C Tokens

A token is the smallest element of a program that is meaningful to the compiler.

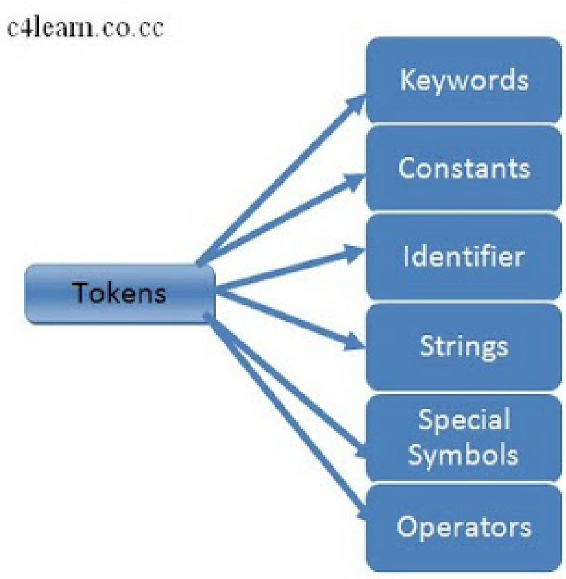
C tokens are the basic buildings blocks in C language which are constructed together to write a C program.

Each and every smallest individual units in a C program are known as C tokens.

C tokens are of six types. They are,

Keywords	(eg: int, while),
Identifiers	(eg: main, total),
Constants	(eg: 10, 20),
Strings	(eg: “total”, “hello”),
Special symbols	(eg: (), {}),
Operators	(eg: +, /,-,*)

Tokens can be classified as follows:



1. **Keyword:** Keywords are pre-defined or reserved words in a programming language. Each keyword is meant to perform a specific function in a program. Since keywords are referred names for a compiler, they can’t be used as variable names because by doing so, we are trying to assign a new meaning to the keyword which is not allowed. You cannot redefine keywords. However, you can specify text to be substituted for keywords before compilation by using C/C++ preprocessor directives.**C** language supports **32** keywords which are given below:

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

Identifiers: Identifiers are used as the general terminology for naming of variables, functions and arrays. These are user defined names consisting of arbitrarily long sequence of letters and digits with either a letter or the underscore(_) as a first character. Identifier names must differ in spelling and case from any keywords. You cannot use keywords as identifiers; they are reserved for special use. Once declared, you can use the identifier in later program statements to refer to the associated value. A special kind of identifier, called a statement label, can be used in goto statements.

There are certain rules that should be followed while naming c identifiers:

- They must begin with a letter or underscore(_).
- They must consist of only letters, digits, or underscore. No other special character is allowed.
- It should not be a keyword.
- It must not contain white space.
- It should be up to 31 characters long as only first 31 characters are significant.

Some examples of c identifiers:

NAME	REMARK
_A9	Valid
Temp.var	Invalid as it contains special character other than the underscore
void	Invalid as it is a keyword

C Constants

C constants refers to the data items that do not change their value during the program execution. Several types of C constants that are allowed in C are:

1. Integer Constants

Integer constants are whole numbers without any fractional part. It must have at least one digit and may contain either + or – sign. A number with no sign is assumed to be positive.

There are three types of integer constants:

1.1. Decimal Integer Constants

Integer constants consisting of a set of digits, 0 through 9, preceded by an optional – or + sign.

Example of valid decimal integer constants
341, -341, 0, 8972

1.2. Octal Integer Constants

Integer constants consisting of sequence of digits from the set 0 through 7 starting with 0 is said to be octal integer constants.

Example of valid octal integer constants
010, 0424, 0, 0540

1.3. Hexadecimal Integer Constants

Hexadecimal integer constants are integer constants having sequence of digits preceded by 0x or 0X. They may also include alphabets from A to F representing numbers 10 to 15.

Example of valid hexadecimal integer constants
0xD, 0X8d, 0X, 0xbD

It should be noted that, octal and hexadecimal integer constants are rarely used in programming.

2. Real Constants

The numbers having fractional parts are called real or floating point constants. These may be represented in one of the two forms called *fractional form* or the *exponent form* and may also have either + or – sign preceding it.

Example of valid real constants in fractional form or decimal notation
0.05, -0.905, 562.05, 0.015

Representing a real constant in exponent form

The general format in which a real number may be represented in exponential or scientific form is

mantissa e exponent

The mantissa must be either an integer or a real number expressed in decimal notation. The letter e separating the mantissa and the exponent can also be written in uppercase i.e. E
And, the exponent must be an integer.

Examples of valid real constants in exponent form are:
252E85, 0.15E-10, -3e+8

3. Character Constants

A character constant contains one single character enclosed within single quotes.

Examples of valid character constants
‘a’ , ‘Z’, ‘5’

It should be noted that character constants have numerical values known as ASCII values, for example, the value of ‘A’ is 65 which is its ASCII value.

Escape Characters/ Escape Sequences

C allows us to have certain non graphic characters in character constants. Non graphic characters are those characters that cannot be typed directly from keyboard, for example, tabs, carriage return, etc.

These non graphic characters can be represented by using escape sequences represented by a backslash() followed by one or more characters.

NOTE: An escape sequence consumes only one byte of space as it represents a single character.

Escape Sequence	Description
\a	Audible alert(bell)
\b	Backspace
\f	Form feed
\n	New line
\r	Carriage return
\t	Horizontal tab
\v	Vertical tab
\\	Backslash
\“	Double quotation mark
\‘	Single quotation mark

\?	Question mark
\0	Null

String Constants

String constants are sequence of characters enclosed within double quotes. For example,
“hello”
“abc”
“hello911”

Every sting constant is automatically terminated with a special character **‘’** called the **null character** which represents the end of the string.

For example, “hello” will represent “hello” in the memory.

Thus, the size of the string is the total number of characters plus one for the null character.

Special Symbols:

The following special symbols are used in C having some special meaning and thus, cannot be used for some other purpose.[] () {}, ; * = #

- **Brackets[]:** Opening and closing brackets are used as array element reference. These indicate single and multidimensional subscripts.
- **Parentheses():** These special symbols are used to indicate function calls and function parameters.
- **Braces{}:** These opening and ending curly braces marks the start and end of a block of code containing more than one executable statement.
- **comma (,):** It is used to separate more than one statements like for separating parameters in function calls.
- **semi colon :** It is an operator that essentially invokes something called an initialization list.
- **asterick (*):** It is used to create pointer variable.
- **assignment operator:** It is used to assign values.
- **pre processor(#):** The preprocessor is a macro processor that is used automatically by the compiler to transform your program before actual compilation.

Operators:

Operators are symbols that triggers an action when applied to C variables and other objects. The data items on which operators act upon are called operands.
Depending on the number of operands that an operator can act upon, operators can be classified as follows:

- **Unary Operators:** Those operators that require only single operand to act upon are known as unary operators.For Example increment and decrement operators
- **Binary Operators:** Those operators that require two operands to act upon are called binary operators. **Binary operators are classified into :**
  1. Arithmetic operators
  2. Relational Operators
  3. Logical Operators
  4. Assignment Operators
  5. Conditional Operators
  6. Bitwise Operators

  Ternary Operators: These operators requires three operands to act upon. For Example Conditional operator(?:).