MODULE II

PROGRAM BASICS

Basic structure of C program: Character set, Tokens, Identifiers in C, Variables and Data Types, Constants, Console IO Operations, printf and scanf .
Operators and Expressions: Expressions and Arithmetic Operators, Relational and Logical Operators, Conditional operator, size of operator, Assignment operators and Bitwise Operators. Operators Precedence
Control Flow Statements: If Statement, Switch Statement, Unconditional Branching using goto statement, While Loop, Do While Loop, For Loop, Break and Continue statements.(Simple programs covering control flow)

What is a language?

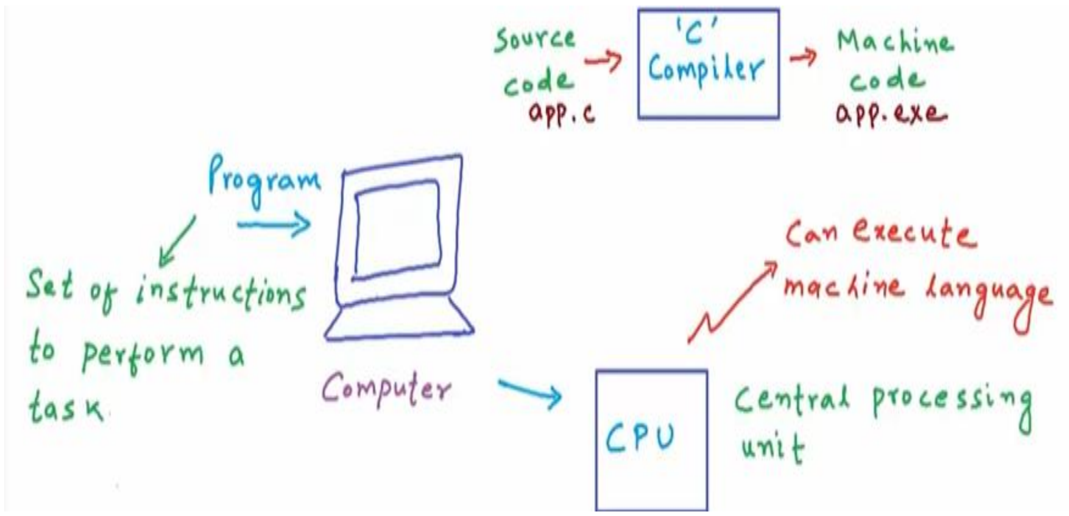
Language is a system of conventional spoken, manual (signed), or written symbols by means of which human beings express themselves.

What is program?

A computer program is a collection of instructions that can be executed by a computer to perform a specific task.

What is a programming language?

A computer programming language is a language used to write computer programs, which involves a computer performing some kind of computation.

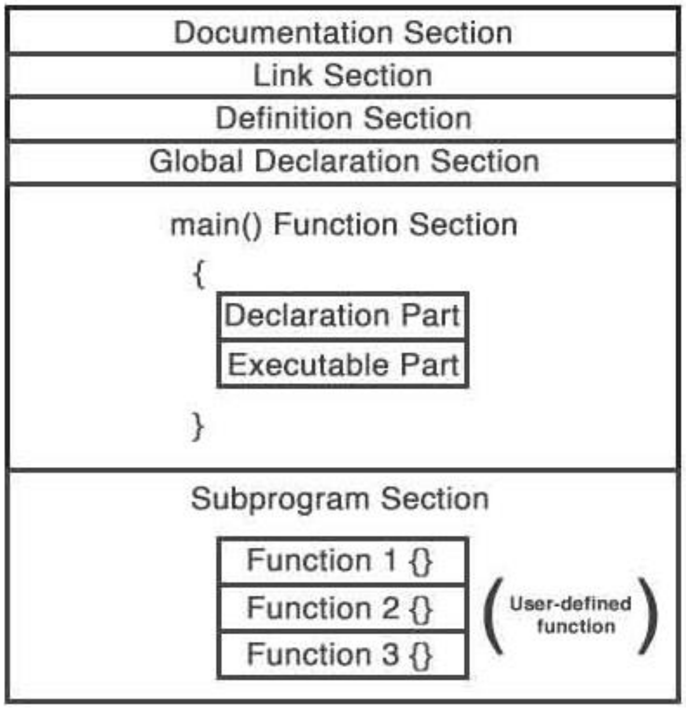


C-Language

C programming is considered as the base for other programming languages, that is why it is known as mother language.

Developed in 1972 by Dennis Ritchie at bell laboratories of AT&T (American Telephone & Telegraph), located in the U.S.A.

Basic structure of C program



Documentation Section: The documentation section is the part of the program where the programmer gives the details associated with the program.

Example

```
/*File Name: Helloworld.c
  Author: Aleena
 Date: 09/08/2019
  Description: a program to display hello world
*/
```

Link Section :This part of the code is used to declare all the header files that will be used in the program. This leads to the compiler being told to link the header files to the system libraries.

Example : #include<stdio.h>

Definition Section : In this section, we define different constants. The keyword define is used in this part.

Example : #define PI=3.14

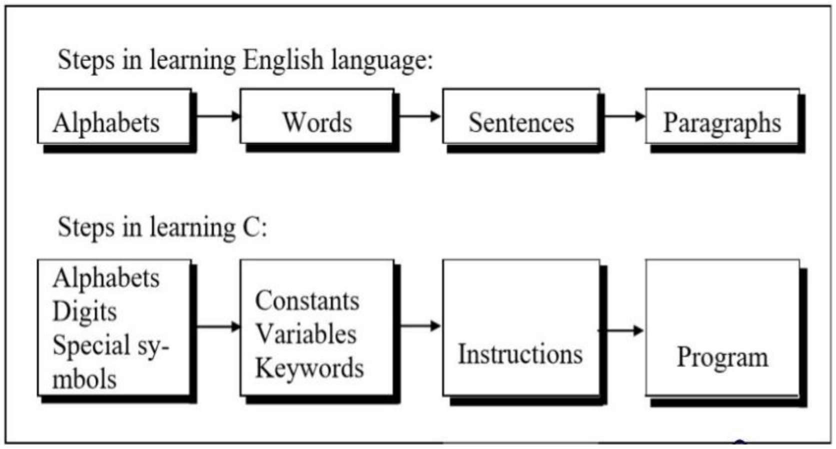
Global Declaration Section : This part of the code is the part where the global variables are declared. The user-defined functions are also declared in this part of the code.

Main Function Section : Every C-programs needs to have the main function. Function main() is the starting point of every C program. Each main function contains 2 parts.

- A declaration part and an Execution part

The declaration part is the part where all the variables are declared. Execution part contains the statements.

Sub Program Section : All the user-defined functions are defined in this section of the program. Functions defined by the user are called user defined functions.


Steps in learning English language:
Alphabets
Words
Sentences
Paragraphs
Steps in learning C:
Alphabets Digits Special sy-mbols
Constants Variables Keywords
Instructions
Program

<u>C- CHARACTER SET</u>

Like any other language C has its own vocabulary and grammar.

As every language contains a set of characters used to construct words, statements, etc, C language also has a set of characters.

C language character set contains the following set of characters.

- Alphabets
- Digits
- Special Symbols
- White Spaces

Alphabets

C language supports all the alphabets from the English language.

Lower and upper case letters together support 52 alphabets.

Lower case letters - a to z

Upper case letters - A to Z

Digits

C language supports 10 digits which are used to construct numerical values in C language.

