First C Program

When starting with C programming, understanding the structure and components of a C program is crucial. In this article, we will take a closer look at a simple C program, explain it line by line, and demonstrate how it is executed on different platforms.

C

```c
#include <stdio.h>
int main()
{
    printf(" Hello,  GeeksforGeeks");
    return 0;
}
```

```
Hello,  GeeksforGeeks
```

Output

```
GeeksforGeeks
```

Line by Line Explanation

Line 1: #include <stdio.h>

This line is a preprocessor directive that includes the **stdio.h** header file. Header files in C contain declarations for various standard functions. Specifically, stdio.h stands for Standard Input Output. It includes declarations for input and output functions like printf (to print output) and scanf (to read input).

Line 2: int main()

This line declares the **main** function. The **main** function is a special function in C because. It is the entry point of every C program.

What is int?

int is the return type of the main function. Functions in C can take inputs and return outputs. The main function typically returns a status code to indicate whether the program executed successfully.

- Returning 0 indicates success.
- Returning a non-zero value indicates an error.

Line 3 and 8: { and }

The curly brackets **{** and **}** define the scope of the main function. The opening bracket { marks the beginning of the function. The closing bracket } marks the end of the function.

What is a Scope?

A **scope** limits the lifetime and accessibility of variables declared within it. For every function in C, you must use curly brackets to define its body.

Line 4: printf("Geeks for Geeks");

The **printf** function is a standard output function used to print a string to the screen.

What is a String?

A string is a sequence of characters enclosed in double quotes (" "). In this case, "Geeks for Geeks" is printed to the screen.

Semicolon

The semicolon (**;**) at the end of the line is a statement terminator. Every statement in C must end with a semicolon. If omitted, the compiler will throw an error and refuse to compile the program.

Compiling and Running the Program

If you are using an IDE (Integrated Development Environment), the program can be written, compiled, and executed with menu buttons or shortcuts like Ctrl+F9 or Alt+F9.

If you are using a traditional text editor like Notepad or VI, you have to manually compile and run the program. Follow the below steps:

- Save the program as a .c file, e.g., program.c.
- Use the GCC compiler to compile and run the program:

For Linux:

- Use the below command to compile the program:

```
gcc program.c
```

- This generates an executable file named **a.out** by default.
- To run the executable, use the following command:

```
./a.out
```

- The executable file name can be customized using the below command:

```
gcc program.c -o myprogram
./myprogram
```

For Windows:

- Compile the program using below command:

```
gcc program.c
```

- This generates an executable file named a.exe.
- To run the program:

```
a
```

After compilation and linking, the generated executable file is an independent program. You can run it without requiring a compiler.

What Happens During Compilation?

The process of compiling a C program involves:

1. **Preprocessing:** The preprocessor processes directives like #include and expands macros. It generates a larger version of the C program.
2. **Compilation:** The compiler converts the preprocessed C code into object code (binary format).
3. **Linking:** The linker combines your object code with external libraries (e.g., printf from stdio.h) to generate an executable file.