**Explanation: Scientific Calculator GUI**

**Overview**

I have created a scientific calculator using GTK+3 in C. It handles both basic and advanced math operations using a graphical user interface (GUI) with GTK widgets.

**Code Breakdown**

**Includes & Global Variables**

#include <gtk/gtk.h>

#include <math.h>

#include <stdlib.h>

#include <string.h>

* gtk/gtk.h: Main header for GTK GUI.
* math.h: For mathematical functions like sin, cos, pow, etc.
* stdlib.h, string.h: Used for atof(), strncpy(), strcat(), etc.

#define HISTORY\_SIZE 10

double memory = 0.0;

char history[HISTORY\_SIZE][100];

int history\_index = 0;

* memory: Stores memory value for M+/M-/MR/MC.
* history: Array to store last 10 results.
* history\_index: Keeps track of history position.

**Function: add\_to\_history()**

void add\_to\_history(const char \*entry) {

strncpy(history[history\_index % HISTORY\_SIZE], entry, 99);

history\_index++;

}

* Adds new entry to history[], in a circular buffer way using modulo.

**Function: on\_operation\_button\_clicked()**

Triggered when any calculator operation button is clicked.

const gchar \*label = gtk\_button\_get\_label(GTK\_BUTTON(widget));

* Retrieves button label (e.g., "Add", "Sqrt").

GtkEntry \*\*entries = (GtkEntry \*\*)data;

* Casts received pointer to GtkEntry\*\*, accessing the input fields and result label.

const gchar \*num1\_text = gtk\_entry\_get\_text(entries[0]);

const gchar \*num2\_text = gtk\_entry\_get\_text(entries[1]);

* Reads user-entered strings from entry widgets.

double num1 = atof(num1\_text);

double num2 = atof(num2\_text);

* Converts string input to floating-point numbers.

**Handles operations like:**

else if (strcmp(label, "Add") == 0) result = num1 + num2;

// Same pattern used for Subtract, Multiply, Divide, Power...

**Special operations:**

* **Factorial**: Uses loop for num1!
* **nPr / nCr**: Uses formulae:
  + nPr = n! / (n-r)!
  + nCr = n! / (r! \* (n-r)!)
* **Deg↔Rad**: Converts between degrees and radians.
* **Trig / Log / Exp**: Uses math functions: sin, log10, exp, etc.
* **Memory (M+, M-, MR, MC)**: Adjusts or returns memory.

snprintf(result\_str, sizeof(result\_str), "Result: %.6lf", result);

gtk\_label\_set\_text(GTK\_LABEL(entries[2]), result\_str);

add\_to\_history(result\_str);

* Converts result to string, displays it in result label, and adds to history.

**Function: on\_history\_button\_clicked()**

GtkLabel \*label = GTK\_LABEL(data);

char history\_str[1024] = "History:\n";

* Creates a string that combines all previous results from history.

gtk\_label\_set\_text(label, history\_str);

* Displays the entire result history in the result label.

**GUI Creation in main()**

* **GTK Initialization**

gtk\_init(&argc, &argv);

* **Window Creation**

GtkWidget \*window = gtk\_window\_new(GTK\_WINDOW\_TOPLEVEL);

gtk\_window\_set\_title(GTK\_WINDOW(window), "Scientific Calculator");

gtk\_window\_set\_default\_size(GTK\_WINDOW(window), 500, 400);

* Creates the main app window.
* Sets window title and default size.

g\_signal\_connect(window, "destroy", G\_CALLBACK(gtk\_main\_quit), NULL);

* Closes app when window is closed.
* **Grid Layout**

GtkWidget \*grid = gtk\_grid\_new();

gtk\_container\_add(GTK\_CONTAINER(window), grid);

* A GtkGrid arranges widgets in rows & columns.
* **Input Fields & Result Label**

GtkWidget \*entry1 = gtk\_entry\_new();

GtkWidget \*entry2 = gtk\_entry\_new();

GtkWidget \*result\_label = gtk\_label\_new("Result:");

* GtkEntry: Input boxes for number 1 and number 2.
* GtkLabel: Displays output result.

gtk\_grid\_attach(GTK\_GRID(grid), gtk\_label\_new("Num 1:"), 0, 0, 1, 1);

gtk\_grid\_attach(GTK\_GRID(grid), entry1, 1, 0, 1, 1);

...

* Labels and entries placed into the grid layout.
* **Buttons Array**

const char \*buttons[] = { ... };

* All operation buttons are stored in a string array.

GtkEntry \*entries[] = { GTK\_ENTRY(entry1), GTK\_ENTRY(entry2), GTK\_ENTRY(result\_label) };

* Grouping entry1, entry2, and result label for passing into callbacks.

for (...) {

GtkWidget \*btn = gtk\_button\_new\_with\_label(buttons[i]);

gtk\_grid\_attach(GTK\_GRID(grid), btn, col, row, 1, 1);

g\_signal\_connect(btn, "clicked", G\_CALLBACK(on\_operation\_button\_clicked), entries);

}

* Each button is created, placed in the grid, and connected to the operation handler.
* **History Button**

GtkWidget \*history\_button = gtk\_button\_new\_with\_label("Show History");

g\_signal\_connect(history\_button, "clicked", G\_CALLBACK(on\_history\_button\_clicked), result\_label);

* Shows recent results in the label when clicked.
* **Final Show**

gtk\_widget\_show\_all(window);

gtk\_main();

* Displays the entire GUI and starts the GTK event loop.
* **GTK Widgets Used**

| **Widget** | **Purpose** |
| --- | --- |
| GtkWindow | Main application window |
| GtkGrid | Layout manager (rows/columns) |
| GtkLabel | Displays static/dynamic text |
| GtkEntry | Input field for number entry |
| GtkButton | Performs operations when clicked |

**Summary of Features**

Handles 22 scientific & arithmetic operations  
 Input via GtkEntry, result via GtkLabel  
 Clean layout using GtkGrid  
 Memory storage and history viewer