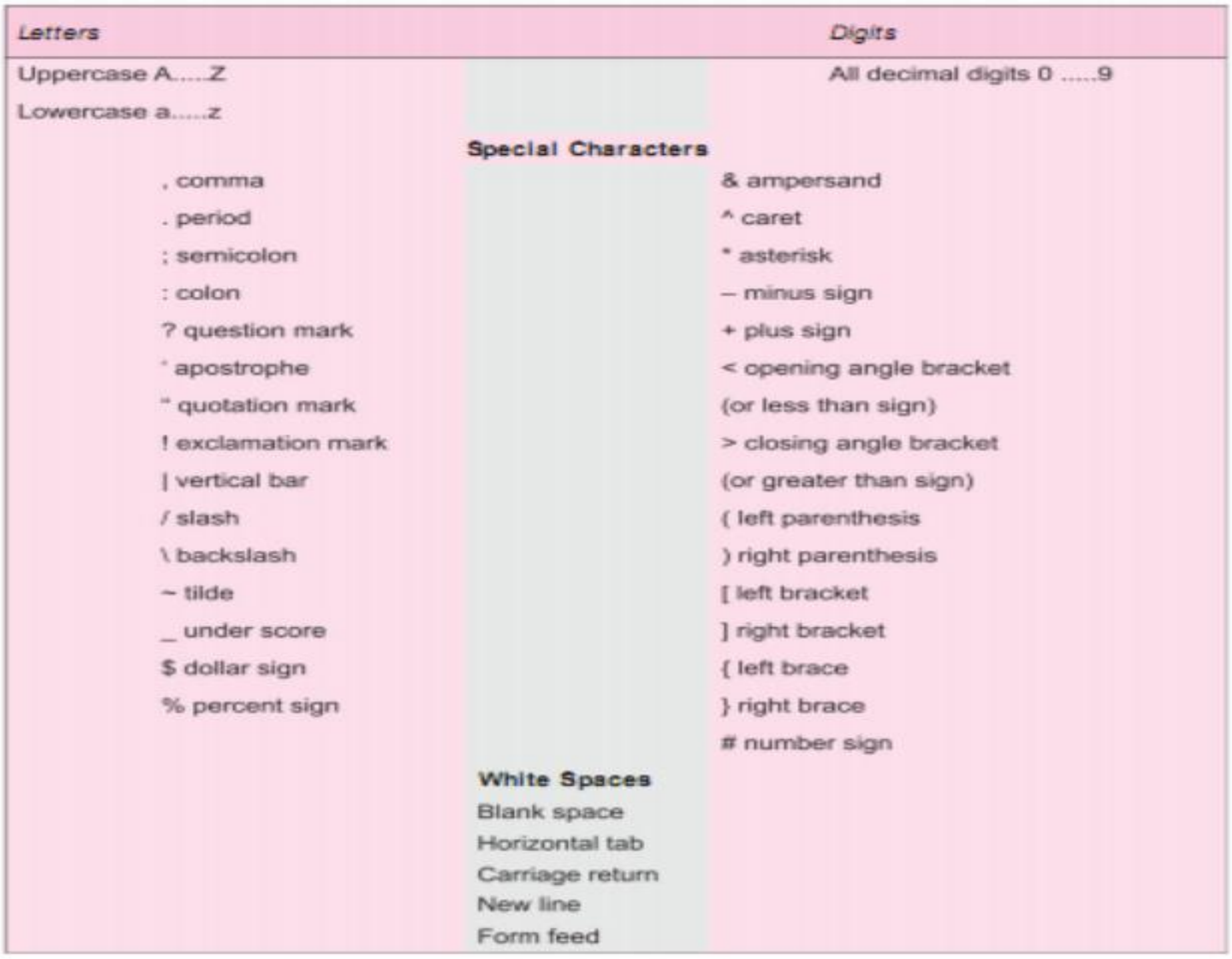
Digits - 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

Special Symbols

Special Symbols - ~ @ # $ % ^ & * ( ) _ - + = { } [ ] ; : ' " / ? . > ,< \ | etc.,

White Spaces

Blank space, horizontal tab, newline character etc.



Letters		Digits
Uppercase A.....Z		All decimal digits 0 9
Lowercase a.....z		
	Special Characters	
, comma		& ampersand
. period		^ caret
; semicolon		* asterisk
: colon		– minus sign
? question mark		+ plus sign
' apostrophe		< opening angle bracket
" quotation mark		(or less than sign)
! exclamation mark		> closing angle bracket
| vertical bar		(or greater than sign)
/ slash		(left parenthesis
\ backslash		) right parenthesis
~ tilde		[left bracket
_ under score		] right bracket
$ dollar sign		{ left brace
% percent sign		} right brace
		# number sign
	White Spaces	
	Blank space	
	Horizontal tab	
	Carriage return	
	New line	
	Form feed	

C PROGRAM TOKENS

Every C program is a collection of instructions and every instruction is a collection of some individual units.

Every smallest individual unit of a C program is called token.

Every instruction in a C program is a collection of tokens.

Tokens are the basic building blocks of a C program.

Tokens include:

- Keywords
- Identifiers
- Operators
- Special Symbols
- Constants
- Strings
- Data values

Keywords

Keywords are specific reserved words in C each of which has a specific feature associated with it.

All key words have fixed meaning and these meanings cannot be changed

Keywords are always in lowercase.

There are total of 32 keywords in C:

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

Identifiers:

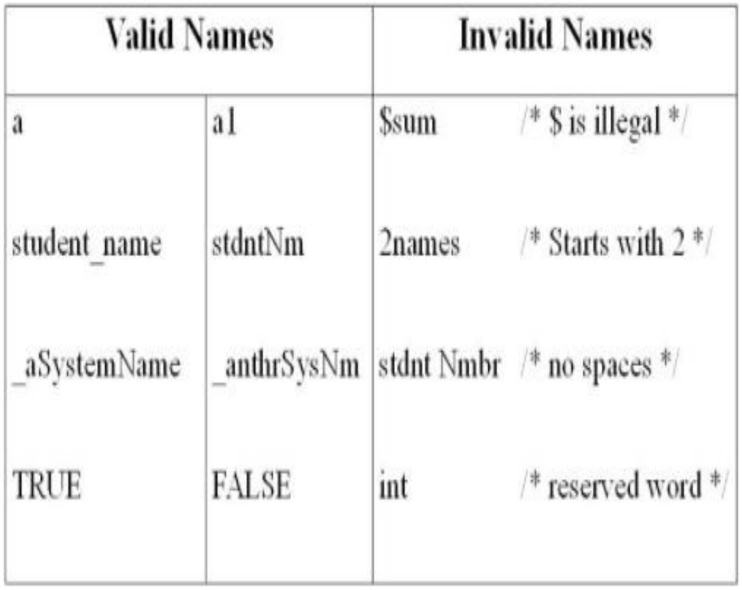
Every C word is classified as either a keyword or an identifier.

Identifiers refer to the names of variables; functions and arrays.

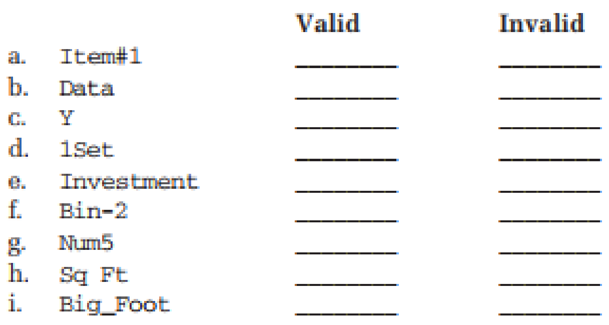
These are user-defined names and consist of a sequence of letters and digits with the letter as a first character.

Rules:

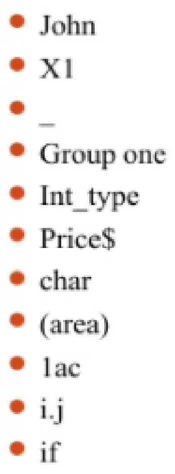
1. May consists of alphabets, underscore and digits
2. First character should not be a digit.
3. Name must not contain blank space.
4. Name can have maximum upto 31 characters
5. No keyword should be used as identifiers
6. Case sensitive



Valid Names		Invalid Names	
a	a1	$sum	/* $ is illegal */
student_name	stdntNm	2names	/* Starts with 2 */
_aSystemName	_anthrSysNm	stdnt Nmbr	/* no spaces */
TRUE	FALSE	int	/* reserved word */



		Valid	Invalid
a.	Item#1	________	________
b.	Data	________	________
c.	Y	________	________
d.	1Set	________	________
e.	Investment	________	________
f.	Bin-2	________	________
g.	Num5	________	________
h.	Sq Ft	________	________
i.	Big_Foot	________	________

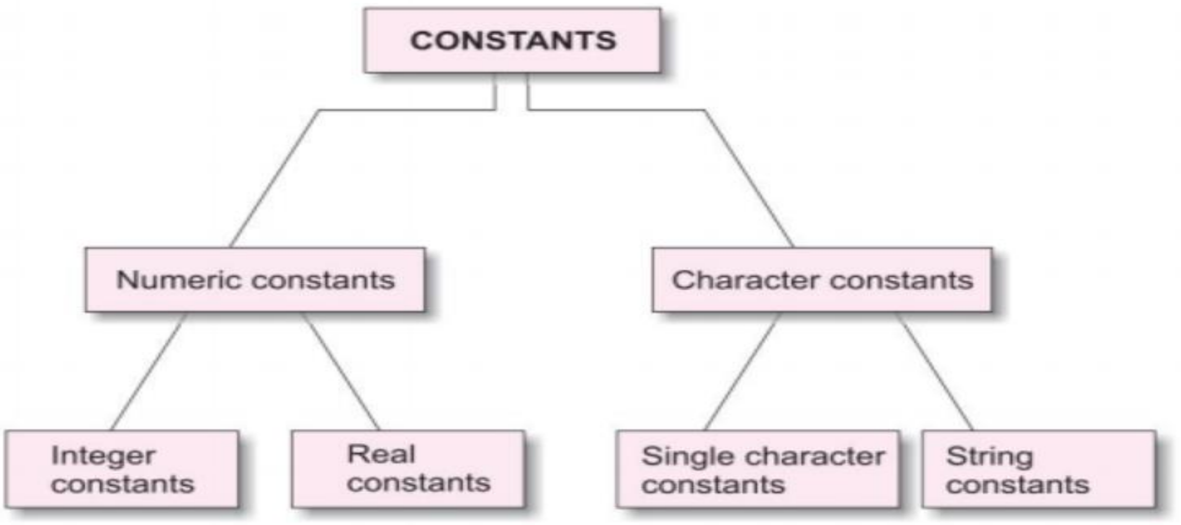


- John
- X1
- _
- Group one
- Int_type
- Price$
- char
- (area)
- 1ac
- i.j
- if

CONSTANTS

Constants refer to fixed values that the **program** may not alter during its execution.

