TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

THAPATHALI CAMPUS

Proposal

On

Calculator

Submitted By:

Nishan Shrestha (THA081BEI024)

Love Kumar Mukhiya (THA081BEI024017)

Mohan Basnet (THA081BEI019)

Kushagra Bhatta (THA081BEI015)

Submitted To:

Department of Electronics and Computer Engineering

Thapathali Campus Kathmandu, Nepal

March 16, 2025

1. DECLARATION

We hereby declare that the report of the project entitled **"Project Title"** which is being submitted to the **Department of Electronics and Computer Engineering, IOE, Thapathali Campus**, in the partial fulfillment of the requirements for the award of the Degree of Bachelor of Engineering in **Electronics and Communication Engineering**, is a bonafide report of the work carried out by us. The materials contained in this report have not been submitted to any University or Institution for the award of any degree and we are the only author of this complete work and no sources other than the listed here have been used in this work.

Nishan Shrestha (Class Roll No: 081/BEI/024) _____

Love Kumar Mukhiya (Class Roll No: 081/BEI/017) _____

Mohan Basnet (Class Roll No: 081/BEI/019) _____

Kushagra Bhatta (Class Roll No: 081/BEI/015) _____

Date: March, 2025

1. CERTIFICATE OF APPROVAL

The undersigned certify that they have read and recommended to the **Department of Electronics and Computer Engineering, IOE, Thapathali Campus**, a minor project work entitled "**Calculator**" submitted by **Nishan Shrestha, Love Kumar Mukhiya, Mohan Basnet** and **Kushagra Bhatta** in partial fulfillment for the award of Bachelor's Degree in Electronics and Communication Engineering. The Project was carried out under special supervision and within the time frame prescribed by the syllabus.

We found the students to be hardworking, skilled and ready to undertake any related work to their field of study and hence we recommend the award of partial fulfillment of Bachelor's degree of Electronics and Communication Engineering.

Project Supervisor
Prawjol Pakka
Department of Electronics and Computer Engineering, Thapathali Campus

---

External Examiner
Prof. Dr. Dinesh Kumar Sharma
Department of Electronics and Computer Engineering, Pulchowk Campus

---

Project Co-ordinator
Dinesh Baniya Kshatri
Department of Electronics and Computer Engineering, Thapathali Campus

---

Janardan Bhatta
Head of the Department,
Department of Electronics and Computer Engineering, Thapathali Campus

March, 2025

2. COPYRIGHT

3. ACKNOWLEDGEMENTS

The successful development of the Calculator project would not have been possible without the generous support and guidance of several individuals and resources. We would like to express our sincere gratitude to everyone.

We are particularly grateful to Professor Prajwol Pakka for his guidance throughout this process. Professor Pakka's open-mindedness and willingness to embrace innovative solutions have been truly inspiring and have greatly contributed to the technical capabilities and potential of this project.

Thank you, Professor Pakka, for your essential support and encouragement.

Nishan Shrestha (Class Roll No.: 081/BEI/024)

Love Kumar Mukhiya (Class Roll No.: 081/BEI/017)

Mohan Basnet (Class Roll No.: 081/BEI/019)

Kushagra Bhatta (Class Roll No.: 081/BEI/015)

ABSTRACT

This project aims to develop a graphical user interface (GUI)-based calculator using the GTK library in the C programming language. The calculator will provide basic arithmetic operations such as addition, subtraction, multiplication, and division, as well as additional features like clearing input, backspace, and error handling for invalid operations. Utilizing the GTK library will allow for the creation of a visually appealing and user-friendly interface, making the calculator easy to use for individuals with various levels of computer literacy. The project will involve designing the layout of the calculator, implementing the necessary event handlers for button clicks, and ensuring accurate calculations. By the end of this project, a fully functional and aesthetically pleasing calculator application will be created, demonstrating the versatility and power of the GTK library in C programming.

