Tokens, Data Types and operators in C

C Tokens

Similarly, the smallest individual unit in a c program is known as a token or a lexical unit. Different tokens in C can be classified as below:

- Keyword (e.g. int, for, if)
- Identifier (e.g. main, a, sum)
- Constant (e.g. 10, 3.14)
- string-literal (e.g. "hello", "Ahmad")
- operator (e.g. +, /, *)
- punctuator (e.g. (), {}, ;)

Keywords

- Keywords are predefined words in a C compiler.
- Each keyword conveys special meaning and perform a specific function in a C program.
- Since keywords are referred names for compiler, they cannot be used as variable name.

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

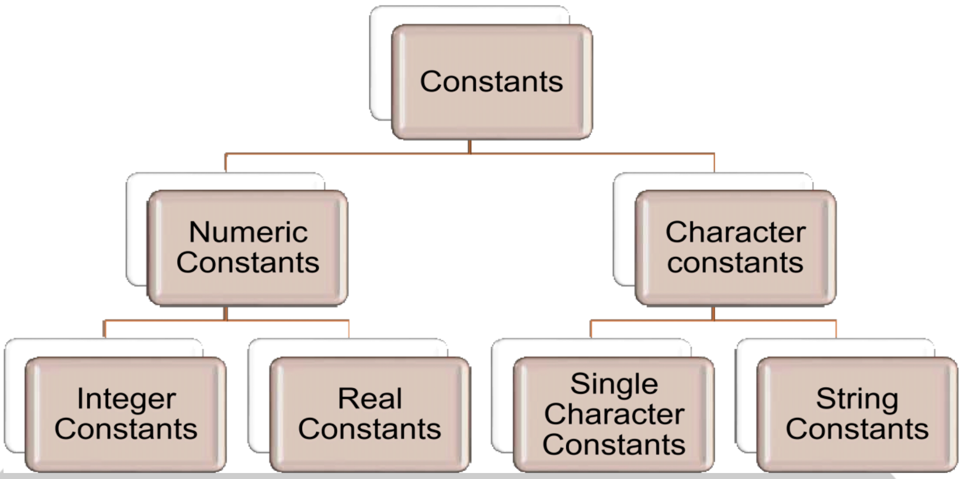
Identifier

- General terminology used for the names of variables, functions and arrays.
- These are user defined names consisting of arbitrarily long sequence of letters and digits.
- Identifiers may either a letter or the underscore(_) as a first character.
- eg. x is a name given to integer variable.

Constant

- Data items that remains same i.e. their value during do not changes during the program execution.

- Several types of C constants that are allowed in C are; integer constant, real constant, character constant, string constant and other.



Variables

- A data name used to store data values.
- May take different values at different times of program execution
- Rules for variable

1. Should not be keyword
2. Should not contain white space
3. Uppercase and lower case are significant
4. Must begin with letter (or underscore)
5. Normally should not be more than eight characters

String Literal

- Sequence of characters enclosed within double quotes. For example, “hello” , “abc”.
- 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.

Operators

- C operators are symbols that trigger an action and perform operation when applied to C variables and other objects.

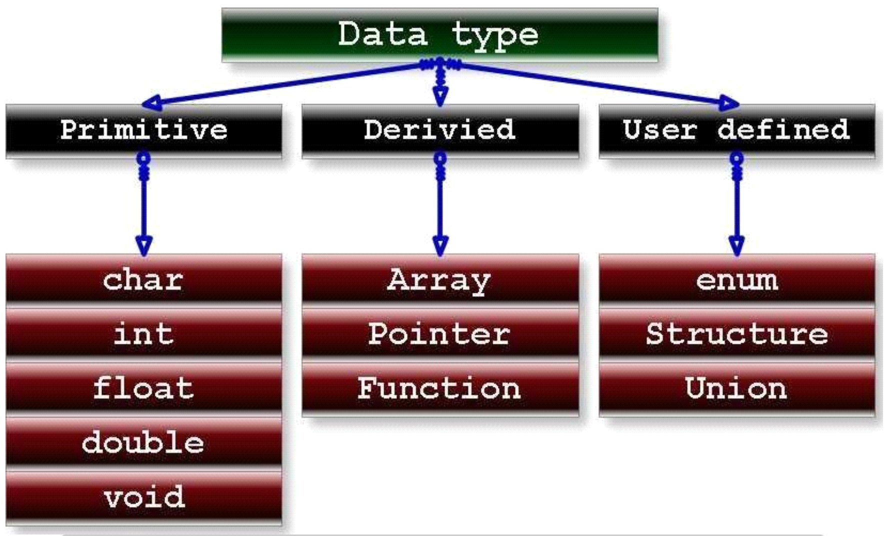
- The data items on which operators act upon are called operands.
- Operators can be classified as follows:
  - *Unary Operators*: Those operators that require only single operand to act upon are known as unary operators.
  - *Binary Operators*: Those operators that require two operands to act upon are called binary operators.
  - *Ternary Operators*: These operators requires three operands to act upon. (Details will be covered in next week)

Punctuator

- It has some special meaning
- Thus, cannot be used for some other purpose.
- [] () {} , ; : * … = #
- **Braces{}:** These opening and ending curly braces marks the start and end of a block of code containing more than one executable statement.
- **Parentheses():** These special symbols are used to indicate function calls and function parameters.
- **Brackets[]:** Opening and closing brackets are used as array element reference. These
  indicate single and multidimensional subscripts.

