C Tokens

Character set :

A character denotes any alphabet, digit or special symbol used to represent information. Valid alphabets, numbers and special symbols allowed in C. The alphabets, numbers and special symbols when properly combined form constants, variables and keywords.

Identifiers :

Identifiers are user defined word used to name of entities like variables, arrays, functions, structures etc.

Rules for naming identifiers are:

1) Name should only consists of alphabets (both upper and lower case), digits and underscore (_) sign.
2) First characters should be alphabet or underscore
3) Name should not be a keyword
4) Since C is a case sensitive, the upper case and lower case considered differently, for example code, Code, CODE etc. are different identifiers.
5) Identifiers are generally given in some meaningful name such as value, net_salary, age, data etc. An identifier name may be long, some implementation recognizes only first eight characters, most recognize 31 characters. ANSI standard compiler recognize 31 characters. Some invalid identifiers are 5cb, int, res#, avg no etc.

Keyword :

There are certain words reserved for doing specific task, these words are known as **reserved word** or **keywords.** These words are predefined and always written in lower case or small letter. These keywords can't be used as a variable name as it assigned with fixed meaning. Some examples are **int, short, signed, unsigned, default, volatile, float, long, double, break, continue, typedef, static, do, for, union, return, while, do, extern, register, enum, case, goto, struct, char, auto, const etc.**

Data types :

Data types refer to an extensive system used for declaring variables or functions of different types before its use. The type of a variable determines how much space it occupies in storage and how the bit pattern stored is interpreted. The value of a variable can be changed any time.

C has the following 4 types of data types

Basic built-in data types: int, float, double, char
Derived data type: pointer, array, structure, union
User Defined data types : typedef,enum

- A variable declared to be of type int can be used to contain integral values only that is, values that do not contain decimal places.
- A variable declared to be of type float can be used for storing floating- point numbers (values containing decimal places).
- The double type is the same as type float, only with roughly twice the precision.
- The char data type can be used to store a single character, such as the

letter *a*, the digit character *6*, or a semicolon similarly a variable declared char can only store character type value.

There are two types of type qualifier in c
Size qualifier: short, long
Sign qualifier: signed, unsigned
When the qualifier unsigned is used the number is always positive, and when signed is used number may be positive or negative. If the sign qualifier is not mentioned, then by default sign qualifier is assumed. The range of values for signed data types is less than that of unsigned data type. Because in signed type, the left most bit is used to represent sign, while in unsigned type this bit is also used to represent the value. The size and range of the different data types on a 16 bit machine is given below:

Basic data type	Data type with type qualifier	Size (byte)	Range
char	char or signed char Unsigned char	1 1	-128 to 127 0 to 255
int	int or signed int unsigned int short int or signed short int unsigned short int long int or signed long int unsigned long int	2 2 1 1 4 4	-32768 to 32767 0 to 65535 -128 to 127 0 to 255 -2147483648 to 2147483647 0 to 4294967295
float	float	4	-3.4E-38 to 3.4E+38
double	double Long double	8 10	1.7E-308 to 1.7E+308 3.4E-4932 to 1.1E+4932

Constants :

Constant is any value that cannot be changed during program execution. In C, any number, single character, or character string is known as a *constant*. A constant is an entity that doesn't change whereas a variable is an entity that may change.

For example, the number 50 represents a constant integer value. The character string "Programming in C is fun.\n" is an example of a constant character string.

Numeric constant :

Numeric constant consists of digits. It required minimum size of 2 bytes and max 4 bytes. It may be positive or negative but by default sign is always positive. No comma or space is allowed within the numeric constant and it must have at least 1 digit. The allowable range for integer constants is -32768 to 32767. Truly speaking the range of an Integer constant depends upon the compiler.
For a 16-bit compiler like Turbo C or Turbo C++ the range is –32768 to 32767.
For a 32-bit compiler the range would be even greater. Mean by a 16-bit or a 32- bit compiler, what range of an Integer constant has to do with the type of compiler.

