the basic structure of a C program

The basic structure of a C program is quite straightforward, but understanding each part is key to writing effective code.

---

1. Preprocessor Commands (Directives) ⚙️

These lines start with a # symbol and are processed before the actual compilation of the code. They typically include:

- **#include**: Used to include **header files**, which contain declarations for functions and macros. For example, #include <stdio.h> includes the standard input/output library, allowing you to use functions like printf() for printing to the console.

  C

  #include <stdio.h> // Includes the standard input/output library
  #include <stdlib.h> // Includes the standard library

---

2. Global Declarations 🌍

This section is where you declare **global variables** and **function prototypes**.

- **Global variables** are accessible from anywhere in the program.
- **Function prototypes** declare a function's return type, name, and parameters before its actual definition, allowing the compiler to know about the function's existence.

  C

  int globalVar = 10; // A global variable

  void myFunction(); // Function prototype

---

3. main() Function 🚀

Every C program must have a **main() function**. This is the **entry point** of your program – execution always begins here.

- It typically returns an int (integer) value, where 0 usually indicates successful execution, and any other value indicates an error.
- It can optionally take arguments (argc and argv) for command-line input.

C

```
int main() {
    // Code execution starts here
    return 0; // Indicates successful execution
}
```

---

4. Local Declarations & Statements (Inside main() and other functions) 📝

Inside main() or any other function, you'll find:

- **Local Declarations:** Variables declared inside a function are **local** to that function and can only be used within it.
- **Statements:** These are the instructions that the program executes. They perform actions like calculations, input/output, conditional checks, and loops. Each statement typically ends with a semicolon (;).

C

```
int main() {
    int localVar = 5; // Local variable declaration
    printf("Hello, C Program!\n"); // Statement to print text
    myFunction(); // Calling another function
    return 0;
}
```

---

5. User-Defined Functions (Optional) 🛠️

Beyond main(), you can define your own functions to break down complex tasks into smaller, manageable pieces. This promotes **modularity** and **code reusability**.

```c
void myFunction() {
    printf("This is a user-defined function.\n");
}
```

Putting It All Together: An Example 👨‍💻

C
```
#include <stdio.h> // Preprocessor command: includes standard input/output library

// Global declaration: a global variable
int globalNumber = 100;

// Function prototype: declares a user-defined function
void greetUser(char name[]);

// main function: entry point of the program
int main() {
    // Local declaration: a local variable
    char userName[] = "Eugene";

    // Statements: print output and call a user-defined function
    printf("Welcome to our C programming lesson!\n");
    printf("Global number: %d\n", globalNumber);
    greetUser(userName); // Call to user-defined function

    return 0; // Indicates successful execution
}

// User-defined function definition
void greetUser(char name[]) {
    printf("Hello, %s! Nice to have you here.\n", name);
}
```

This structure provides a clear and organized way to write C programs, making them easier to read, debug, and maintain.

Here's a visual representation of the basic C program structure: