Tokens in C

In this tutorial you will learn about the six tokens such as keywords, Identifiers, constants, strings, special characters and operators used in C with their examples.You have already mastered the basic syntax of C program in our previous tutorial which will help you to understand the structure of C program.

Tokens in C

In the world of programming, the term 'tokens' imply the basic elements of a program. In English language, verbs, adverbs, nouns are the smallest elements of a sentence; similarly, a computer program consists of different types of tokens. Here is a sample program which will help you understand the basic syntax and properties of tokens.

```
int main()
{
int a,b,c;
a=2,b=3;
c=a+b;
printf("\n Total= %d",c);
getch();
}
```

Output:

```
Total= 5
```

In the above mentioned program the following tokens have been used-- int, a, b, c, {, }, (,), %, =, +, ;. Tokens can be classified broadly into six categories and we will get an idea of them in the next section:

Keywords in C

Keywords are a group of words which are solely reserved for doing specific functions. In C, there are such 32 keywords such as char, int, do, while, sizeof etc.

```
int a,b,c;
```

In the above-mentioned example, 'int' is a keyword which signifies an integer value. The statement implies that a, b, c can have integer values (that is their value can be 1, 2, 3 or -1, -2, -3 but can never be like 0.1, 0.2, -0.1,0.2).

Another important point is these **keywords cannot be used for another purpose inside the program**. For example 'int car, char;' is not a valid statement; whereas 'int jar, far' is a valid statement. It is because 'char' is the name of a function and it is a reserved keyword which implies a specific character.

The following list shows the Keywords in C. These words are strictly reserved for special purposes and may not be used as constants or variables or any other identifier names. C being a case sensitive language all its keywords are given in lowercase.

auto	else	long	switch
break	enum	register	typedef
case	extern	return	union
char	float	short	unsigned
const	for	signed	void
continue	goto	sizeof	volatile
default	if	static	while

Identifiers in C

In every C program, there are some entities which define the characteristics of the elements of a program. Variables, array, function come under this category.

```
int a =2,b =3,c;
c=a+b;
printf("\n Total= %d",c);
getch();
```

In the above-mentioned program a, b, c, printf, getch are program identifiers. Here a, b, c are variable as their value can be assigned any time. If you write any statement like a=5 after writing a=2 anywhere in the program, the compiler will consider the latest assigned value only that is 5.

Rules for naming Identifiers in C

1. Valid identifiers must start either with an alphabet (uppercase or lowercase) or an underscore (_) followed by alphanumeric letters or underscores.
2. Identifiers must not be a keyword in C
3. An identifier can be of any length

Valid Identifiers	Invalid Identifiers
_age	17age
Num ,NUM	int,float
Stud_1,Stud_2	hash#

Num and NUM are taken as two different variables as C is case sensitive.

Constants in C

As its name suggests, constants account for some fixed values which can not be changed later anywhere in the program. Constant is declared by Keyword '**const**'.

```
const pi = 3.14;
```

If you write a statement 'const int Z=10;', you can never change the value of Z to 11 or 12 later. It is very helpful in minimizing errors in programs with complex mathematical algorithms (for ex π=3.1414)

Strings in C

Strings are nothing but an array of characters. In the English language, a group of alphabets is called words. Similarly here a set of characters that is alphabets, numbers or symbols is called String. In the above-mentioned program, '**Total**' is a string.

```
printf("\n Total= %d",c);
```

Special Characters in C

In C, special characters mean a set of symbols tell the compiler to perform a specific task. Here '\n' means the next line. In the printf() section wherever you will type it, the compiler will put the cursor in the next line.

```
printf("\n Total= %d",c);
```

Operators in C

We know that the CPU of a computer works with ALU – Arithmetic and Logic Unit. Similarly, the C compiler performs all its operations using arithmetic and logic operators.

```
c=a+b;
```

Arithmetic operators used in C are +, -, /, *. In the sample program use of '+' has been shown. Logical operators of C are <, >, etc. All these operators work extensively with the logic gates of the chipset to produce the desired output.

Semicolons in C

Semicolons in C denote the **end of every statement** which is symmetrical with the ending of a sentence with the full stop in English language. If you leave a whitespace after a statement, the compiler will not recognize it as an end of the statement. Moreover in programming '.' is used as an extension name(stdio.h). This is why semicolons have been introduced here to mark the statements end.

Whitespace in C

Whitespace in C means a series of '**null**' characters which displays a blank space when the program is run. Blanks, tabs, newline characters and comments are commonly found whitespaces in C.

Normally printf(" ") gives a blank space as output. If we want to move to the next line we can use '\n' or '\n\n' (shown in the sample program). We can also write '\t' to move the cursor to the next tab.

White Space/ Special characters	Meaning	Description
\n	newline	Shifts the cursor to the beginning of the next line.
\t	tab	Shifts the cursor to the next tab stop.
\b	backspace	Shifts the cursor back by one space on the current line.
\r	carriage return	Shifts the cursor to the beginning of the current line.
\a	bell(alert)	Generates a beep sound.
\\	backslash	Prints the backslash (\) character.
\0	null	indicates a null character.
\'	single quote	Prints the single quote (') character.