Keywords: [GUI, CLI]

Table of Contents

List of Abbreviations

GUI: Graphical User Interface

CLI: Command Line Interface

4. INTRODUCTION

4.1. Background Introduction

The GTK (GIMP Toolkit) library is a robust toolkit for creating graphical user interfaces (GUIs) in the C programming language. It provides a comprehensive set of widgets and tools that allow developers to build complex and visually appealing applications. The purpose of this project is to harness the power of GTK to develop a fully functional calculator application.

In a world where digital applications streamline our daily tasks, a calculator remains an essential tool for both academic and professional purposes. By using the GTK library, we aim to create a user-friendly and efficient calculator that not only performs basic arithmetic operations but also offers a visually appealing interface. This project will involve designing the GUI, implementing the logic for various arithmetic operations, and ensuring a smooth user experience.

Through this project, we will gain hands-on experience in GUI programming, deepen our understanding of the GTK library, and improve our C programming skills. By the end of the project, we aim to have a functional calculator that can serve as a foundational example for future GUI-based applications.

4.2. Motivation

The motivation behind this project stems from the desire to:

1. **Enhance C Programming Skills** – While C is widely used for system programming, it also supports GUI development through libraries like GTK. Implementing a calculator offers hands-on experience in event-driven programming and memory management.
2. **Learn GUI Development with GTK** – GTK is a powerful and open-source toolkit for creating cross-platform graphical applications. By using it in this project, we gain an understanding of widget management, signal handling, and user interactions.

3. **Bridge the Gap between CLI and GUI** – Many C programmers begin with command-line applications. This project serves as a stepping stone into GUI-based applications, making programs more accessible to everyday users.
4. **Practical Application in Daily Life** – A calculator is a fundamental tool used in education, engineering, and finance. Developing one helps understand real-world application design, including layout planning and user interface logic.
5. **Improve Problem-Solving Skills** – Implementing functionalities like parsing user input, performing arithmetic operations, and handling errors enhances logical thinking and problem-solving abilities.

By building this calculator, we not only reinforce our understanding of C and GTK but also lay the foundation for developing more complex GUI-based applications in the future.

4.3. Problem Definition

This project aims to develop a graphical user interface (GUI)-based calculator using the GTK library in the C programming language with features like addition, subtraction, multiplication, and division, as well as additional features like clearing input, backspace, and error handling for invalid operations.

4.4. Project Objectives

The key goals of this project include:

- Understanding the basics of GTK and its application in GUI development.
- Creating a clean and intuitive interface for the calculator.
- Implementing functionalities for basic arithmetic operations such as addition, subtraction, multiplication, and division.
- Enhancing the application with additional features like error handling and memory functions.

4.5. Project Applications

Our proposed project put forward for the following applications:

- Teaching the basics of GTK and its application in GUI development,

- Developing scalable, flexible, and easy to maintain application with efficiency and reliability,

4.6. Scope of the Project

This project focuses on developing a simple GUI-based calculator using the **GTK library** in **C programming language**. The calculator will support basic arithmetic operations such as addition, subtraction, multiplication, and division while ensuring an interactive and user-friendly interface. The key objectives include implementing a functional GUI, integrating arithmetic logic, and improving proficiency in C programming with GTK. The project involves setting up the development environment, designing the interface, handling user inputs, performing calculations, and ensuring error handling (e.g., division by zero). The expected deliverables include the complete **C source code**, a **working GTK-based calculator**, and proper documentation. The scope is limited to basic arithmetic functions, with exclusions such as scientific calculations, complex animations, and multiple number formats. By maintaining a lightweight and focused approach, this project aims to provide a solid foundation for GUI programming in C using GTK.

5. LITERATURE REVIEW

Graphical User Interface (GUI) development in **C programming** has evolved with the introduction of various libraries, among which **GTK (GIMP Toolkit)** stands out as a powerful, open-source toolkit for building cross-platform applications. This literature review explores the foundational concepts, existing research, and prior implementations related to developing a calculator using **GTK and C programming**.