Data Types

- C data types are defined as the data storage format that a variable can store a data to perform a specific operation.
- Data types are used to define a variable before to use in a program.
- Size of variable, constant and array are determined by data types.



Type	Size (bits)	Size (bytes)	Range
char	8	1	-128 to 127
unsigned char	8	1	0 to 255
int	16	2	-2^{15} to $2^{15}-1$
unsigned int	16	2	0 to $2^{16}-1$
short int	8	1	-128 to 127
unsigned short int	8	1	0 to 255
long int	32	4	-2^{31} to $2^{31}-1$
unsigned long int	32	4	0 to $2^{32}-1$
float	32	4	3.4E-38 to 3.4E+38
double	64	8	1.7E-308 to 1.7E+308
long double	80	10	3.4E-4932 to 1.1E+4932

Now, write the following program, compile and execute it to understand the above explained topics.

```
    #include<stdio.h>
     int main() { int
    a,b,sum;
    printf("Enter two numbers to add\n");
    scanf("%d%d",&a,&b); sum = a + b;
    printf("Sum of entered numbers = %d\n",sum); return 0;
}
```

Type Casting in C Data Types

- Type casting is a way to convert a variable from one data type to another data type.
- For example, if you want to store a 'long' value into a simple integer then you can type cast 'long' to 'int'.
- Implicit Type conversion · Explicit Type Conversion
- You can convert the values from one type to another explicitly using the cast operator as follows –

  (type_name) expression

-
```
#include <stdio.h>
        int main ()
        {
        float x;     x =  7/5 ;
        printf("%f",x);
        return 0;
        }
```

- In the above C program, 7/5 alone will produce integer value as 1.
- So, type cast is done before division to retain float value (1.4) as follows;
-
```
#include <stdio.h>
      int main ()
      {
       float x;
x = (float) 7/5;     printf("%f",x);
      }
```

- The above code will produce the output as 1.400000 .

OPERATORS AND EXPRESSIONS

Definition:

"An operator is a symbol (+,-,*,/) that directs the computer to perform certain mathematical or logical manipulations and is usually used to manipulate data and variables.

Operators in C:

- **Arithmetic operators**
- **Relational operators**
- **Logical operators**
- **Assignment operators**
- **Increment and decrement operators**
- **Conditional operators**
- **Bitwise operators**
- **Special operators**

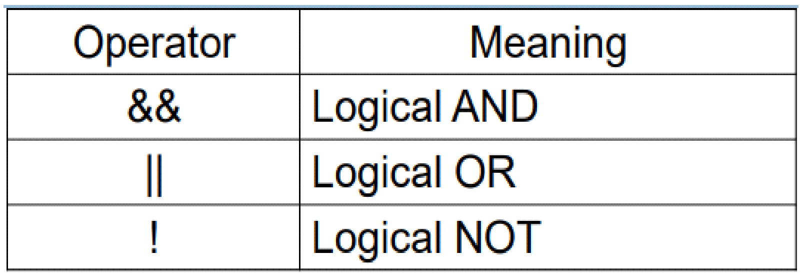
ARITHMETIC OPERATORS :

Operator	example	Meaning
+	a + b	Addition –unary
-	a – b	Subtraction- unary
*	a * b	Multiplication
/	a / b	Division
%	a % b	Modulo division- remainder

RELATIONAL OPERATOR :

Operator	Meaning
<	Is less than
<=	Is less than or equal to
>	Is greater than
>=	Is greater than or equal to
==	Equal to
!=	Not equal to