It is categorized as :

1. Integer constant :-

An integer constants are whole number which have no decimal point.
Types of integer constants are:

Decimal constant:	0-------9(base 10)
Octal constant:	0-------7(base 8)
Hexa decimal constant:	0----9, A------F(base 16)

In decimal constant first digit should not be zero unlike octal constant first digit must be zero(as 076, 0127) and in hexadecimal constant first two digit should be 0x/ 0X (such as 0x24, 0x87A). By default type of integer constant is integer but if the value of integer constant is exceeds range then value represented by integer type is taken to be unsigned integer or long integer. It can also be explicitly mention integer and unsigned integer type by suffix l/L and u/U.

2.Real constant :-

It is also called floating point constant. To construct real constant we must follow the rule of ,
-real constant must have at least one digit.
-It must have a decimal point.
-It could be either positive or negative.
-Default sign is positive.

-No commas or blanks are allowed within a real constant. Ex.: +325.34,426.0,-32.76

3.Character Constant :-

Character constant represented as a single character enclosed within a single quote. These can be single digit, single special symbol or white spaces such as '9','c','$', ' ' etc. Every character constant has a unique integer like value in machine's character code as if machine using ASCII (American standard code for information interchange). Some numeric value associated with each upper and lower case alphabets and decimal integers are as:

A------------ Z	ASCII value (65-90)
a-------------z	ASCII value (97-122)
0-------------9	ASCII value (48-59)
;	ASCII value (59)

String constant

Set of characters are called string and when sequence of characters are enclosed within a double quote (it may be combination of all kind of symbols) is a string constant.

String constant has zero, one or more than one character and at the end of the string null character(\0) is automatically placed by compiler.

Some examples are "Hello" , "908", "3"," ", "A" etc.

In C although same characters are enclosed within single and double quotes it represents different meaning such as "A" and 'A' are different because first one is string attached with null character at the end but second one is character constant with its corresponding ASCII value is 65.

Symbolic constant

Symbolic constant is a name that substitute for a sequence of characters and, characters may be numeric, character or string constant. These constant are generally defined at the beginning of the program as #define name value , here name generally written in upper case.

Example :
#define MAX 10
#define CH 'b'
#define NAME "sony"

Variables

Variable is a data name which is used to store some data value or symbolic names for storing program computations and results. The value of the variable can

be change during the execution. The rule for naming the variables is same as the naming identifier.
Before used in the program it must be declared. Declaration of variables specify its name, data types and range of the value that variables can store depends upon its data types.

Syntax:
```
int a;
char c;
float f;
```

Variable initialization

When we assign any initial value to variable during the declaration, is called initialization of variables. When variable is declared but contain undefined value then it is called garbage value. The variable is initialized with the assignment operator such as

Syntax :
```
Datatype  variable_name=constant;
```

Example:
```
int a=20;
```

Expressions

An expression is a combination of variables, constants, operators and function call.

It can be arithmetic, logical and relational for example:-
```
int z= x+y          // arithmetic expression
a>b                 //relational
a==b                // logical
```

Expressions consisting entirely of constant values are called *constant expressions*.

So, the expression
121 + 17 - 110
is a constant expression because each of the terms of the expression is a constant value.

But if i were declared to be an integer variable, the expression
180 + 2 – i
would not represent a constant expression.

Operators :-

This is a symbol use to perform some operation on variables, operands or with the constant. Some operator required 2 operand to perform operation or Some required single operation.
Several operators are there those are, arithmetic operator, assignment, increment , decrement, logical, conditional, comma, size of , bitwise and others.

1. Arithmetic Operator

This operator used for numeric calculation.
These are of either Unary arithmetic operator, Binary arithmetic operator.
Unary arithmetic operator required only one operand such as +,-, ++, --,!.
Binary arithmetic operator on other hand required two operand and its operators are +(addition), -(subtraction), *(multiplication), /(division), %(modulus). But modulus cannot applied with floating point operand as well as there are no exponent operator in c.

Unary (+) and Unary (-) is different from addition and subtraction.
When both the operand are integer then it is called integer arithmetic and the result is always integer. When both the operand are floating point then it is called floating arithmetic and when operand is of integer and floating point then it is called mix type or mixed mode arithmetic . And the result is in float type.

2.Assignment Operator

A value can be stored in a variable with the use of assignment operator. The assignment operator(=) is used in assignment statement and assignment expression.
Operand on the left hand side should be variable and the operand on the right hand side should be variable or constant or any expression. When variable on the left hand side is occur on the right hand side then we can avoid by writing the compound statement.
For example,

```
int x= y;
int Sum=x+y+z;
```

3.Increment and Decrement

The Unary operator ++, --, is used as increment and decrement which acts upon single operand. Increment operator increases the value of variable by one. Similarly decrement operator decrease the value of the variable by one. And these operator can be only used with the variable, but can't use with expression and constant as ++6 or ++(x+y+z).

It again categories into prefix post fix . In the prefix the value of the variable is incremented 1st, then the new value is used, whereas in postfix the operator is written after the operand(such as m++,m--).

Example :

```
        let y=12;
        z= ++y;
        y= y+1;
        z= y;
```

Similarly in the postfix increment and decrement operator is used in the operation . And then increment and decrement is perform.

Example :

```
        let x= 5;
        y= x++;
        y=x;
        x= x+1;
```

4. Relational Operator

It is use to compared value of two expressions depending on their relation. Expression that contain relational operator is called relational expression. Here the value is assign according to true or false value.

a. (a>=b) || (b>20)
b. (b>a) && (e>b)
c. (b!=7)

5. Conditional Operator

It sometimes called as ternary operator. It requires three expressions as operand and it is represented as **(? :)**.

Syntax :

```
        exp1 ? exp2 :exp3;
```

Here exp1 is first evaluated. It is true then value return will be exp2 . If false then exp3.

Example :

```
void main()
{
        int a=10, b=2
        int s= (a>b) ? a:b;
```

```
    printf(“Max:%d”,s);
}
Output:
Max:10
```

6. Comma Operator

Comma operator is use to permit different expression to be appear in a situation where only one expression would be used. All the expression are separator by comma and are evaluated from left to right.

Example :

```
    int i, j, k, l;
    for(i=1,j=2;i<=5;j<=10;i++;j++)
```

7. Sizeof Operator

Size of operator is a Unary operator, which gives size of operand in terms of byte that occupied in the memory. An operand may be variable, constant or data type qualifier.

Generally it is used make portable program (program that can be run on different machine) .

It determines the length of entities, arrays and structures when their sizes are not known to the programmer. It is also use to allocate size of memory dynamically during execution of the program.

EXAMPLE

```
main( )
{
    int sum;
    float f;
    printf( "%d%d" ,sizeof(f), sizeof (sum) );
    printf("%d%d", sizeof(235), sizeof(‘A’));
}
```

8. Bitwise Operator

Bitwise operator permit programmer to access and manipulate of data at bit level. Various bitwise operator enlisted are

one's complement	~
bitwise AND	&
bitwise OR	|
bitwise XOR	^
left shift	<<
right shift	>>

These operators can operate on integer and character value but not on float and double.
In one's complement all 0 changes to 1 and all 1 changes to 0. In the bitwise OR its value would obtaining by 0 to 2 bits.
As the bitwise OR operator is used to set on a particular bit in a number.

9. Logical or Boolean Operator
Operator used with one or more operand and return either value zero (for false) or one (for true). The operand may be constant, variables or expressions. And the expression that combines two or more expressions is termed as logical expression.
C has three logical operators :

Operator	**Meaning**
&&	AND
||	OR
!	NOT

Where logical NOT is a unary operator and other two are binary operator. Logical AND gives result true if both the conditions are true, otherwise result is false. And logical OR gives result false if both the condition false, otherwise result is true.

Precedence and associativity of operators

Operators	**Description**	**Precedence level**	**Associativity**
()	function call	**1**	left to right
[]	array subscript		
->	arrow operator		
.	dot operator		
+	unary plus	2	right to left
-	unary minus		
++	increment		
- -	decrement		
!	logical not		
~	1's complement		
*	indirection		
&	address		
(data type)	type cast		
sizeof	size in byte		
*	multiplication	3	left to right
/	division		

%	modulus		
+	addition	4	left to right
-	subtraction		
<<	left shift	**5**	left to right
>>	right shift		
<=	less than equal to	6	left to right
>=	greater than equal to		
<	less than		
>	greater than		
==	equal to	7	left to right
!=	not equal to		
&	bitwise AND	8	left to right
^	bitwise XOR	9	left to right
|	bitwise OR	10	left to right
&&	logical AND	11	
||	logical OR	12	
?:	conditional operator	13	
=, *=, /=, %= **&=, ^=, <<=** **>>=**	assignment operator	14	right to left
,	comma operator	15	