The types of constants are



Integer constant: These are whole numbers without any fractional part. It must have at least one digit and must not contain any decimal point. It can be either positive or negative.

Examples are 426,+200,-760

Real or Floating point constants : These numbers have fractional part. These numbers are shown in decimal notation

Examples are10.3, 20.2, 450.6

Single Character Constants: A Single character constant (character constant) is one character enclosed in single quotes.

Examples are ‘A’, ’5’ ,’=’ .

A character constant have corresponding ASCII (American Standard Code for Information Interchange) values. For example ASCII value of ‘A’ is 65 and ASCII value of ‘a’ is 97.

String constant: Multiple character constants are treated as

string constant.

A string constant is a sequence of characters surrounded by double quotes.

Examples are “abcd”, ”seena”.

Each string constant is by default (automatically) added with a special character ’\0’ which makes the end of a string. Thus the size of a string is

Number of characters + null character (‘\0’)

For example “abc” size is 4. Thus “abc” will be automatically represented as “abc\0” in the memory.’\0’ is an end-of- string marker

Symbolic Constants

A symbolic constant is a name given to some numeric constant, or a character constant or string constant, or any other constants.

When a program is compiled, each occurrence of a symbolic constant is replaced by its corresponding character sequence.

Symbolic constants are usually defined at the beginning of a program.

Syntax: #define name value

Example: #define PI 3.14

```
#define MAX 500
```

PI,MAX are symbolic constants

VARIABLES

A variable is a data name that may be used to store a data value.

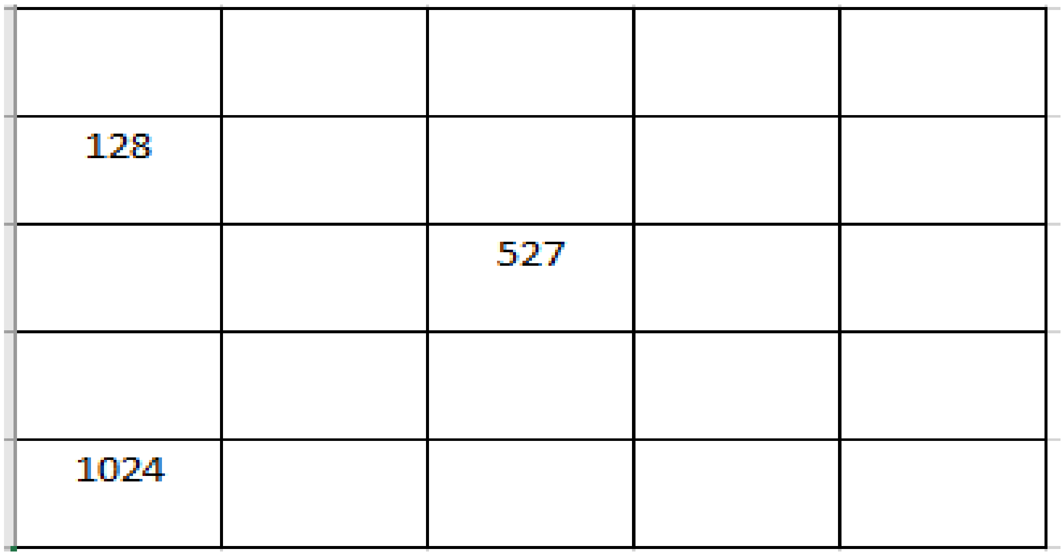
A variable may take different values at different times during execution.

Each variable has a specific storage location in memory where its value is stored. The variables are called symbolic variables.

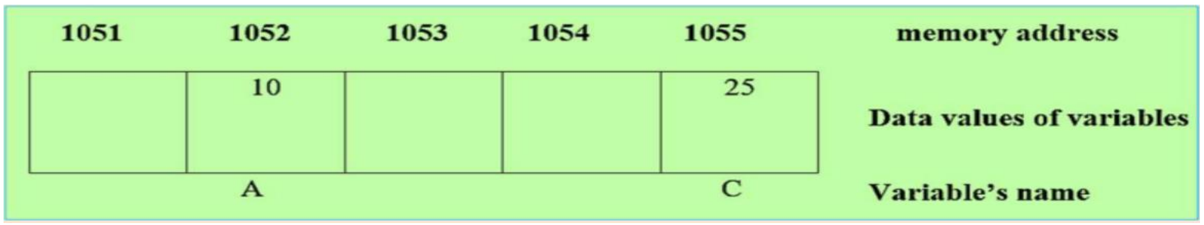
There are two values associated with a symbolic variable.

Data value : value stored at some location in memory.

Location value : This is the address in memory at which its data value is stored.



128				
		527		
1024				



Variable A : data-value of A=10

location-value of A=1052

Variable C : data-value of C=25

location-value of C=1055

A variable name can be chosen by the programmer in a meaningful way so as to reflect its function or nature in the program.

Some examples are –

Average, height, Total, counter_1, class_strength, house_name,etc.

DATA TYPES

Data types in the C programming language are used to specify what kind of value can be stored in a variable.

The memory size and type of the value of a variable are determined by the variable data type.

In the C programming language, data types are classified as follows

Fundamental (primary) Data types

Derived Data types

User defined data types

Fundamental (primary) data types

1.**int :** for integers (2 byte memory space allocates in memory)

2.**char :** for characters(1 byte memory space allocates in memory)

3.**float :** for single precision floating point numbers(4 byte memory space allocates in memory)

4.**double** : for double precision floating point numbers(8 byte memory space allocates in memory)

5.**void:** for empty set of values and non-returning functions. The void type has no value.

DATA TYPE	MEMORY(BYTES)	FORMAT SPECIFIER

int	2	%d
char	1	%c
float	4	%f
double	8	%lf

Derived data types - Derived data types are constructed from fundamental data types.

examples – arrays, functions, etc

User defined data types - It allows users to define an identifier that would represent an existing data type.

examples – structure, union,etc

Variable Declaration

After designing suitable variable names, we must declare them to the compiler.

Declaration done two things:

- It tells the compiler what the variable name is
- It specifies what type of data the variable will hold

All variables must be declared before they can appear in executable statements.

A declaration associates a group of variables with a specific data type.

A declaration consists of a data type, followed by one or more variable names, ending with a semicolon.

Syntax : datatype variable-name v1,v2,….vn;

Example : int a,b,c ;
float d;
char e;

Variable Initialization

The process of giving initial values to variables is called initialization.

Example : int x=10;
float n=22.889;
char answer='y';

POINTS TO REMEMBER

- Every statement in C should end with a semicolon.
- In C everything is written in lower case. However upper case letters used for symbolic names representing constants.
- #include is a pre-processor directive.
- stdio.h is a header file, for standard input output functions. It activate keyboard and monitor.
- Single line comment is represented using //
- Multiple line comment is represented using /*……………..*/
- Comment lines are not executable statements and therefore anything between/*and*/ is ignored by the compiler.
- Execution begins from main ( )
- main has no arguments(or Parameters)
- Every program must have exactly one main function.
- C permits different forms of main statements.
- They are

```
main ( )
int main ( )
void main ( )
main (void)
void main (void)
int main (void)
```

<u>PREPROCESSOR DIRECTIVES</u>

The pre-processor is a program that processes the source code before it passes through the compiler.

Before the source code passes through the compiler, it is examined by the pre-processor for any pre-processor directives.

