PROGRAMMING IN C

MAJOR PROJECT REPORT

1. Title Page

Project Tile: ***Function Graph Plotter***

Prepared By***: Iqra Junaid***

Submitted to: ***Dr. Tanu Singh***

2. Abstract

This Project Presents a command line function graph plotting system written in C-Language

The System allows users to select and plot mathematical functions such as:

- **Sine**
- **Cosine**
- **Tangent**
- **Exponential**
- **Polynomial functions**

The Graph is using ASCII characters in the terminal, making it platform independent.

The implementation required the use of cuntion pointers, wrapper functions, structured data storing and modular file structure dicing the project into main.c, function.c , plot.c .

3. Problem Definition

In most educational environments, plotting mathematical functions requires GUI tools like MATLAB, GeoGebra and similar tools.

These tools are often heavy on systems, not available in basic programming environments and often not suitable for learning low level plotting mechanism

OBJECTIVE

To create a terminal- based function plotting tool in C that can:

- **Take user input for the mathematical function**
- **Evaluate multiple points of these functions**
- **Map the result into a scalable ASCII coordinate system**
- **Display multiple curves in different colors**

SCOPE

The project focuses on:

- **Mathematical function evaluation**
- **ASCII plotting**
- **Modular programming**
- **Function pointers**
- **Basic numerical computation**

Not included:

- **GUI plotting**
- **Real-time animations**
- **3-D graphing**

4. System design

ALGORITHM

1. Start the program.

2. Read the user input for:

   - Function type

   -Plotting range

   -Resolution

3. Create function objects

   -Assign the pointer to the mathematical functions used

   -Store any extra parameters that a function may require

   -Auto-assign a unique color to the functions

4. Generate plot data

   -For each selected function

   -For each x-value in the specified range

   - Evaluate the function and store corresponding y-values

5. Scale axes and prepare the plotting grid.

6. Plot the functions on the terminal:

   -Map computed points to row and column positions

   -Draw axes and graph characters

   -Use each function’s assigned color for visualisation

7. Display the final ASCII plot to the user

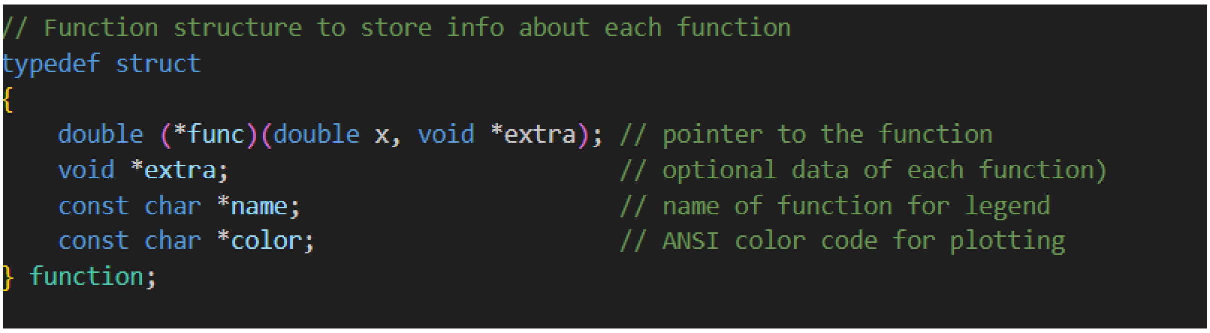
8. End the program

5. Implementation Details

The program uses a modular design:

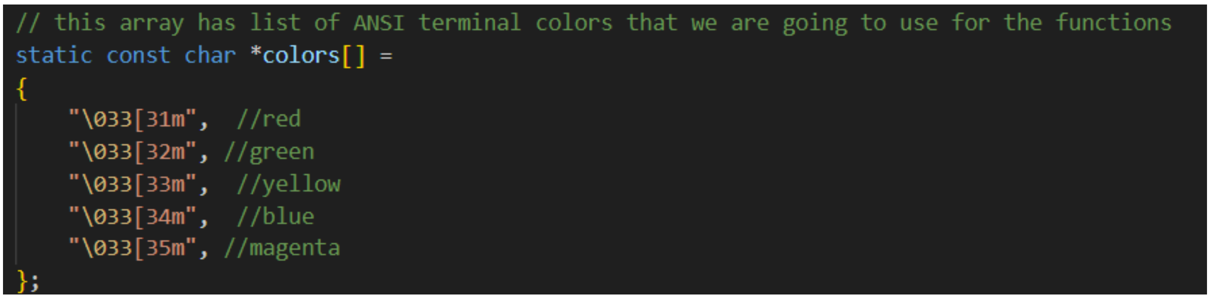
- **Main.c : has user input + function selection design**
- **Function.c : function structure definition**
- **Plot.c : ASCII graphing engine**
- **Plot.h/function.h: header declarations**

A. Function Structure



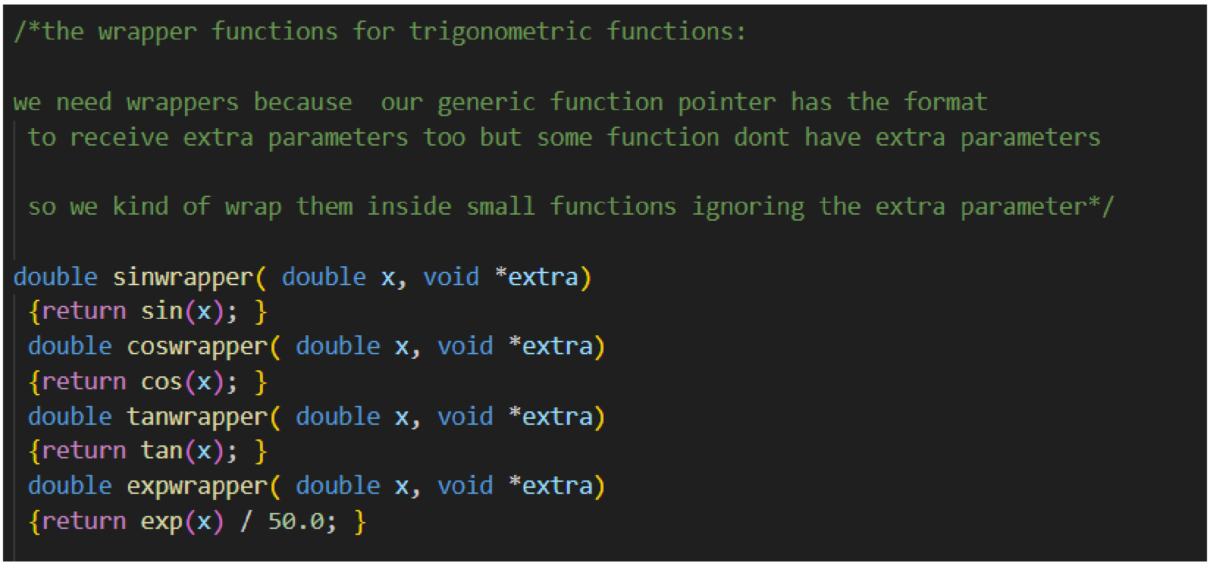
```c
// Function structure to store info about each function
typedef struct
{
    double (*func)(double x, void *extra); // pointer to the function
    void *extra;                           // optional data of each function)
    const char *name;                      // name of function for legend
    const char *color;                     // ANSI color code for plotting
} function;
```

B. ANSI terminal colors



```c
// this array has list of ANSI terminal colors that we are going to use for the functions
static const char *colors[] =
{
    "\033[31m",  //red
    "\033[32m", //green
    "\033[33m",  //yellow
    "\033[34m",  //blue
    "\033[35m", //magenta
};
```