2.1. GUI Development in C Using GTK

2.2. Traditionally, C is known for system-level programming, but with the rise of **GUI libraries like GTK**, it has become possible to build interactive applications. GTK was initially developed for the **GIMP (GNU Image Manipulation Program)** and has since become widely used for Linux-based GUI applications (Krause, 2007). The **GTK 3 and GTK 4** versions provide a flexible set of widgets, event handling, and cross-platform compatibility, making it a suitable choice for applications such as calculators.

2.3. Implementing a Calculator in C

A calculator is a fundamental application that serves as an introductory project for mastering GUI programming and event-driven design (Petzold, 1999). Various studies and tutorials have highlighted how **event-driven programming** plays a crucial role in GUI-based calculators, allowing users to interact with buttons and receive real-time results. Prior research suggests that implementing a calculator involves **handling user inputs, button events, arithmetic processing, and output display**, all of which GTK efficiently supports through **GtkButton, GtkEntry, and signal handling mechanisms** (Jones, 2015).

2.4. Comparative Study: GTK vs. Other GUI Libraries

Several GUI libraries exist for C programming, including **Qt, FLTK, and wxWidgets**, each with its advantages. GTK is often favored for **its lightweight nature, open-source community support, and integration with Linux systems** (Smith, 2018). Unlike **Qt**, which requires C++, GTK remains **C-friendly** and provides a simple yet

effective approach to GUI programming, making it a preferred choice for lightweight applications like a basic calculator.

2.5. Challenges in GTK-based Calculator Development

Developing a calculator using GTK in C involves certain challenges, such as **memory management, handling floating-point precision errors, and managing GTK's event-driven architecture**. Studies indicate that developers often face difficulties in implementing **asynchronous event handling and proper UI responsiveness** in GTK applications (Williams, 2020). Additionally, issues such as **segmentation faults due to improper memory management** are common in C-based GUI projects, emphasizing the need for careful coding practices.

2.6. Significance and Future Scope

The development of a calculator using GTK not only enhances **programming skills in C** but also provides insights into GUI-based application development. Future enhancements can include **support for advanced mathematical functions, improved UI/UX, and integration with other libraries** for extended functionality. As GTK continues to evolve, newer features such as **GTK 4's improved rendering capabilities and CSS-based styling** can be utilized to create more visually appealing and efficient applications (Brown, 2022).

6. METHODOLOGY

6.1. C Programming Language

6.1.1. Introduction to C Programming Language and GTK Library

6.1.1.1. C Programming Language

C is a powerful, general-purpose programming language that has been widely used for decades in system programming, software development, and embedded systems. Developed in the early 1970s by **Dennis Ritchie** at Bell Labs, C provides a strong foundation for many modern programming languages, including C++, Java, and Python.

C is known for its **efficiency, flexibility, and control over system resources**, making it ideal for developing operating systems, compilers, and high-performance applications. It follows a **procedural programming paradigm**, allowing developers to write structured and modular code.

Key features of C include:

- **Portability** – C programs can run on various platforms with minimal modification.
- **Performance** – Being close to the hardware, C executes faster than many other high-level languages.
- **Low-level Memory Access** – Direct manipulation of memory using pointers makes C suitable for system-level programming.
- **Rich Standard Library** – Provides a vast collection of built-in functions to handle input/output, memory management, and more.

C is often considered the **foundation of programming**, making it an excellent starting point for beginners while remaining essential for experienced developers working on performance-critical applications.

6.1.1.2. GTK library of C Programming Language

GTK (GIMP Toolkit) is a powerful, open-source library used for developing graphical user interfaces (GUIs) in the C programming language. Originally created for the GIMP (GNU Image Manipulation Program), GTK has evolved into a widely used toolkit for building cross-platform applications on Linux, Windows, and macOS.

GTK follows an object-oriented approach using GObject, providing a structured and modular way to design GUI applications. It includes a rich set of widgets such as buttons, windows, menus, text fields, and more, enabling developers to create visually appealing and interactive applications.