If there are any, appropriate actions are taken and then the source program is handed over to the compiler. Some categories of directives are

#define	-- macro substitution
#include	-- File inclusion directives

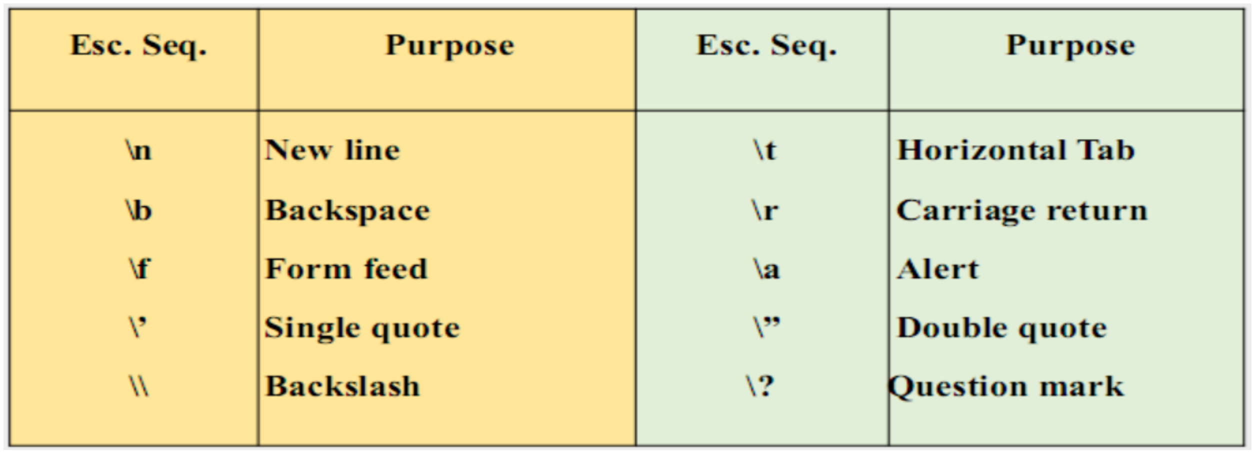
Syntax

```
                    #include<file name>
Eg:  #include<stdio.h>
          #include<stdlib.h>
          #include<math.h>
```

ESCAPE SEQUENCES (BACKSLASH CHARACTER CONSTANTS)

These are special constants that are used in output functions.

For example ‘\n’ stands for newline character



Esc. Seq.	Purpose	Esc. Seq.	Purpose
\n	New line	\t	Horizontal Tab
\b	Backspace	\r	Carriage return
\f	Form feed	\a	Alert
\’	Single quote	\”	Double quote
\\	Backslash	\?	Question mark

VARIOUS INPUT/OUTPUT FUNCTIONS

These functions are used for input and output in C language.

1) printf()

The printf() function is used for output.

It prints the given statement to the console.

Syntax :

```
                    printf("format string",argument_list);
```

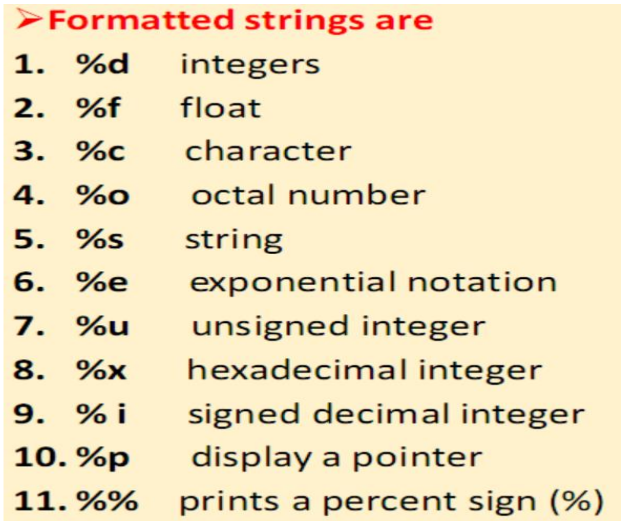
Examples :

printf (“%d”, a);

printf(“the result is :%d”,sum);

printf(“the value of A is %d\n the value of B is %d”,a,b);

printf (“S2 CSE PRGRAMMING IN C”);



➢Formatted strings are

1.	%d	integers
2.	%f	float
3.	%c	character
4.	%o	octal number
5.	%s	string
6.	%e	exponential notation
7.	%u	unsigned integer
8.	%x	hexadecimal integer
9.	% i	signed decimal integer
10.	%p	display a pointer
11.	%%	prints a percent sign (%)

2. scanf ( )

The scanf() function function is used to take input from the user.

This function reads formatted input from the standard input such as keyboards.

scanf ( ) means “scan formatted” .

Syntax :

```
        scanf (“formatted string”, addressed variable);
```

```
Eg:         scanf (“%d”, &a);
                        scanf (“%d%f”,&num1,&num2);
```

3. getchar ( )

getchar () function reads a single character from standard input.

It takes no parameters and its returned value is the input character.

Syntax

```
                variable name=getchar( );
```

```
Eg: char c;
        printf(“Enter a character”);
c=getchar ( );
```

4. putchar( )

It displays a single character on the screen. This function takes one argument, which is the character to be sent.

It also returns this character as its result.

Syntax

```
                    putchar(variable_name);
Eg:   char ans='y'
                    putchar(ans);
```

5. gets( ) :receives a string from the keyboard.

```
                              gets(variable_name);
```

6. puts ( ) : Outputs a string to the screen

```
                              puts (variable_name);
Eg:   char vehicle [40];
          printf("Enter your vehicle name");
          gets(vehicle);
puts(vehicle);
```

These lines use the gets and puts to transfer the line of text into and out of the computer.

When this program is executed, it will give the same result as that with scanf and printf function for input and output of given variable or array.

Sample programs

```
#include <stdio.h>
void main ()
{
          printf("Hello welcome to C programming \n");
}
```

Output

```
                    Hello welcome to C programming
```

```
#include <stdio.h>
void main ()
{
        printf("Hello welcome to\n C programming \n");
}
```

Output

```
                Hello welcome to
                C programming
```

Program to read two integers

```
#include <stdio.h>
   /*header files*/

void main ( )              /* main
   function*/
```

OPERATORS AND EXPRESSIONS

Expression in C Program

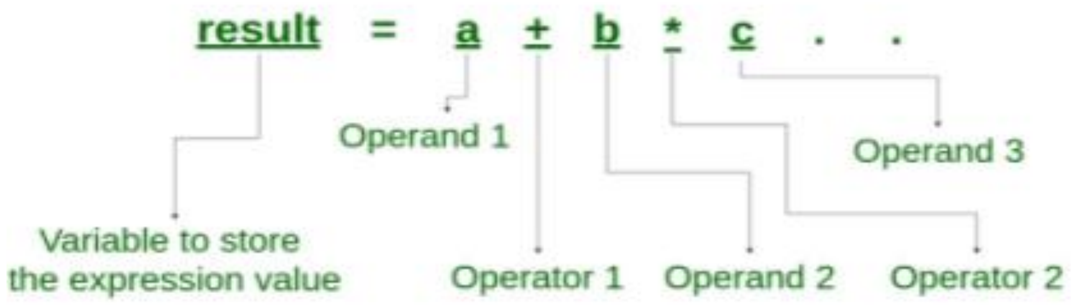
An expression is a combination of operators and operands which reduces to a single value.

An Operation is performed on a data item which is called an Operand.

An Operator indicates an operation to be performed on data.

Example :

```
        result=a+b*c
```



C OPERATORS

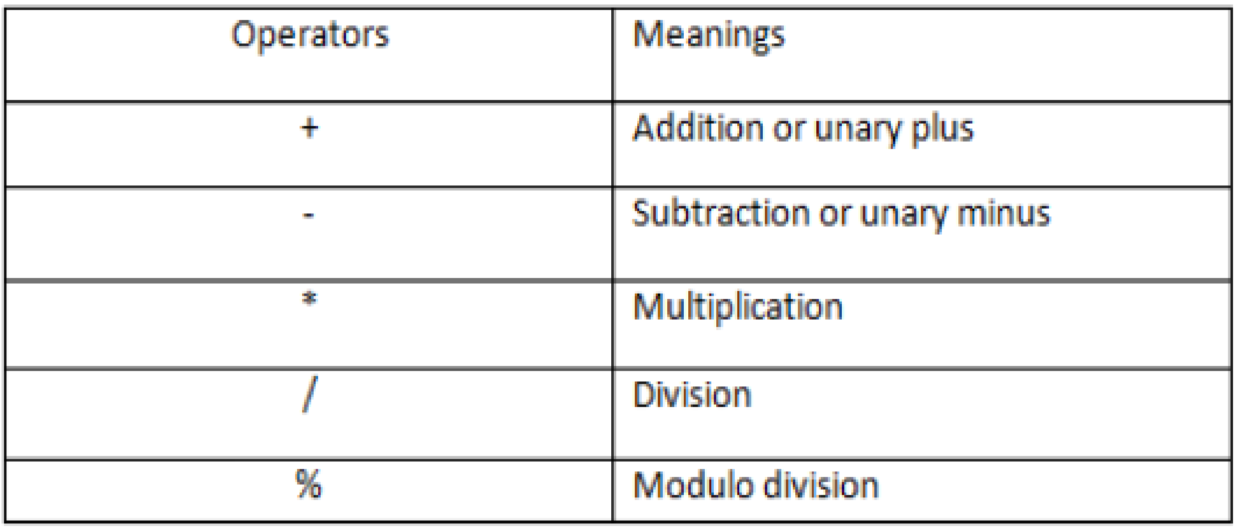
An operator is a symbol used to perform arithmetic and logical operations in a program.