C. Function wrappers



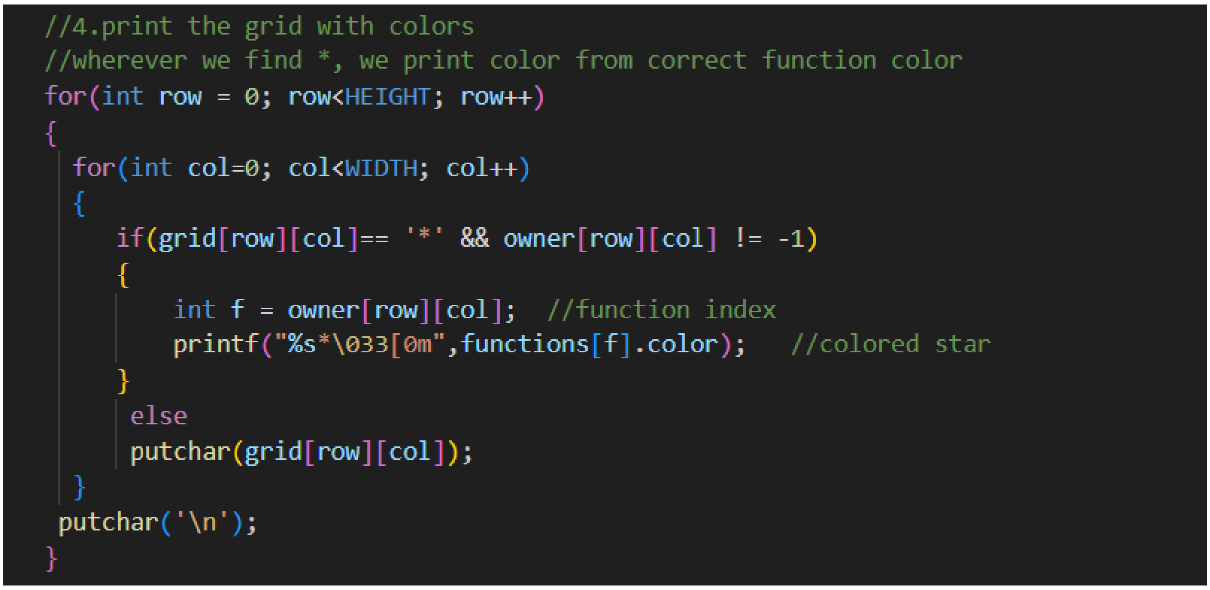
```
/*the wrapper functions for trigonometric functions:

we need wrappers because  our generic function pointer has the format
 to receive extra parameters too but some function dont have extra parameters

 so we kind of wrap them inside small functions ignoring the extra parameter*/

double sinwrapper( double x, void *extra)
 {return sin(x); }
 double coswrapper( double x, void *extra)
 {return cos(x); }
 double tanwrapper( double x, void *extra)
 {return tan(x); }
 double expwrapper( double x, void *extra)
 {return exp(x) / 50.0; }
```

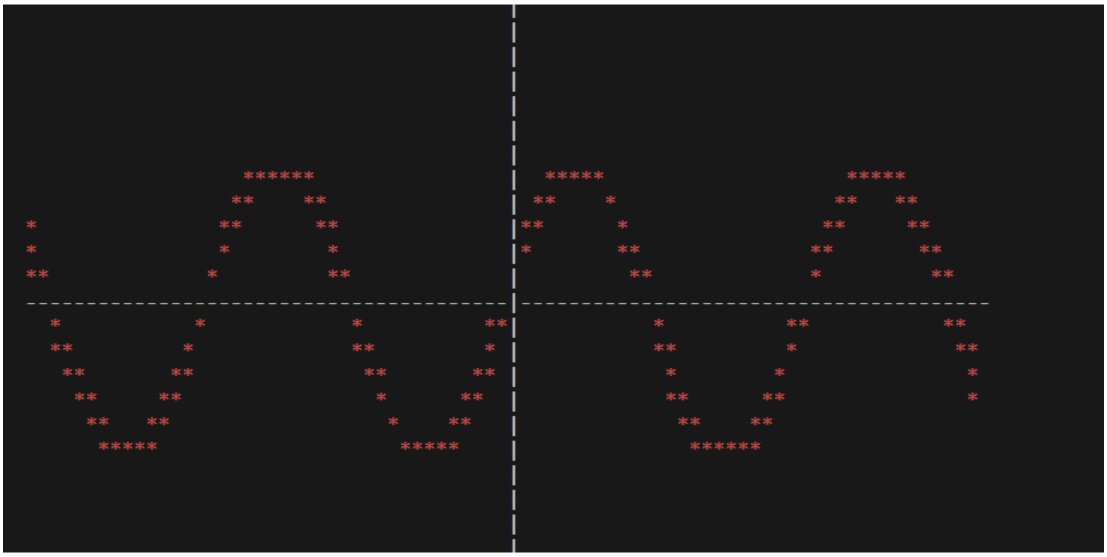
D. Plotting with color logic



```
//4.print the grid with colors
//wherever we find *, we print color from correct function color
for(int row = 0; row<HEIGHT; row++)
{
  for(int col=0; col<WIDTH; col++)
  {
     if(grid[row][col]== '*' && owner[row][col] != -1)
     {
         int f = owner[row][col];  //function index
         printf("%s*\033[0m",functions[f].color);   //colored star
     }
      else
      putchar(grid[row][col]);
  }
 putchar('\n');
}
```

