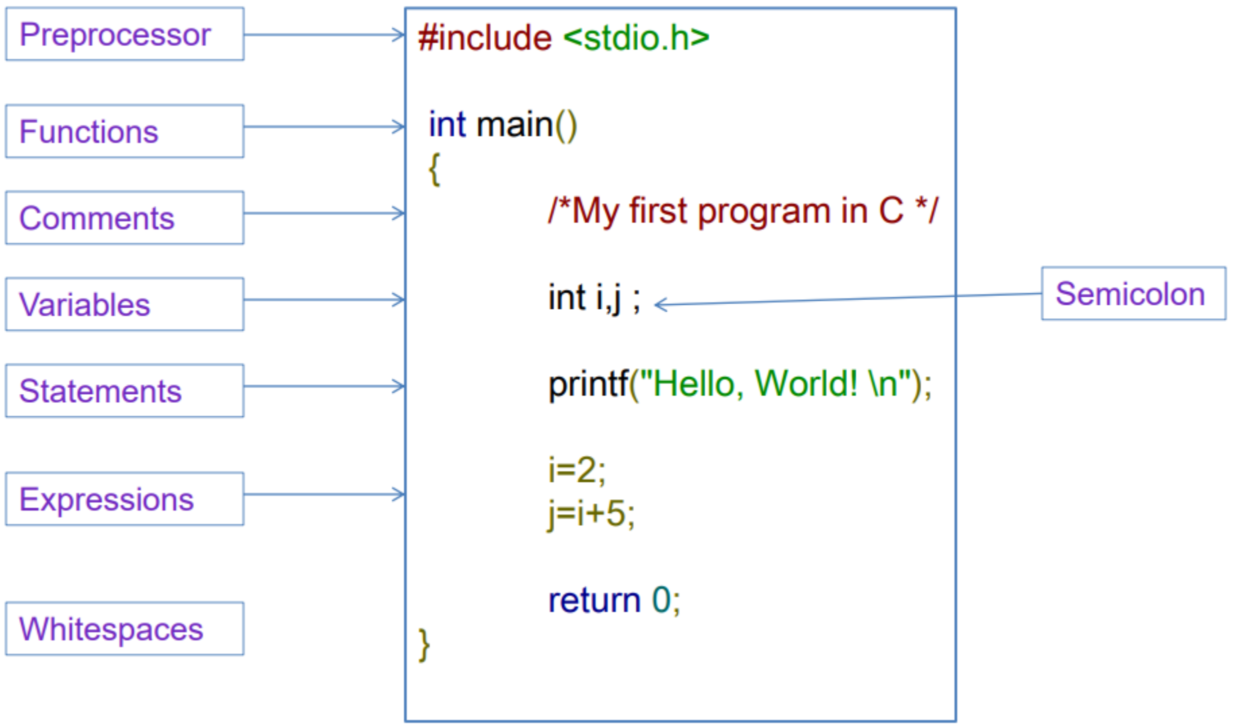
C Program Basic Structure

Any program can be written in this structure only. Writing a C program in any other structure will hence lead to a Compilation Error.



```
#include <stdio.h>

int main()
{
        /*My first program in C */

        int i,j ;

        printf("Hello, World! \n");

        i=2;
        j=i+5;

        return 0;
}
```

Program Structure Explanation:

1 - Pre-processor

As the name suggests Preprocessors are programs that process our source code before compilation.

#include is a pre-processor directive in 'C.'

Preprocessor programs provide preprocessors directives which tell the compiler to preprocess the source code before compiling. All of these preprocessor directives begin with a ‘#’ (hash) symbol. The ‘#’ symbol indicates that, whatever statement starts with #, is going to the preprocessor program, and preprocessor program will execute this statement.

Examples of some preprocessor directives are: *#include*, *#define*, *#ifndef* etc. Remember that # symbol only provides a path that it will go to the preprocessor, and command such as include is processed by preprocessor program. For example, include will include extra code to your program. We can place these preprocessor directives anywhere in our program.

2 - Header file

A Header file is a collection of built-in(readymade) functions, which we can directly use in our program. Header files contain definitions of the functions which can be incorporated into any C program by using pre-processor #include statement with the header file. Standard header files are provided with each compiler, and covers a range of areas like string handling, mathematical functions, data conversion, printing and reading of variables.

With time, you will have a clear picture of what header files are, as of now consider as a readymade piece of function which comes packaged with the C language and you can use them without worrying about how they work, all you have to do is include the header file in your program.

Before using any function, we have to first include the required file, also known as a header file (.h).

You can also create your own functions, group them in header files and declare them at the top of the program to use them. To include a file in a program, use pre-processor directive

```
#include <file-name>.h
```

File-name is the name of a file in which the functions are stored. Pre-processor directives are always placed at the beginning of the program.

Some of C Header files:

1. stddef.h – Defines several useful types and macros.
2. stdint.h – Defines exact width integer types.
3. stdio.h – Defines core input and output functions
4. stdlib.h – Defines numeric conversion functions, pseudo-random network generator, memory allocation
5. string.h – Defines string handling functions
6. math.h – Defines common mathematical functions

#include <stdio.h>, stdio is the library where the function **printf** is defined. printf is used for generating output.

the printf() function. All header files will have an extension .h

3 - The main function

The main function is a part of every 'C' program. We can represent the main function in various forms, such as:

- main()
- int main()
- void main()
- main(void)
- void main(void)
- int main(void)

The empty parentheses indicate that this function does not take any argument, value or a parameter. You can also represent this explicitly by placing the keyword void inside the parentheses.

The keyword void means the function does not return any value, in this case, the last statement is always getch ().

```
#include<stdio.h>        //Pre-processor directive
int main()               //main function declaration
{
printf("Hello World");   //to output the string on a display
return 0;                //terminating function
}
```

In the above example, the keyword int means the function will return an integer value. In this case, the last statement should always return 0.