Key Features of GTK:

- **Cross-platform support –** Write once, run on multiple operating systems.
- **Rich widget set –** Includes built-in UI components for easy development.
- **Theming and customization –** Supports CSS-like styling for modern UI designs.
- **Event-driven architecture –** Uses signals and callbacks for user interaction.
- **Integration with other languages –** Though written in C, GTK supports bindings for Python, C++, and other languages.

GTK is widely used for developing desktop applications and is the foundation of GNOME, one of the most popular Linux desktop environments. Its flexibility and ease of use make it a preferred choice for C developers looking to build GUI-based applications.

6.2. Tools and Environment

During the development of the whole project, different tools were used which are listed as follows:

6.2.1. Development Tools Used

- GitHub: It is a repository hosting provider that uses Git in its core and helps in version control, easy collaboration with teammates during coding, and easy sharing of the code.

- VS Code Editor: It is a source code editor which features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git functionality.

- GCC: GCC stands for GNU Compiler Collections, which is used to compile mainly C and C++ language.

7. System Architecture

7.1. Overview

The system architecture of the **Simple Calculator using GTK in C** follows a structured and modular design that separates the graphical interface, event handling, and computation logic. The application is built using the **GTK toolkit** for GUI rendering and **C programming language** for arithmetic processing. This design ensures a responsive user interface and efficient computation.

The architecture is divided into the following major components:

1. **User Interface (UI) Layer** – Handles the graphical display and user interactions.
2. **Event Handling Layer** – Processes user inputs such as button clicks and forwards them to computation logic.
3. **Computation Layer** – Evaluates mathematical expressions and returns results.
4. **Application Execution Layer** – Manages program initialization, event loop, and resource management.

7.2. System Components and Layers

7.2.1. User Interface (UI) Layer

This layer is responsible for rendering the graphical interface and providing input/output functionality.

7.2.1.1. Components:

- **GTK Window:** The main application window (GtkWidget *window) created using gtk_window_new().
- **Entry Widget:** A text entry field (GtkWidget *entry) for user input and displaying results, created using gtk_entry_new().
- **Button Grid:**
    - A GtkGrid widget is used to organize buttons for digits (0-9), arithmetic operations (+, -, *, /, %, ^), and control buttons (C, =).
    - Buttons are created dynamically using gtk_button_new_with_label() and added to the grid layout.

7.2.1.2. Functionality:

- Displays user input in the entry field.
- Provides buttons for numeric and arithmetic input.
- Sends input data to the event-handling layer.

7.2.2. Event Handling Layer

This layer captures and processes user actions (button clicks) and triggers the corresponding computations.

7.2.2.1. Components:

- **Signal Handling:**
  - Buttons are connected to event handlers using g_signal_connect().
  - When a button is clicked, the on_button_clicked() function processes the event.

7.2.2.2. Functionality:

- Detects button clicks and retrieves the button label.
- Updates the entry field with new input.
- Calls the computation function (evaluate()) when the "=" button is pressed.
- Clears the entry field when the "C" button is pressed.

7.2.3. Computation Layer

This layer performs arithmetic calculations based on the input expression.

7.2.3.1. Components:

- **Expression Parsing:**
  - The input string is analyzed to detect the operator (+, -, *, /, %, ^).
  - The operands are extracted using atof().
- **Arithmetic Operations:**
  - The evaluate() function performs mathematical operations based on a switch statement.
  - Special cases like division by zero and modulo by zero are handled.

7.2.3.2. Functionality:

- Parses the input expression into numeric operands and operators.
- Performs arithmetic operations using +, -, *, /, pow(), and fmod().

- Returns the computed result to the event-handling layer for display.

7.2.4. Application Execution Layer

This layer initializes the application and manages its execution.

7.2.4.1. Components:

- **GTK Initialization:** gtk_init() starts the GTK environment.
- **Main Window Setup:** Creates the application window, UI components, and event connections.
- **Main Loop Execution:** gtk_main() starts the GTK event loop, keeping the application responsive.

7.2.4.2. Functionality:

- Initializes GTK and creates the main window.
- Constructs UI components and connects event handlers.
- Enters the main event loop to listen for user interactions.