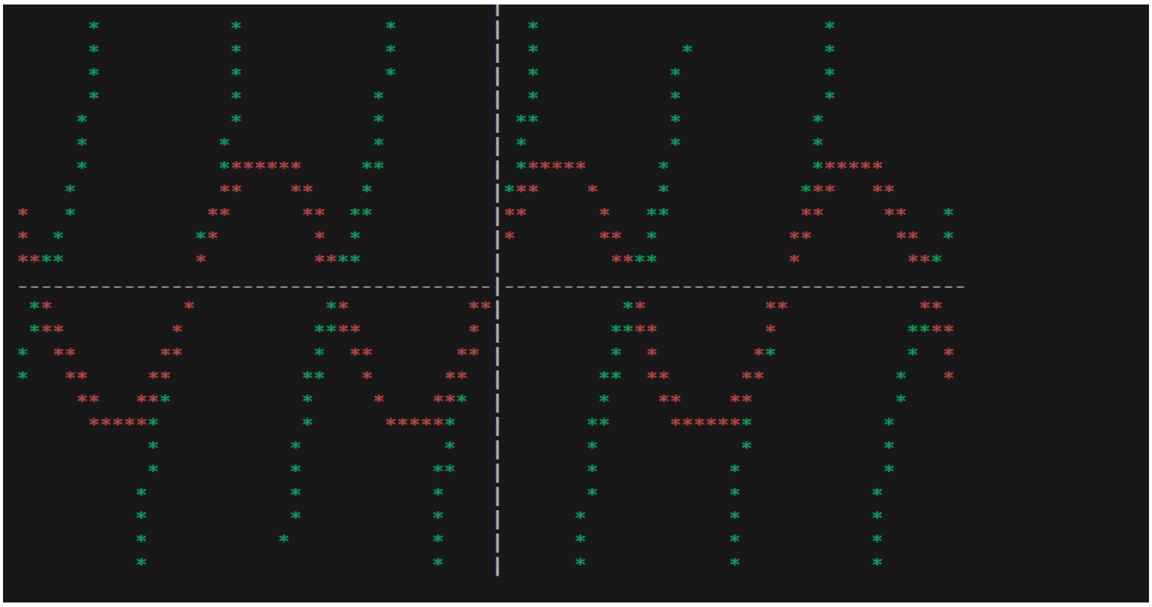
6. Testing And results

Graph of sin(x):



Multiple Functions together

Sin(x) and tan(x):



7. Conclusion and future work

The project successfully demonstrates that:

- **Mathematical graphs can be printed using RASCII without graphic libraries**
- **The modular approach makes the code maintainable.**

Future work

1. Supporting more functions

2. Adding polynomials with coefficients

3. Better plotting logic for smoother curves

4. Zoom features

8. References

1." let us c" book for core concepts

2. Online C language Tutorials