C language is rich in built-in operators and provides the following types of operators –

1. Arithmetic Operators
2. Relational Operators
3. Logical Operators
4. Increment & Decrement Operators
5. Assignment Operators
6. Bitwise Operators
7. Conditional Operator
8. Special Operators

Arithmetic Operators

An arithmetic operator performs mathematical operations such as addition, subtraction, multiplication, division etc on numerical values (constants and variables).



Operators	Meanings
+	Addition or unary plus
-	Subtraction or unary minus
*	Multiplication
/	Division
%	Modulo division

Example:

a=10 and b=3

S=a+b;   S=10+3=13

S=a-b;   S=10-3=7

S=a*b;   S=10*3=30

S=a/b;   S=10/3=3

S=a%b;   S=10%3=1

Relational operators

Relational operators are used to compare two quantities and take certain decision depending on that relation.

If the relation is true, it returns 1; if the relation is false, it returns 0

Relational operators are used in loops and decision making statements .

Operators	Meanings
<	Is less than
<=	Is less than or equal to
>	Is greater than
>=	Is greater than or equal to
==	Is equal to
!=	Is not equal to

Example:

Let a=10 , b=5 and c=5

a == b	evaluated to 0
a > b	evaluated to 1
a < b	evaluated to 0
a != b	evaluated to 1
a >= b	evaluated to 1
a <= b	evaluated to 0

Logical Operators

Logical operators are used when more than one conditions are to be tested and based on that result, decisions have to be made.

C programming offers three logical operators.

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

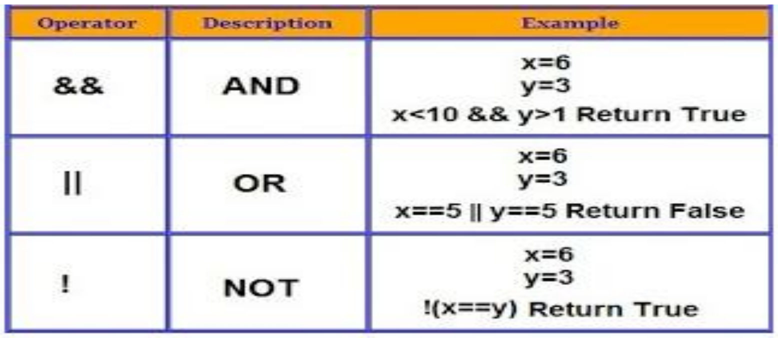
Example:

Let age=55 , salary=60,000

(age=>50) && (salary>50,000)

(age<60)||(salary<35,000)

!(age>75)



Operator	Description	Example

&&	AND	x=6
		y=3
		x<10 && y>1 Return True

||	OR	x=6
		y=3
		x==5 || y==5 Return False

!	NOT	x=6
		y=3
		!(x==y) Return True

Assignment operators

Assignment operators are used to assign result of an expression to a variable.

'=' is the assignment operator in C.

Syntax :

variable_name = expression;

eg: area=length*breadth;

num1=10;

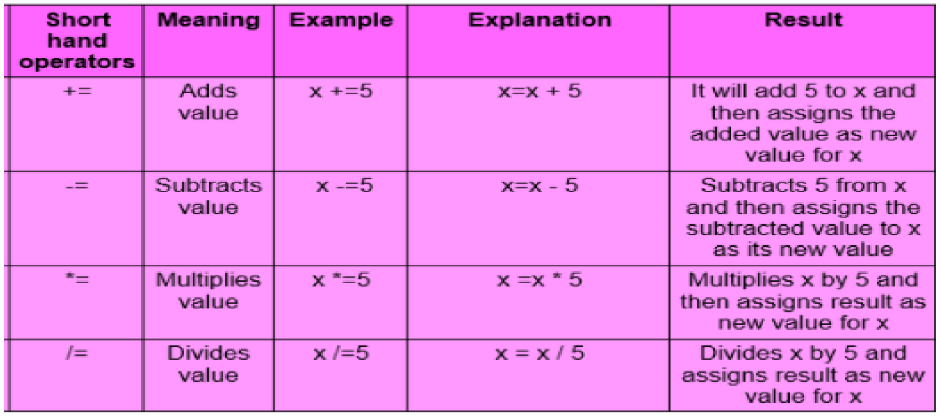
C also allows the use of shorthand assignment operators.

Shorthand operators take the form:

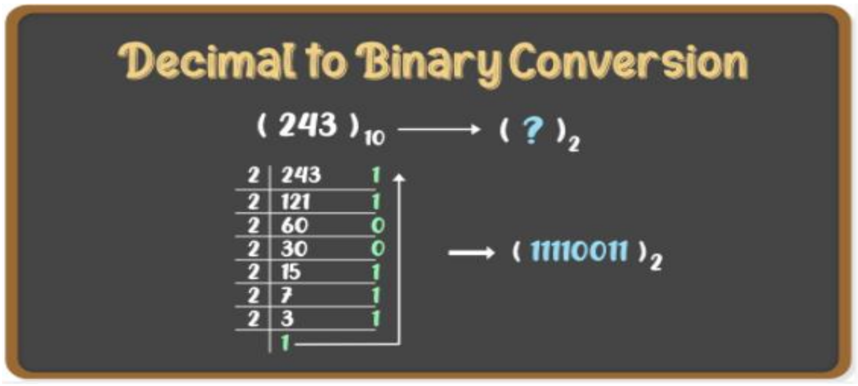
var op = exp;

var→variable, op→arithmetic operator,exp→ expression

Eg : a+=1 same as a=a+1;



Short hand operators	Meaning	Example	Explanation	Result
+=	Adds value	x +=5	x=x + 5	It will add 5 to x and then assigns the added value as new value for x
-=	Subtracts value	x -=5	x=x - 5	Subtracts 5 from x and then assigns the subtracted value to x as its new value
*=	Multiplies value	x *=5	x =x * 5	Multiplies x by 5 and then assigns result as new value for x
/=	Divides value	x /=5	x = x / 5	Divides x by 5 and assigns result as new value for x

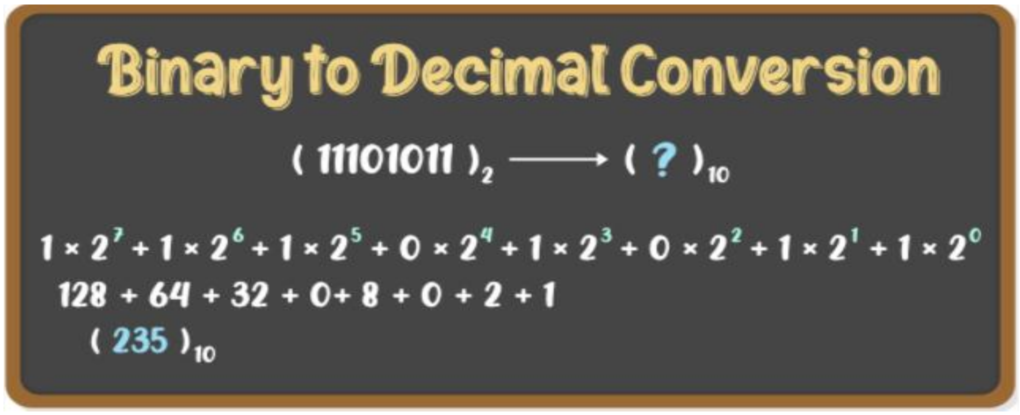


Decimal to Binary Conversion

$(243)_{10} \longrightarrow (?)_2$

2	243	1
2	121	1
2	60	0
2	30	0
2	15	1
2	7	1
2	3	1
	1	

$\longrightarrow (11110011)_2$



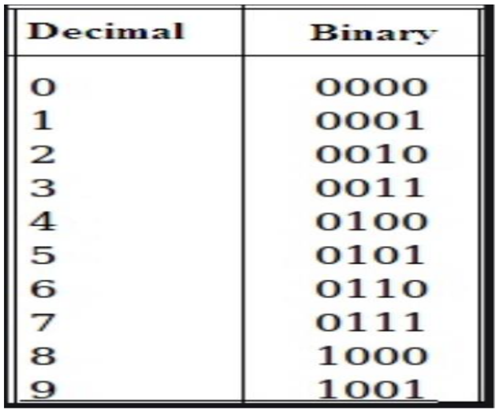
Binary to Decimal Conversion

$(11101011)_2 \longrightarrow (?)_{10}$

$1 \times 2^7 + 1 \times 2^6 + 1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$

$128 + 64 + 32 + 0 + 8 + 0 + 2 + 1$

$(235)_{10}$



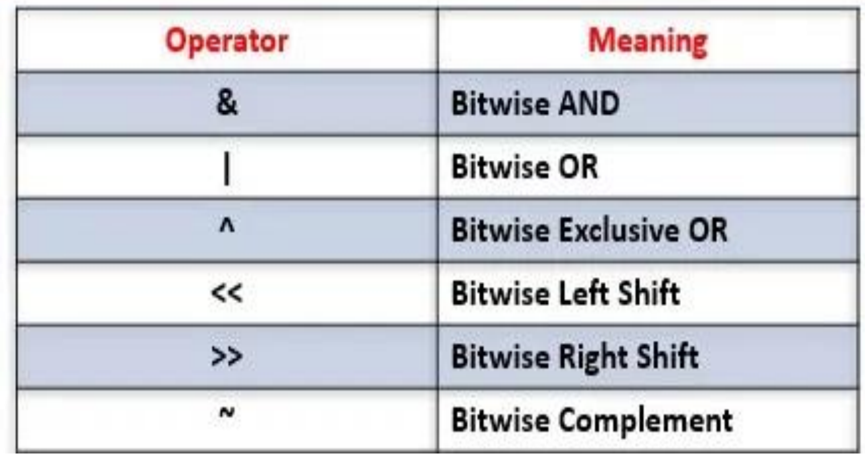
Decimal	Binary
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Bitwise Operators

Bitwise operators are used for testing the bits or shifting them left or right.

To perform bit-level operations bitwise operators are used

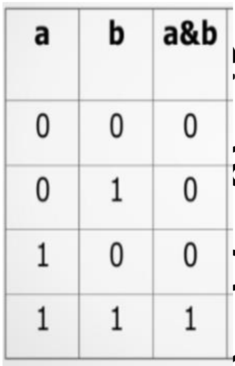
The bitwise operators available in C are:



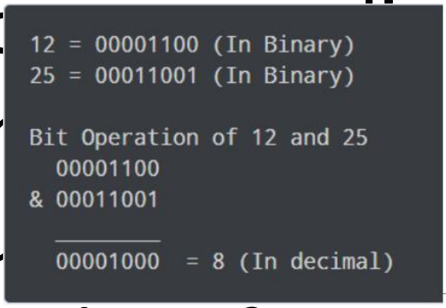
Operator	Meaning
&	Bitwise AND
|	Bitwise OR
^	Bitwise Exclusive OR
<<	Bitwise Left Shift
>>	Bitwise Right Shift
~	Bitwise Complement

Bitwise AND Operator (&)

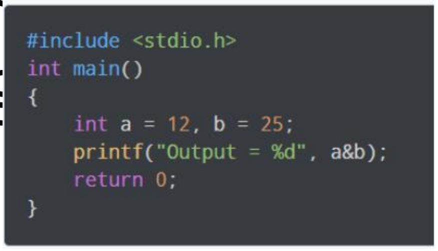
- The output of bitwise AND is 1



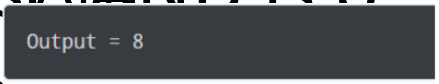
a	b	a&b
0	0	0
0	1	0
1	0	0
1	1	1



```
12 = 00001100 (In Binary)
25 = 00011001 (In Binary)

Bit Operation of 12 and 25
  00001100
& 00011001
  ________
  00001000  = 8 (In decimal)
```



```
#include <stdio.h>
int main()
{
    int a = 12, b = 25;
    printf("Output = %d", a&b);
    return 0;
}
```



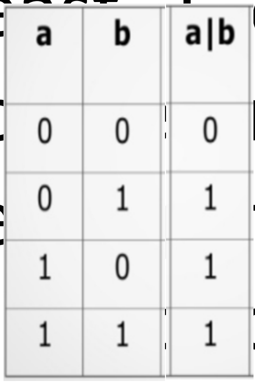
```
Output = 8
```

se AND is 0

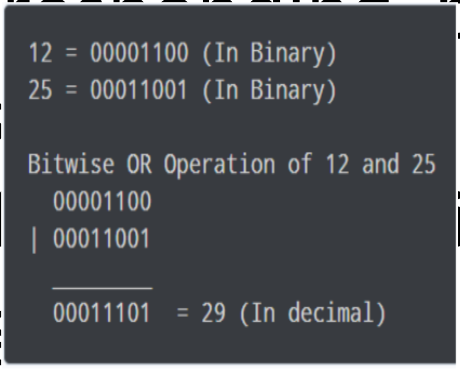
if either bit of an operand is 0.

Bitwise OR Operator (|)

- The output of bitwise OR is 1 if atle cor bit of twc nds
- The t of is 0 if cor ding



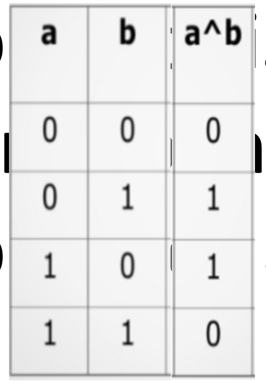
a	b	a|b
0	0	0
0	1	1
1	0	1
1	1	1



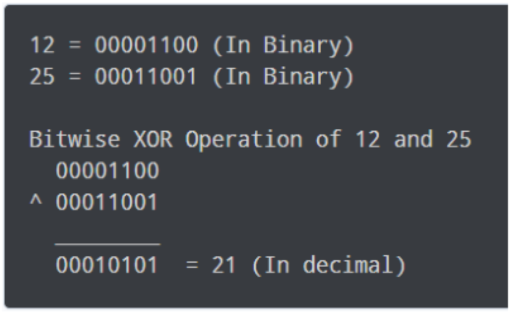
```
12 = 00001100 (In Binary)
25 = 00011001 (In Binary)

Bitwise OR Operation of 12 and 25
  00001100
| 00011001
  ________
  00011101  = 29 (In decimal)
```

Bitwise XOR Operator(^)

- The result of bitwise XOR op s 1 co din op are



a	b	a^b
0	0	0
0	1	1
1	0	1
1	1	0

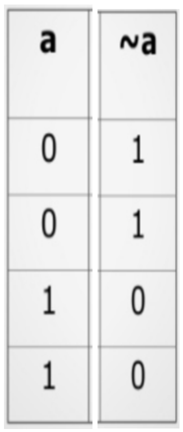


```
12 = 00001100 (In Binary)
25 = 00011001 (In Binary)

Bitwise XOR Operation of 12 and 25
  00001100
^ 00011001
  ________
  00010101  = 21 (In decimal)
```

Bitwise Compliment(~)

If the operand bit is 1 the result of bitwise compliment is 0.

If the operand bit is 0 the result of bitwise compliment is 1.