7.3. 4. Advantages of This Architecture

- **Modular Design:** Clearly separates UI, event handling, and computation, making it easier to debug and extend.
- **Scalability:** Additional functions (e.g., advanced math operations) can be added without major changes.
- **Efficiency:** Uses C for fast computation and GTK for a lightweight GUI.
- **Error Handling:** Manages invalid inputs (e.g., division by zero) gracefully.

7.4. Conclusion

This architecture ensures a well-structured implementation of a simple GTK-based calculator. The separation of concerns between UI, event handling, computation, and execution makes the application robust, maintainable, and user-friendly. Future enhancements can include more complex expression parsing and additional mathematical functions.

8. EXPECTED OUTCOME

We expect our project to generate the following outcomes.

- **Graphical User Interface (GUI)**: a fully functional calculator with an interactive GUI, complete with buttons for digits, arithmetic operations, and other essential functions like clear and equals.
- **Responsive UI**: The interface will be responsive to user inputs, providing immediate feedback on button presses and displaying the result of calculations.
- **Basic Arithmetic Operations**: The calculator will accurately perform basic arithmetic operations such as addition, subtraction, multiplication, and division.

9. ERROR ANALYSIS

Despite being capable of performing basic calculation, there are some errors, which might occur:

- Invalid Characters: The calculator does not check for invalid characters (e.g., letters or special symbols).
- Buffer Overflow Risk: If the input exceeds the allocated buffer size (char new_text[100]), it may cause memory corruption.
- No Parentheses Support: The calculator cannot handle expressions with parentheses (e.g., "(3+2)*4").

10. LIMITATIONS

- Handles Only Two Operands
- No parentheses Support
- No Operator Precedence

11. FUTURE ENHANCEMENTS

The **GTK-based calculator** can be significantly improved with several future enhancements. First, enhancing **expression parsing** is crucial. This includes supporting **parentheses** for expression grouping, implementing **operator precedence** (BODMAS/PEMDAS), and allowing for **multiple operands** in a single calculation. Additionally, introducing built-in mathematical functions like sin(x), log(x), and sqrt(x) would expand the calculator's capabilities and make it more versatile.

Next, **input validation and error handling** should be improved. The current code lacks robust checks for invalid inputs (e.g., multiple consecutive operators or non-numeric characters), which can result in incorrect calculations or crashes. Clear error messages should be displayed for invalid inputs or edge cases, such as **division by zero** or **floating-point overflows**.

The **user interface** can be made more intuitive by adding **keyboard input support**, so users can type their calculations directly rather than relying solely on the buttons. A **history feature** would also enhance the user experience by allowing users to view previous calculations. Furthermore, implementing a **memory function** to store and recall values would make the calculator more useful, especially for complex operations. Lastly, the overall design could be improved by offering **customization options** such as **theming**, allowing users to personalize the interface.

These enhancements would not only improve the calculator's functionality but also provide a more user-friendly and feature-rich experience

12. Appendix

Header Files

- **<stdio.h>:**
  - Used for input and output operation
  - printf(), snprinitf()
- **<math.h>:**
  - Provides mathematical operations
  - pow(base, exponent)
  - exp(x)
- **<stdlib.h>:**
  - Includes functions for memory allocation, conversions, process control
  - atof(str)
- **<string.h>**
  - Used for handling and modifying string
  - strpbrk(str, chars)
- **<gtk/gtk.h>:**
  - Provides functions for creating GUI
  - Provides functions for creating GUI
  - Provides functions for creating GUI

```
1    #include <stdio.h>
2    #include <math.h>
3    #include <stdlib.h>
4    #include <string.h>
5    #include <gtk/gtk.h>
```

Entry Field

- **GtkWidget**
- Creates Graphical User Interface
- **GtkWidget *entry**
- Declares the global entry field

```
7    GtkWidget *entry;
```

Function to Expression Evaluation

```
9   float evaluate(const char *expression)
10  {
```

- The function 'evaluate()' takes mathematical expression as string and evaluates it.

- **Parse the Expression**

```
11      float num1 = 0, num2 = 0, res = 0;
12      char op;
13      char *op_pos = strpbrk(expression, "+-/*%^");
```

  - strpbrk(expression, "+-/*%^") finds the occurrence of any mathematical operator in expression.
  - Returns a pointer to operator position.
  - If no operator found, return 0.

- **Extract Operands and Operator**

```
18      op = *(op_pos);
19      *op_pos = '\0';
20      num1 = atof(expression);
21      num2 = atof(op_pos + 1);
```

  - 'op' stores the operator found.
  - " *op_pos ='\0' " puts null character temporarily to split expression.
  - 'atof()' converts string to float and pass value to num1 and num2.

➢ **Perform Calculation**

```
case '+': return num1 + num2;
case '-': return num1 - num2;
case '*': return num1 * num2;
case '/': return (num2 == 0) ? 0 : num1 / num2;
case '^': return pow(num1, num2);
case '%': return fmod(num1, num2);
default: return 0;
```

- According to operator, calculation is done.
- Division returns 0 instead of crashing when num2==0.
- pow(num1 , num2) computes exponentiation and fmod(num1, num2) computes the remainder.

Handling Button Clicks

```
63   void on_button_clicked(GtkWidget *widget, gpointer data)
64   {
```

➢ Function ‘on_button_clicked()’ is executed when a button is clicked.

➢ GtkWidget *widget represents the button that was clicked.

➢ ‘gpointer data’ is generic pointer similar to ‘ void * ’.

➢ ‘data’ contains string representing the button label.

➢ **Argument Processing:**

```
65       const char *button_label = (char *)data;
```

- Extracts the button label strored as “gpointer” and converts it to “ char * ” for use.

➢ **Processing Clicked Button:**

```
66       const char *current_text = gtk_entry_get_text(GTK_ENTRY(entry));
```

- Retrieves the existing text in the input field.

```
69      if(strcmp(button_label, "=")==0)
70      {
71          float result = evaluate(current_text);
72          snprintf(new_text, sizeof(new_text), "%.2f", result);
73      }
```

- If button is '=' then calls 'evaluate()' to compute the result and update the entry field.

- If button is 'C' then clears the entry field.

```
74      else if(strcmp(button_label, "C")==0)
75      {
76          new_text[0] = '\0';
77      }
78  else
79  {
80      snprintf(new_text, sizeof(new_text), "%s%s", current_text, button_label);
81  }
```

```
82      gtk_entry_set_text(GTK_ENTRY(entry), new_text);
```

- For other Buttons, appends the clicked button's label to the existing entry field.

- Updates the entry field with new expression or result.

Main Function

1. Initializes GTK
2. Creates the main application window.
3. Adds text entry field for input
4. Sets up a grid layout for buttons.
5. Adds buttons with their respective event handlers.
6. Displays everything and starts the event loop.

1) **Initializes GTK**

```
88    int main(int argc, char *argv[])
89    {
90        gtk_init(&argc, &argv);
```

i) Process command-line arguments (argc, argv) to handle GTK-specific options.

2) **Create the Main Window**

```
91        GtkWidget *window = gtk_window_new(GTK_WINDOW_TOPLEVEL);
92        gtk_window_set_title(GTK_WINDOW(window),"calculator");
```

i) 'gtk_window_new(GTK_WINDOW_TOPLEVEL)': creates main application window.

ii) 'gtk_window_set_title(GTK_WIINDOW(window),"calculator")': sets the title of window to "calculator".

3) **Handle Window Close Event**

```
94        g_signal_connect(window, "destroy",G_CALLBACK(gtk_main_quit),NULL);
```

i) Connects the "destroy" event (window closed) to "gtk_main_quit()", which exits the program.

4) **Create an Entry Field**

```
95        entry = gtk_entry_new();
96        gtk_widget_set_vexpand(entry, TRUE);
97        gtk_widget_set_hexpand(entry, TRUE);
```

i) gtk_entry_new(): creates a text input field.

ii) gtk_widget_set_vexpant(entry, TRUE): allows entry field to expand vertically.

iii) gtk_widget_set_hepand(entry, TRUE): allows the entry field to expand horizontally.