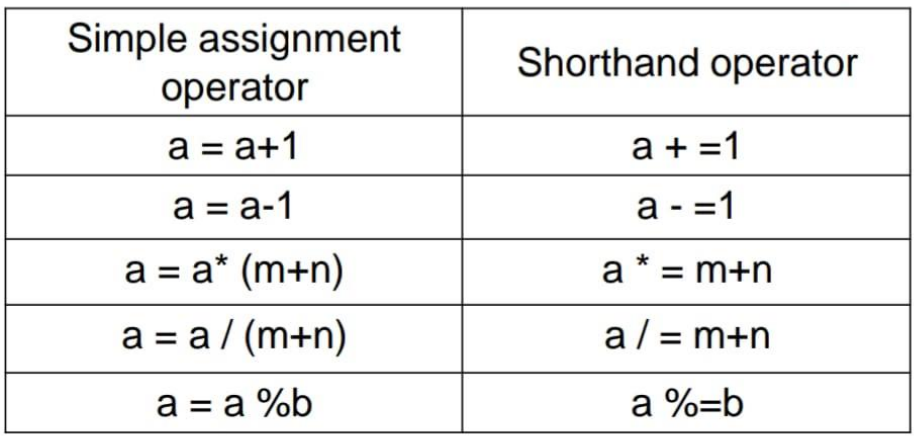
LOGICAL OPERATOR :



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

Logical expression or a compound relational expression. An expression that combines two or more relational expressions.

Assignment Operator:



Simple assignment operator	Shorthand operator
a = a+1	a + =1
a = a-1	a - =1
a = a* (m+n)	a * = m+n
a = a / (m+n)	a / = m+n
a = a %b	a %=b

Increment and Decrement Operator:

C supports 2 useful operators namely

1. Increment ++

2. Decrement – operators

The ++ operator adds a value 1 to the operand

The – operator subtracts 1 from the operand

++a or a++

--a or a—

RULES FOR ++ &-- operator:

1. These require variables as their operand

2. When postfix either ++ or – is used with the variable in a given expression, the expression is evaluated first and then it is incremented or decremented by one.

3. When prefix either ++ or – is used with the variable in a given expression, it is incremented or decremented by one first and then the expression is evaluated with the new value.

Bit-wise Operator:

Operator	Meaning
&	Bitwise AND
|	Bitwise OR
^	Bitwise exclusive OR
<<	Shift left
>>	Shift right

Special Operator:

1. Comma operator

2. size-of operator

3. Pointer operator

4. Member selection operator

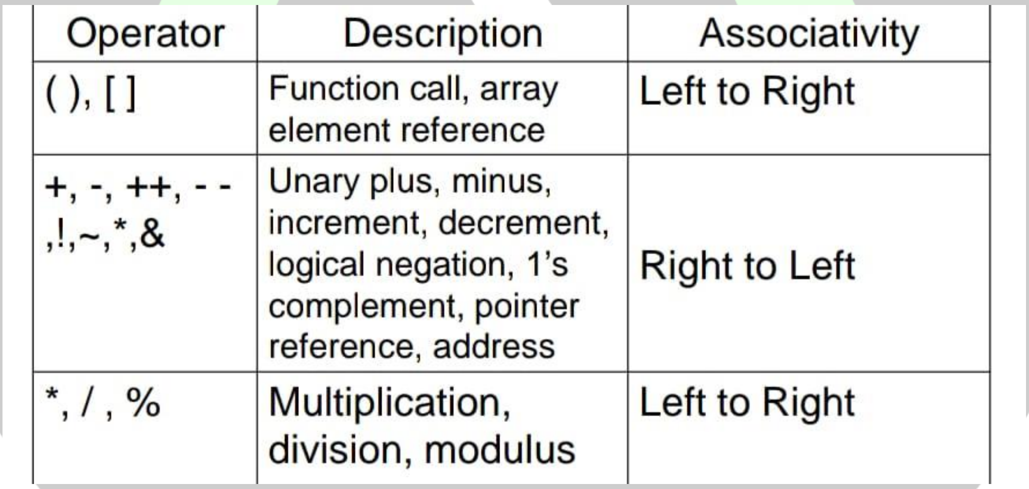
Arithmetic Expression:

Algebraic expression	C expression
axb-c	a*b-c
(m+n)(x+y)	(m+n)*(x+y)
$\left[\frac{ab}{c}\right]$	a*b/c
$3x^2+2x+1$	3*x*x+2*x+1
$\frac{a}{b}$	a/b
S= $\frac{a+b+c}{2}$	S=(a+b+c)/2

Rules for evaluation of expression:

1. First parenthesized sub expression from left to right are evaluated.
2. If parentheses are nested, the evaluation begins with the innermost sub expression.
3. The precedence rule is applied in determining the order of application of operators in evaluating sub expressions.
4. The associatively rule is applied when 2 or more operators of the same precedence level appear in a sub expression.
5. Arithmetic expressions are evaluated from left to right using the rules of precedence.
6. When parentheses are used, the expressions within parentheses assume highest priority.

Hierarchy of Operators:



Operator	Description	Associativity
( ), [ ]	Function call, array element reference	Left to Right
+, -, ++, - - ,!,~,*,&	Unary plus, minus, increment, decrement, logical negation, 1's complement, pointer reference, address	Right to Left
*, / , %	Multiplication, division, modulus	Left to Right