

Name: Kassim Musa Abass
Reg: SPH32/3338/2025
Program: Embedded Systems Intro

C Programming: Geometry Calculator

Objective:

To build a modular C program that computes the area of different shapes by separating function implementations into an object file and linking them with the main program.

Introduction:

This assignment explores how real C programs are built from multiple source files. A separate file (geometry.c) implements area calculation functions for circles, squares, rectangles, and triangles, which are compiled into an object file (geometry.o).

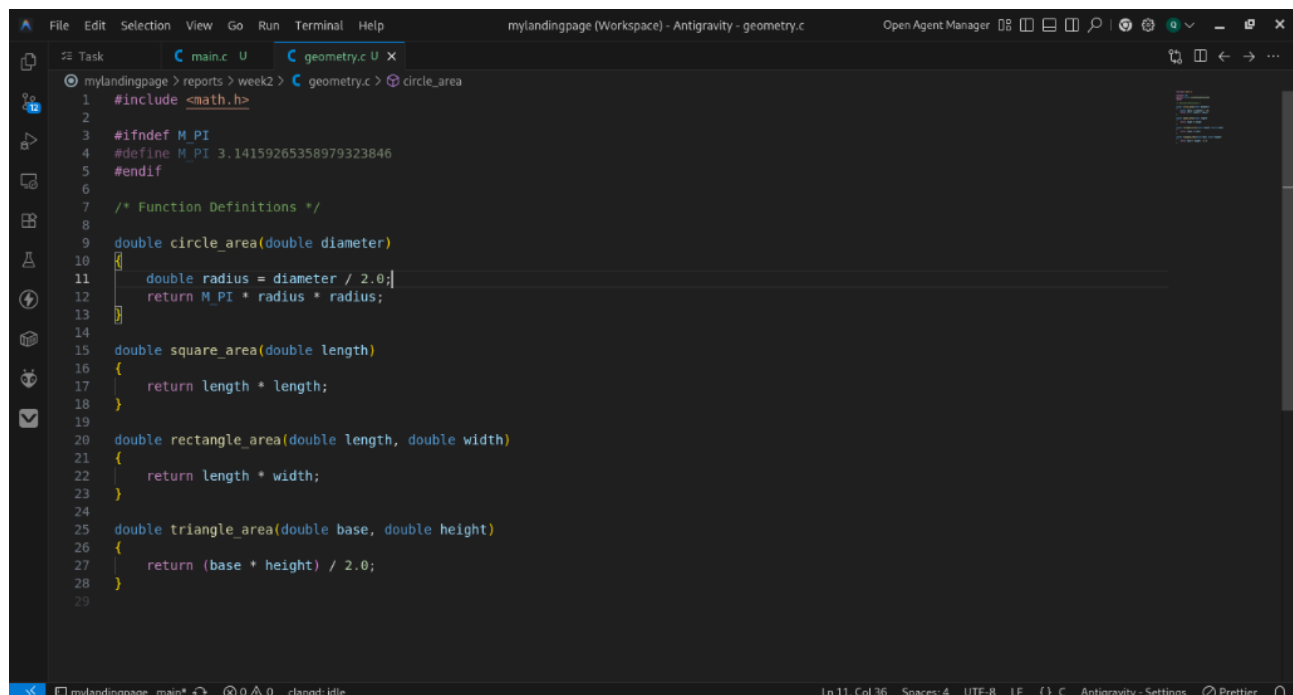
Method:

1. Documentation

The main.c file acts as the entry point and caller. The geometry.c file contains the actual math logic. By running ``clang -c geometry.c``, we create an object file. Then, ``clang main.c geometry.o -o geometry`` links both files together to create the executable geometry calculator.

