Basic Syntax in C Programming

In this section, let see basic syntax rules that we need to follow while writing a C program. It mainly composed with tokens, identifiers, keywords, semicolons, comments and whitespaces.

Tokens

All identifiers, keywords, symbols, literals etc are together known as tokens. For example, printf, curly braces ({,}), round braces ((,)), semicolon (;), any string statements inside printf or code are known as tokens.

```
printf (“Enter the name :”);
```

Here printf, ‘(‘, ‘Enter the name:’, ‘)’ and ‘;’ are the tokens.

Identifiers

These are the variable names, function names or any other user defined names within the program. Standard format for any identifier name is to begin with alphabets (upper or lower case) or underscore (_). It is then followed by any alphabets (upper or lower case) or digits. But it does not allow ‘@’, ‘$’ and ‘%’ to be used within the identifier. C identifiers are case sensitive.

```
Var1, var2, _sum, str_name, strName, fltValue, intNumValue, avg_std_100
```

Keywords

These are the reserved words in C, which are used to identify the variables or perform some functions within the program. For example, printf, scanf, if, else, for, while, loop, switch, int, float, char, double, struct, const, goto, return, typedef etc.

Semicolons

All codes in C have to be terminated by semicolon. It indicates the end of the line of the code.

```
printf (“Enter the name :”);

getchar ();

return 0;
```

Comments

Comments are the non-compliable lines of code in the program. They are used to give the information about the code. When compiler encounters comments, it ignores those lines and proceeds with next compliable code. In C comments are written within ‘/*’ and ‘*/’ for multiline comments and single line comments are written after ‘//’.

/* defines an equivalent notation for ‘=’ as ‘EQ’.
Whenever the compiler sees EQ in the code, it replaces it by ‘=’*/

// defines an equivalent notation for ‘=’ as ‘EQ’.

Whitespaces

Whitespaces are used to separate two identifiers, keywords, any tokens or to have blank line, new line etc. It differentiates the any tokens from other while compiling and when user sees the code

```
printf (“Enter the name :”);

sum = var1 + var2;
```