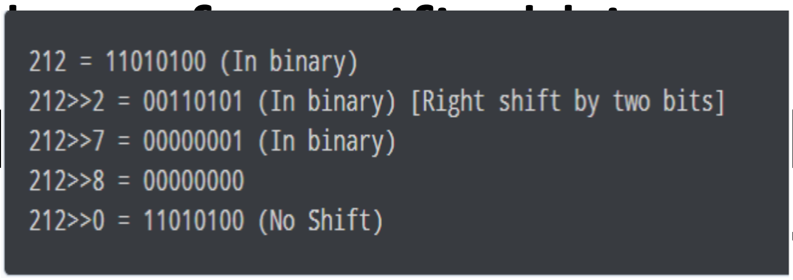


a	~a
0	1
0	1
1	0
1	0

Variable	**Decimal Value**				
a	11	1011	1011	1011	1011
b	7	0111	0111	0111	
		a&b	**a|b**	**a^b**	**~a**
Output					

Right Shift Operator(>>)

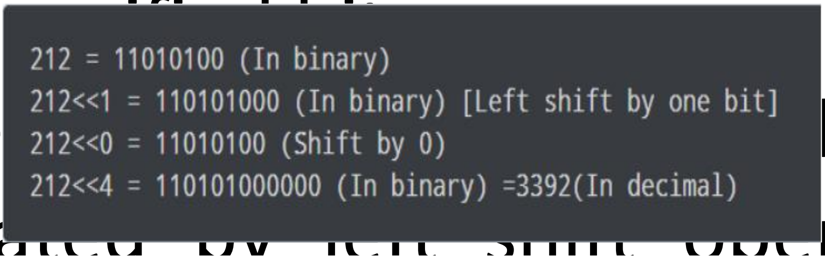
- Right Shift Operator shifts all bits towards right by certain num
- The been



```
212 = 11010100 (In binary)
212>>2 = 00110101 (In binary) [Right shift by two bits]
212>>7 = 00000001 (In binary)
212>>8 = 00000000
212>>0 = 11010100 (No Shift)
```

Left Shift Operator(<<)

- Left Shift Operator shifts all bits towards left by certain number of s
- The been vac rator



```
212 = 11010100 (In binary)
212<<1 = 110101000 (In binary) [Left shift by one bit]
212<<0 = 11010100 (Shift by 0)
212<<4 = 110101000000 (In binary) =3392(In decimal)
```

INCREMENT AND DECREMENT OPERATORS

C programming has two operators increment ++ and decrement -- to increase or decrease the value of a variable by 1.

Increment Operator (++) : increases the value of a variable by 1

Decrement Operator (--) : decreases the value of a variable by 1

These two operators are unary operators, meaning they only operate on a single operand.

Increment Operators: The increment operator is used to increment the value of a variable in an expression.

Increment operators are of two types:

Pre-increment operator: Operator ++ is placed before the variable.

A pre-increment operator is used to increment the value of a variable before using it in a expression.

In the Pre-Increment, value is first incremented and then used inside the expression.

```
Syntax:    ++var;
           ++x;          //++x is same as x=x+1
```

Example:

```
  int a=5;
     ++a;
 int x = 12, y=1;
    y = ++x;  // Now y=13 and x=13
```

Post-increment operator: Operator ++ is placed after the variable.

A post-increment operator is used to increment the value of variable after executing expression completely in which post increment is used.

In the Post-Increment, value is first used in a expression and then incremented.

```
Syntax: var++;
           x++;
```

Example:

```
int a=5;
   a++;
int x = 12, y = 1;
   y = x++;  // Now y=12 and x=13
a = 2,b = 4,c = 3
  x = ++a+ b++ - ++c;
  Value of X =3
```

Decrement Operators: The decrement operator is used to decrement the value of a variable in an expression.

Decrement operators are of two types:

Pre-Decrement operator: Operator -- is placed before the variable.

A pre-decrement operator is used to decrement the value of a variable before using it in a expression.

In Pre-Decrement, value is first decremented and then used inside the expression.

Syntax: --var;

Eg: --x;

Example:

```
  int a=5;
        --a;

 int x = 12, y = 1;
        y = --x;  // Now y=11 and x=11
```

Post-decrement operator: Operator -- is placed after the variable.

A post-decrement operator is used to decrement the value of variable after executing expression completely in which post decrement is used.

In the Post-decrement , value is first used in a expression and then decrement .

Syntax: var--;

Eg: x--;

Example:

```
  int a=5;
        a--;

 int x = 12, y = 1;
        y = x--;  // Now y=12 and x=11
```

```
 a = 2,b = 4,c = 5
x = --a+ b++ - c--;
```

Value of X =0

Conditional Operator (?:)

The conditional statements are the decision-making statements that depend upon the output of the expression.

Conditional operator works on three operands, so it is also known as the ternary operator.

It starts with a condition, hence it is called a conditional operator.

Conditional operators return one value if condition is true and returns another value if condition is false.

Syntax : expression1 ? expression2 : expression3;

Example : max = (num1 > num2) ? num1 : num2;

Example:

```
#include <stdio.h>

void main()

{

int x=1, y ;

y = ( x ==1 ? 2 : 0 ) ;

printf("Value of y is %d", y);

}
```

OUTPUT:

```
Value of y is 2
```

Write a C program to find the maximum in the given two numbers using the conditional operator.

```
#include<stdio.h>

int main()
{
int num1, num2, max;
printf("Enter two numbers: ");
scanf("%d%d", &num1, &num2);
max = (num1 > num2) ? num1 : num2;
printf("Maximum of %d and %d is %d", num1, num2, max);
}
```

OUTPUT:

```
Enter two numbers: 12 10
Maximum of 12 and 10 is 12
```

Special Operators

1.sizeof() operator:

This operator is used to compute the size of its operand.

Operand can be a variable, constant, any datatype, expression or array.

When sizeof() is used with the variables, it returns the size of a variable.

When sizeof() is used with the data types, it simply returns the amount of memory allocated to that data type.

```
Syntax : sizeof(a);
         sizeof(int);
```

```
#include <stdio.h>
int main()
{
int a = 16;
printf("Size of variable a : %d\n",sizeof(a));
printf("Size of int data type : %d\n",sizeof(int));
printf("Size of char data type : %d\n",sizeof(char));
printf("Size of float data type : %d\n",sizeof(float));
printf("Size of double data type : %d\n",sizeof(double));
}
```

OUTPUT :

```
Size of variable a : 2
Size of int data type : 2
Size of char data type : 1
Size of float data type : 4
```

Size of double data type : 8

2. Comma Operator (,)

- Comma operators are used to link related expressions together. Example: int a, c = 5, d;

3. Address of operator or reference operator (&):

- It is used to return the address of the variable.

  Example : &x

4. Pointer or dereference operator or indirection operator (*).

- It is used to return the value pointed by the pointer variable.

  Example : *x

Unary, Binary and Ternary Operators

- Unary operators have one operand to operate up on

  & Address operator ~ Bitwise complement

  ++ Increment ¯¯ decrement , etc

- Binary operators require two operands to operate up on.

  + Addition - Subtraction * Multiplication

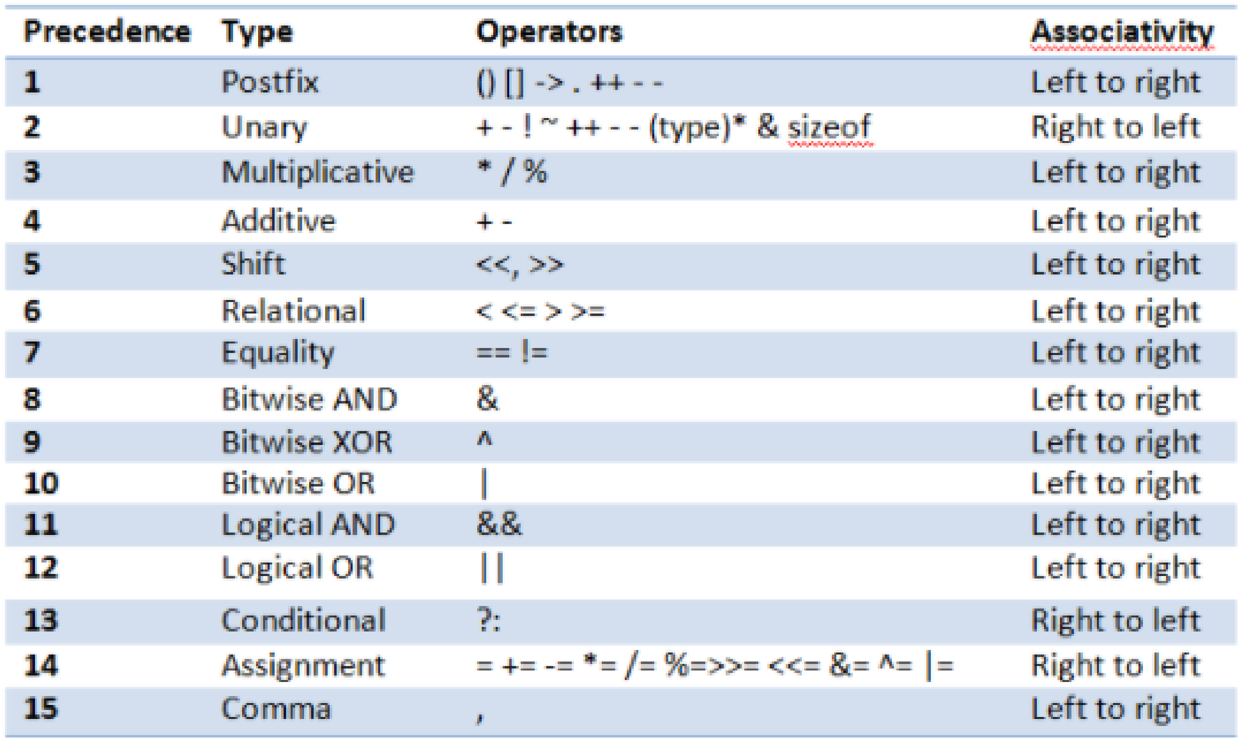
  / Division % remainder (modulo division) , etc

- Ternary operators require three operands to operate up on.