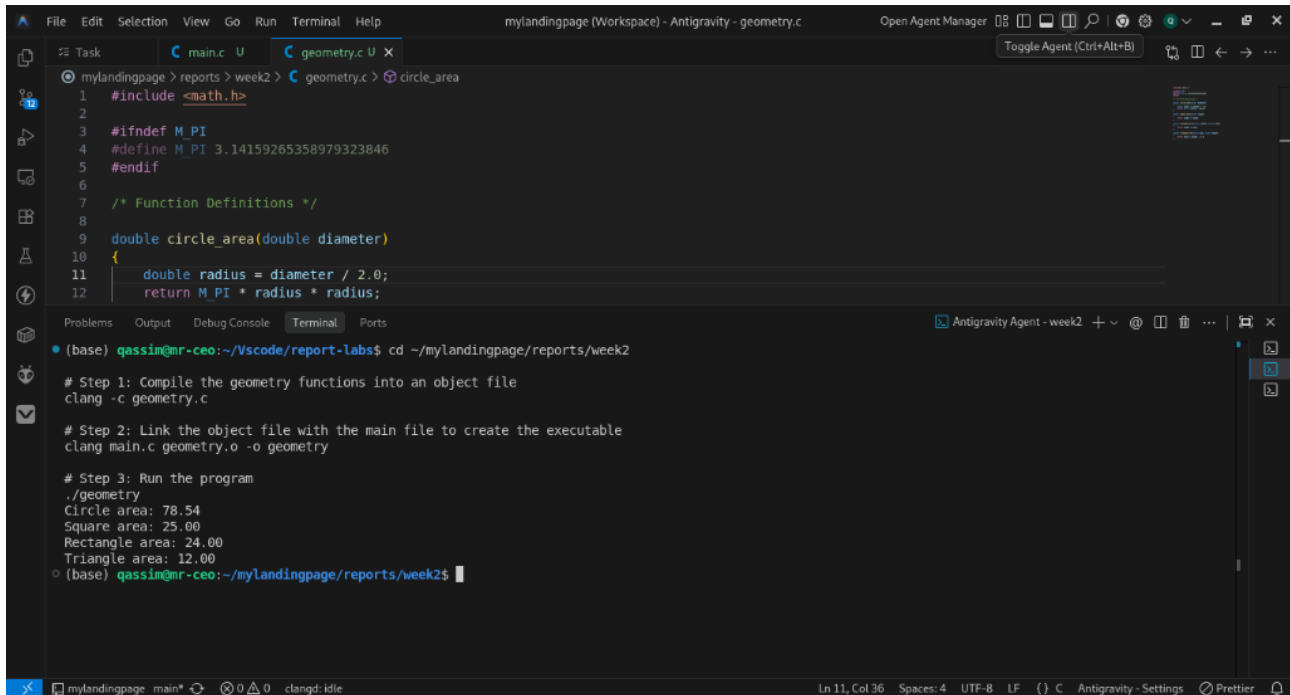
2. Code Listing

Source Code (geometry.c):



```
1 #include <math.h>
2
3 #ifndef M_PI
4 #define M_PI 3.14159265358979323846
5 #endif
6
7 /* Function Definitions */
8
9 double circle_area(double diameter)
10 {
11     double radius = diameter / 2.0;
12     return M_PI * radius * radius;
13 }
14
15 double square_area(double length)
16 {
17     return length * length;
18 }
19
20 double rectangle_area(double length, double width)
21 {
22     return length * width;
23 }
24
25 double triangle_area(double base, double height)
26 {
27     return (base * height) / 2.0;
28 }
29
```

3. Testing and Results



The screenshot shows a VS Code workspace named 'mylandingpage (Workspace) - Antigraivty - geometry.c'. The editor has two tabs: 'main.c' and 'geometry.c'. The 'geometry.c' tab is active, showing the following code:

```
1 #include <math.h>
2
3 #ifndef M_PI
4 #define M_PI 3.14159265358979323846
5 #endif
6
7 /* Function Definitions */
8
9 double circle_area(double diameter)
10 {
11     double radius = diameter / 2.0;
12     return M_PI * radius * radius;
13 }
```

The terminal window at the bottom shows the following commands and output:

```
(base) qassin@mr-ceo:~/Vscode/report-labs$ cd ~/mylandingpage/reports/week2
# Step 1: Compile the geometry functions into an object file
clang -c geometry.c
# Step 2: Link the object file with the main file to create the executable
clang main.c geometry.o -o geometry
# Step 3: Run the program
./geometry
Circle area: 78.54
Square area: 25.00
Rectangle area: 24.00
Triangle area: 12.00
(base) qassin@mr-ceo:~/mylandingpage/reports/week2$
```

The status bar at the bottom indicates the current file is 'main.c', the editor is using 'clangd' as the language server, and the encoding is 'UTF-8'.

4. Analysis, Discussion and Conclusion

The program successfully links multiple C files together and outputs the correct calculations, demonstrating how modular C application structure simplifies compilation and testing.