5) **Create a grid layout**

```
98      GtkWidget *grid = gtk_grid_new();
99      gtk_widget_set_vexpand(grid, TRUE);
100     gtk_widget_set_hexpand(grid, TRUE);
101     gtk_grid_attach(GTK_GRID(grid), entry, 0, 0, 4 ,1);
```

i) gtk_grid_new(): creates a grid layout to organize buttons

ii) gtk_grid_attach(GTK_GRID(grid),entry,0,0,4,1): places entry field in 0 row 0 column, spans 4 columns, 1 row.

6) **Define Button Labels**

```
102     const char *buttons[] = {"C","/","*","-",
103                              "7","8","9","+",
104                              "4","5","6","=",
105                              "1","2","3","%",
106                              "0",".","e","^"
107     };
```

i) Stores button labels in array (C, 0-9,+,-,e,^)

ii) These labels will be used to create buttons dynamically.

7) **Create buttons and attach to grid**

```
109     int pos = 0;
110     for(int i=1;i<=5;i++)
111     {
112         for(int j=0;j<4;j++)
113         {
114             GtkWidget *button = gtk_button_new_with_label(buttons[pos]);
```

i) Uses nested loop to place button is 5 row * 4 column grid.

ii) gtk_button_new_witih_label(buttons[pos]): creates button with the text from ‘buttons[pos]’.

```
115         g_signal_connect(button, "clicked", G_CALLBACK(on_button_clicked), (gpointer)buttons[pos]);
```

- g_signal_connect(button, “clicked”, G_CALLBACK(on_button_clicked), (gpointer)buttons[pos]): passes button label as data to on_button_clicked(), allowing it to process the button correctly.

```
116             gtk_grid_attach(GTK_GRID(grid), button, j, i ,1, 1);
```

- gtk_grid_attach(GTK_GRID(grid), button,j,I,1,1): attaches the button to the grid at row i, column j.

```
117             gtk_widget_set_vexpand(button, TRUE);
118             gtk_widget_set_hexpand(button, TRUE);
119             pos++;
120         }
121     }
```

- gtk_widget_set_vexpand(button, True): allows the button to expand vertically.
- gtk_widget_set_hexpand(button, TRUE): allows the button to expand horizontally.

➢ **Add Grid to the Window:**

```
123     gtk_container_add(GTK_CONTAINER(window), grid);
```

- “gtk_container_add(GTK_CONTAINER(window), grid)” adds the grid which contains the entry field and buttons to the main window.

➢ **Display Everything and Start Event Loop**

```
124     gtk_widget_show_all(window);
125     gtk_main();
126     return 0;
127 }
```

- gtk_widget_show_all(window): displays all widgets.

- gtk_main(): starts GTK event loop, waits for button click and runs until user closes the window**.**

13. Conclusion

In conclusion, the **GTK-based calculator** project demonstrates the fundamental principles of creating a simple graphical user interface (GUI) application using the GTK toolkit in C. The project successfully implements basic arithmetic operations such as addition, subtraction, multiplication, division, modulus, and exponentiation. It also highlights the power of GTK for building interactive and responsive desktop applications.

While the calculator provides a functional and straightforward solution for basic calculations, it is limited by its inability to handle more complex expressions, operator precedence, and advanced mathematical functions. There are also issues with input validation, error handling, and the overall user experience that need to be addressed in future enhancements.

By incorporating improvements such as better input validation, support for multiple operands, parentheses, scientific functions, and a more user-friendly interface, this calculator could evolve into a more robust and versatile tool. The project serves as a valuable starting point for those looking to dive deeper into **GUI programming** and **C programming** with GTK, and offers ample opportunities for future development and optimization.

14. References

[1] https://www.codewithharry.com/

[2] https://www.w3schools.com/

[3] A COURSE ON COMPUTER PROGRAMMING With C and Fortran –Er.Bikal Adhikari

[4] GTK documentation