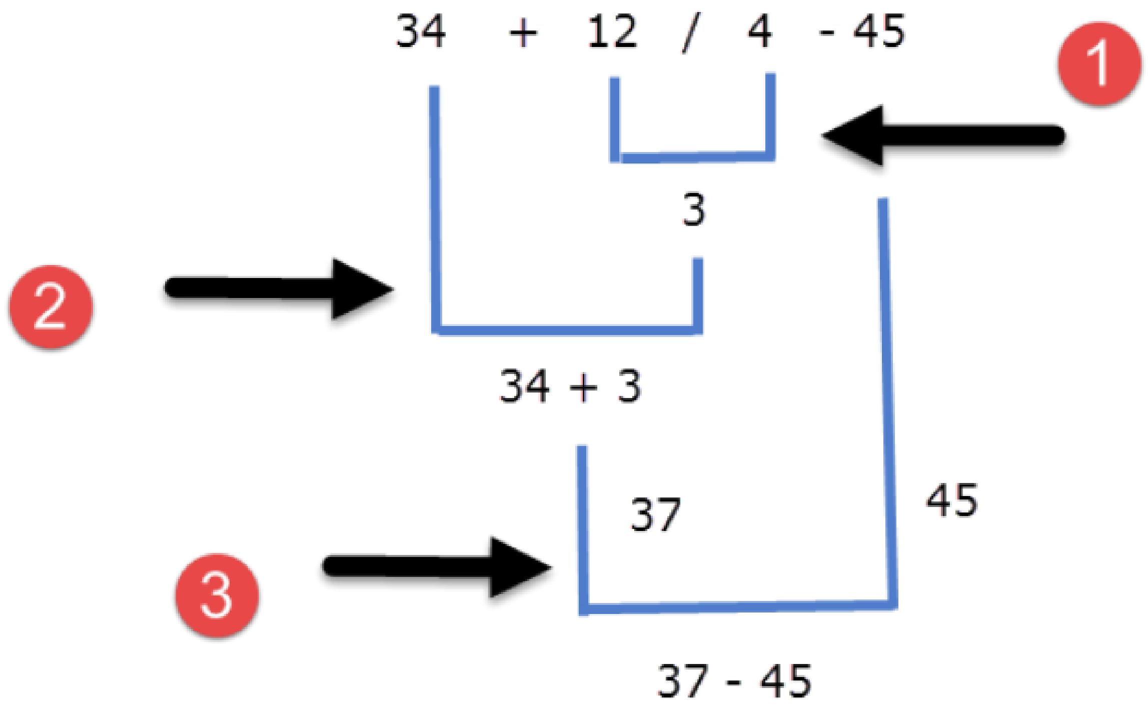
  ?: Conditional operator

Operator Precedence

- Operator precedence come into picture when in an expression we need to decide which operator will be evaluated first.
- Operator with highest priority will be evaluated first.



Precedence	Type	Operators	Associativity
1	Postfix	() [] -> . ++ - -	Left to right
2	Unary	+ - ! ~ ++ - - (type)* & sizeof	Right to left
3	Multiplicative	* / %	Left to right
4	Additive	+ -	Left to right
5	Shift	<<, >>	Left to right
6	Relational	< <= > >=	Left to right
7	Equality	== !=	Left to right
8	Bitwise AND	&	Left to right
9	Bitwise XOR	^	Left to right
10	Bitwise OR	|	Left to right
11	Logical AND	&&	Left to right
12	Logical OR	| |	Left to right
13	Conditional	?:	Right to left
14	Assignment	= += -= *= /= %=>>= <<= &= ^= |=	Right to left
15	Comma	,	Left to right



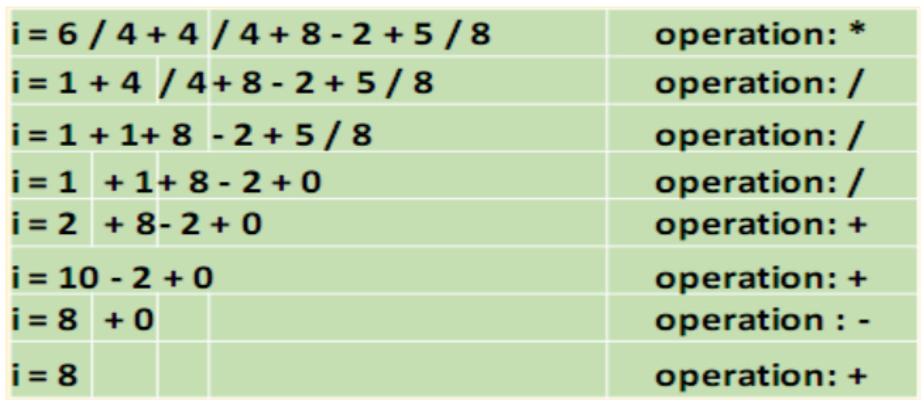
Ans : -8

- Associativity of operators come into picture when precedence of operators are same and we need to decide which operator to evaluate first.
- Examples:

  1) x=3 * 4 % 5 / 2 2) x=3 * (4 % 5) / 2 3) a=3 * 4 + 5 * 6

Evaluate the following expression?

i = 2 * 3 / 4 + 4 / 4 + 8 - 2 + 5 / 8



i = 6 / 4 + 4 / 4 + 8 - 2 + 5 / 8	**operation: ***
i = 1 + 4 / 4+ 8 - 2 + 5 / 8	**operation: /**
i = 1 + 1+ 8 - 2 + 5 / 8	**operation: /**
i = 1 + 1+ 8 - 2 + 0	**operation: /**
i = 2 + 8- 2 + 0	**operation: +**
i = 10 - 2 + 0	**operation: +**
i = 8 + 0	**operation : -**
i = 8	**operation: +**

<u>Typecasting in C</u>

Typecasting is converting one data type into another one.

It is also called as data conversion or type conversion.

Two types of Typecasting:

Implicit Type Conversion

Explicit Type Conversion

1.Implicit Type Conversion:

Compiler automatically converts one data type into another type. RULES

An arithmetic operation between an integer and integer always yields an integer result.

An operation between a real (float) and real operation is performed, the result is real.

An operation between an integer and real always yields a real result.

In this operation the integer is first promoted to a real and then the operation is performed. Hence the result is real.

2. Explicit Type Conversion:

When data of one type is converted explicitly to another type.

Syntax : (type-name) expression

Eg: b= (float) (x+y/2);

Cosider the code:

```
#include <stdio.h>
void main() {
int num1=5,count=2;
float avg;
 avg=num1/count;
printf("Average is %f",avg);
}
```

Write a C program to perform all arithmetic operations

```
#include <stdio.h>
void main()
{
  int num1, num2;
   int sum, sub, mult, mod,division;
   printf("Enter any two numbers: ");
   scanf("%d%d", &num1, &num2);
  // Perform all arithmetic operations
   sum = num1 + num2;
   sub = num1 - num2;
    mult = num1 * num2;
   division = num1 / num2;
    mod = num1 % num2;
  //Print result of all arithmetic operations
   printf("SUM = %d\n", sum);
    printf("DIFFERENCE = %d\n", sub);
   printf("PRODUCT = %d\n", mult);
    printf("QUOTIENT = %d\n",division);
   printf("Remainder = %d", mod);
}
#include<stdio.h>
void main()
{
      int a,b;
      printf("Enter the value of a and b");
      scanf("%d%d",&a,&b);
      // Uniary Operator
      printf("Unary Minus=%d\n",-a);
      // Binary Operator
      printf("Sum =%d\n",a+b);
      printf("Difference =%d\n”,a-b);
      printf("Mul =%d\n",a*b);
      printf("Division =%d\n",a/b);
      printf("Modulo =%d\n",a%b);
}
```

Write a C program to separate integer and decimal part

```
#include<stdio.h>
void main()
{
        float num,a;
        int i;
        printf("\n enter a float number :");
        scanf("%f",&num);
        i=num;
        a= num-i;
        printf("\n number=%f",num);
        printf("\n integer part=%i",i);
        printf("\n decimal part=%f",a);
}
```

Output:

```
 enter a float number :234.76
number=234.76
integer part=234
decimal part=0.76
```