4 - The source code Block

After the main function has been declared, we have to specify the opening and closing parentheses. Curly brackets { }, indicate the starting and end of a program. These brackets must be always put after the main function.

All the program code is written inside these brackets, such as declarative and executable part.

The printf function generates the output by passing the text "Hello World!"

The semicolon; determines the end of the statement. In C, each statement must end with a semicolon.

So, we have successfully installed the compiler and now can begin working in 'C.' We will write a simple program that will say hello to us. Let's start.

Variable Declaration:

Any C program is the variable declaration. It refers to the variables that are to be used in the function. Please note that in the C program, no variable can be used without being declared. Also, in a C program, the variables are to be declared before any operation in the function.

```
int ccat;
```

Semicolon;

Semicolon; is used to mark the end of a statement and beginning of another statement. Absence of semicolon at the end of any statement, will mislead the compiler to think that this statement is not yet finished and it will add the next consecutive statement after it, which may lead to compilation(syntax) error.

Comment

Comments can be inserted into C programs by bracketing text with the /* and */ delimiters. As will be discussed later, comments are useful for a variety of reasons. Primarily they serve as internal documentation for program structure and functionality.

Using `//` This is used to write a single line comment.

```
// single line comment example
```

Using `/* */`: The statements enclosed within `/*` and `*/` , are used to write multi-line comments.

```
/* Sample Multiline Comment
Line 1
Line 2
….
…
*/
```

Why use comments?

1. Documentation of variables and functions and their usage
2. Explaining difficult sections of code
3. Describes the program, author, date, modification changes, revisions…

Return statement - return 0;

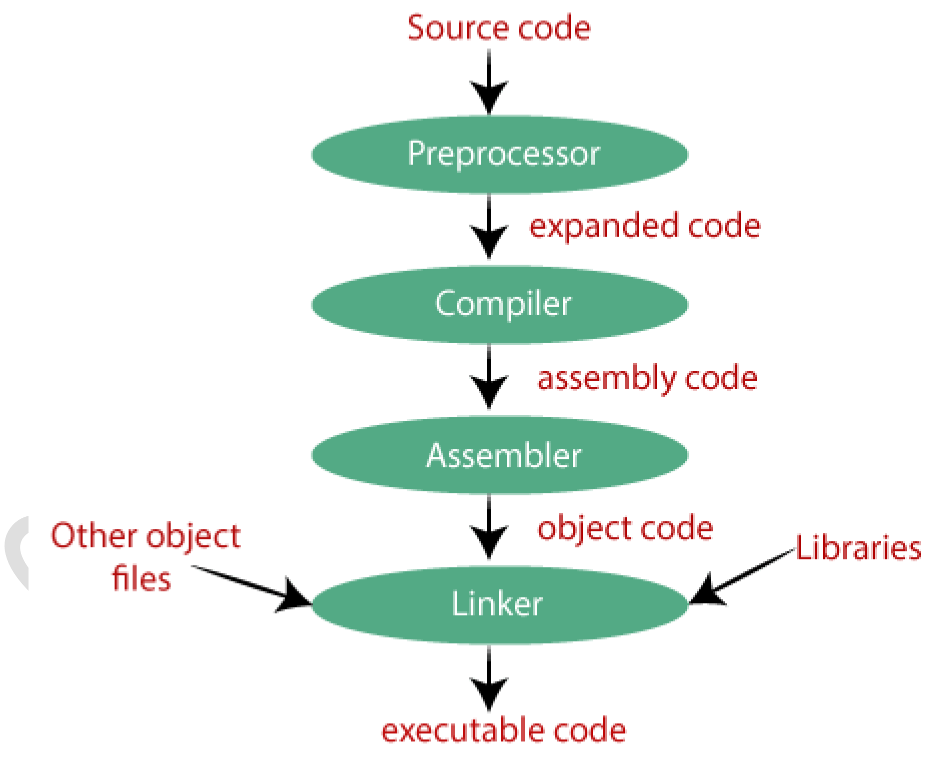
A return statement is just meant to define the end of any C program.

All the C programs can be written and edited in normal text editors like Notepad or Notepad++ and must be saved with a file name with extension as .c

If you do not add the extension .c then the compiler will not recognise it as a C language program file.

Execution Process

1. Firstly, the input file, i.e., HelloWorld.c, is passed to the pre-processor, and the pre-processor converts the source code into expanded source code. The extension of the expanded source code would be HelloWorld.i.
2. The expanded source code is passed to the compiler, and the compiler converts this expanded source code into assembly code. The extension of the assembly code would be HelloWorld.s.
3. This assembly code is then sent to the assembler, which converts the assembly code into object code.
4. After the creation of an object code, the linker creates the executable file. The loader will then load the executable file for the execution.



Difference between Compile and Run

You must be thinking why it is a 2-step process, first we compile the code and then we run the code. So, compilation is the process where the compiler checks whether the program is correct syntax wise, and there are no errors in the syntax.

When we run a compiled program, then it actually executes the statements inside the **main ()** function

Summary

1. C is a case sensitive language so all C instructions must be written in lower case letter.
2. The main function is a mandatory part of every 'C' program.
3. To use the functionality of a header file, we have to include the file at the beginning of our program.
4. Every 'C' program follows a basic structure.
5. All C statement must end with a semicolon.
6. C has a free-form line structure. End of each statement must be marked with a semicolon.
7. The two braces, { and }, signify the begin and end segments of the program.
8. Whitespace is used in C to describe blanks and tabs
9. Whitespace is required between keywords and identifiers. We will learn about keywords and identifiers in the next tutorial.