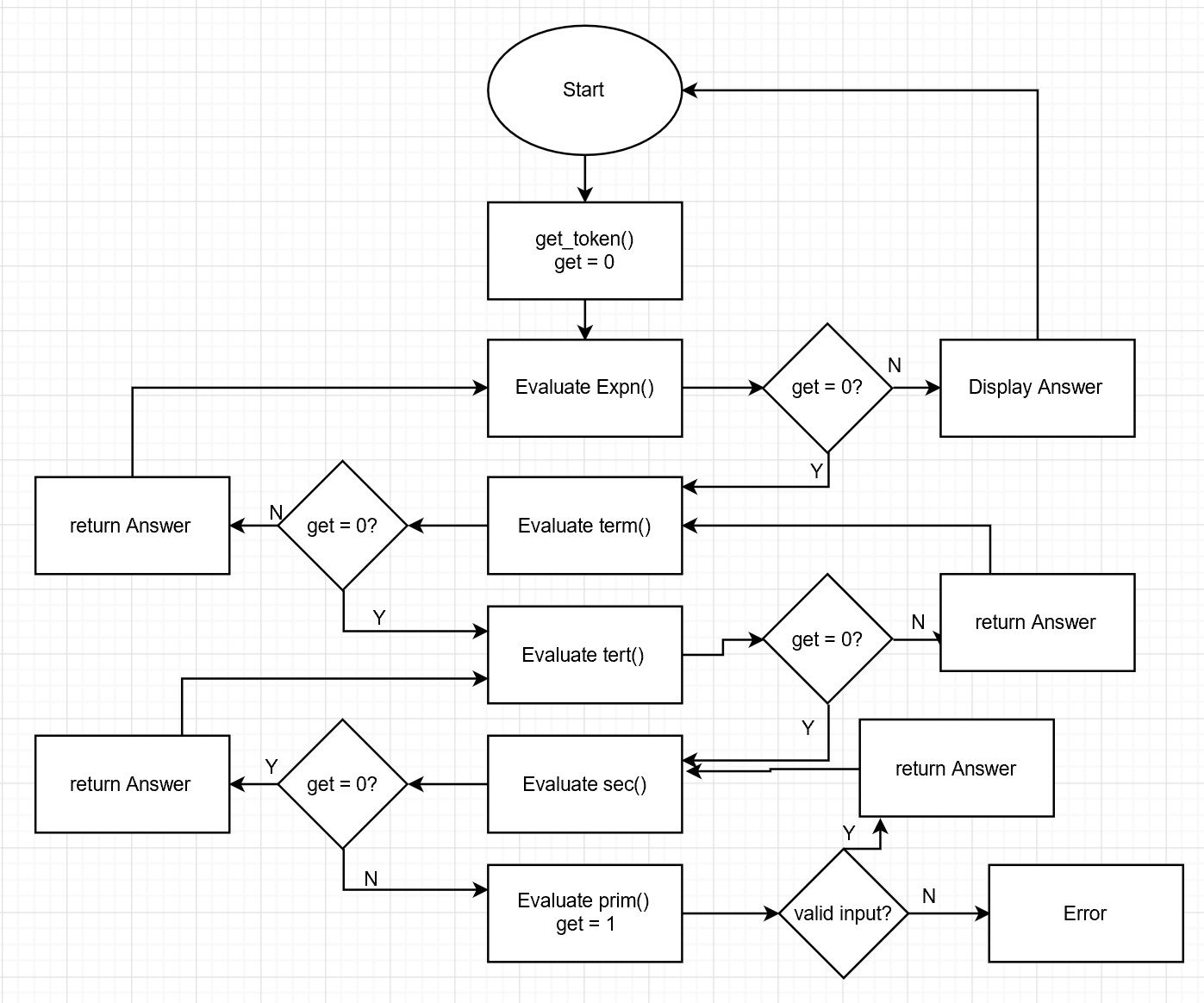
1. **Console** Figure 3: The Console Window

Pressing ‘a’ in the main menu enters the console, which is the main hub for doing the major arithematic operations in the program. Here, we have implemented the basic operations – add subtract, multiply and divide, along with other useful operations of trigonometric functions – sin, cos, tan, their inverses, exponentials, logarithms(both natural as well as base 10), and even an operator to calculate equivalent resistance of the parallel resistance ( | operator in this case).   
The basic principle on which the console works is string parsing and a grammar we implemented to set the precedence of operators in different layers.

**The get\_token() function:**

The get\_token() function does the bulk of the work in the console. The user inputs the expression for which they want the answer to; and the individual characters of the input act as ‘tokens’ that need to be interpreted before the particular action can be carried out. For this, the istream functions cin.get() and cin.putback() are used. The cin.get() function extracts an individual character from the stream (which is where all the keyboard inputs are stored. The function get\_token() serves the purpose of interpreting the latest character extracted from the stream by cin.get() into a variable ‘ch’. Based on what type it is: i.e. if it is a number, a print character( enter, =, or ;), a END character (assigned to ‘q’ here), an operation character (+,-,/,\*,|,^), or a string (designed to interpret functions like trigonometric and exponential functions), or an unrecognized character (error), it returns different values of the current\_token variable, which represents the latest character typed into the console window by the user. E.g. if the latest character is 3, then it returns NUMBER, which is the member of the enumeration token\_value (The enumeration contains representations for all the valid inputs).  
For the arithmetic symbols, it assigns the value of the typed characters to the variable ‘ch’, before returning, while for numbers, cin.putback(ch), is used, to put the character (now a number) back into the steam, so that it can be extracted later. This is done because numbers can have more than one character – which means that we can extract numbers only after another symbol is pressed.  
After get\_token() is done, there comes another vital part of the implementation, which is the application of BODMAS and other mathematical rules for precedence into the program, initiated by the expn() function.

**Implementation of grammar through expn(), term(), tert(), sec(), and prim():**

  
 *Figure 4: Flowchart for console*

First, the control goes to the function **exprn().** Boolean Variable **‘**get’ in this function denotes if it is the first token typed, and on the first call of exprn() via the open() function, is initialized to zero. If the get is zero, by successive calls of function i.e. (expn() calls term(),term calls tert(), tert() calls sec(), and sec() calls prim()), the control goes to the function prim(), that analyzes a single token itself.

**Prim():**The prim() function analyzes what we define as a primary – which are a number – a unary minus, brackets enclosing an expression and strings representing - trigonometric, exponential and logarithmic functions. (sin, cos, tan, asn, acs, atn, log, lnt, exp). These operations handled by the prim() can be thought of as operators having the highest precedence. If anything other than these are entered, it also displays error and returns user back to console. It recognizes the token, and also in case of numbers gets more tokens as numbers can be more than one character long stores the values as well.   
It then returns the obtained value to be handled by the sec() or secondary function. In case of brackets, it calls expn() function once again to evaluate expression, but gives it a parameter of true as it is the the first token to be analyzed. Also, if it doesnot get a closing bracket after opening bracket is typed, it displays error message.

**Sec():**

The sec() or the secondary handles the operation of power ^, which has lower precedence than the operations handled by the primary, but higher than that handled by tert(). The value returned by prim() is stored in a variable named left, and if operator ^ is found, the power operation is carried out and value is returned to the tert(), else the value is directly returned without carrying out any operation.

**Tert()**

The tert() or the tertiary handles the operation of parallel resistance |, which has lower precedence than the operations handled by the secondary, but higher than that handled by term(). The value returned by sec() is stored in a variable named left, and if operator | is found, the corresponding operation is carried out and value is returned to the term(), else the value is directly returned without carrying out any operation.

**Term()**

The term() or handles the operations of multiplication and division, both of equal precedence, which has lower precedence than the operations handled by the tert, but higher than that handled by expn(). The value returned by tert() is stored in a variable named left, and if operators \* or / are found, the corresponding operation is carried out and value is returned to the expn(), else the value is directly returned without carrying out any operation.

**Expn()**

The expn() or handles the operations of addition and subtraction, both of equal precedence, which has lower precedence than the operations handled by the term(),which is the lowest precedence. The value returned by term() is stored in a variable named left, and if operators + or - are found, the corresponding operation is carried out and value is returned to the open() function where the value is printed as the final answer. If + or – are not found, the value is directly returned and printed as the output.

1. **Engg Mode**

**(INSERT THE MATRIX RA EQUATION KO EXPLANATION HERE)**

1. Applications:

Our program endeavours to solve engineering-related problems, so it can be used by the engineering students a fast problem-solving tool. Some engineering mathematical problems are difficult and would take pages (if not ages) to solve, in such a case, it can be done easily with the help of the calculator. Especially problems involving very long and complex mathematical expressions (where the console of our program can be used), and solving equations of four variables and matrices of higher order (which are not found in the normal everyday calculator, where our matrix mode and eqn mode can be used) can be done easily using this program. Not only in engineering it can also be used by the account students as there are mathematical functions included in it. Most important of all, it saves time and gives required results correctly and efficiently.

1. Literature Survey
2. Results
3. Problems faced and solutions
4. Limitations and future enhancements  
   The prospects of this project aren’t as narrow as those of any other project, because the world of mathematics and engineering are very wide and we can continue to add new features and will not run out. The future enhancements may e the addition of several other features that might be useful to engineers as well as the general public alike. Calculations for higher degree differential equations as well as integrals spring to mind. Similarly, computation of several complex entities such as laplace transforms as well as fourier series of functions can also be done.   
   Also, more can be added to the Aesthetics department as well. There is plenty of headroom for the addition of a Graphical User interface without losing the easy and user friendly console. The overall look of the app can be made more attractive and also other functions as mentioned above can be fitted into the project in the future as well.
5. Conclusions and recommendations  
   Hence, the project was completed. A fully fledged calculator was created using the object oriented approach in the C++ languages, having a fully working console mode, and an engg mode consisting of problem solving for matrices as well as a system of linear equations of upto four variables.  
   There are many recommendations that have to be given to improve this project.
6. References  
   1. Bjarne Stroustrup, Programming Principles and Practices using C++  
   2. The Secrets of Object Oriented Programming in C++, Daya Sagar Baral, Diwakar Baral  
   3. Object Oriented Programming with C++, E Balaguruswamy  
   4.