Notes

Monday, September 19, 2022 1:52 PM

Let us see a program

name

Raju

output

Hi Raju

```
#include<stdio.h>
#include<string.h>
main()
{
     char name[5];
     strcpy(name,"Raju");
     printf(" Hi %s", name);
}
```

name

char name[5];

variable name

```
#include<stdio.h>
#include<string.h>
main()
{
     char name[5]="Raju";
     printf(" Hi %s", name);
}
```

I want the output as welcome to Hyderabad

```
#include<stdio.h>
#include<string.h>
main()
{
      char name[10];
      strcpy(name,"Hyderabad");
      printf("Welcome to %s",name);
}
```

```
#include<stdio.h>
#include<string.h>
main()
{
      char name[10]="Hyderabad";
      printf("Welcome to %s",name);
}
```

Scanf Function

For example

```
#include<stdio.h>
main()
{
   int x=3,y=4,z;
   z=x+y;
   printf("%d",z);
}
```

If I run this program 10 times
Everytime x=3, y=4 and z value is 7 only

If I run this program 100 times
z value is 7 only

If I want to give x, y different values ,
during the time of running a program
how it is possible?

If I gave x value as 10
y value as 20
I should get z value as 30

If I gave x value as 40
y value as 60
I should get z value as 100

If I gave x value as 80
y value as 40
I should get z value as 120

How is it possible?

It is possible through scanf function

<u>scanf()</u>

1. **scanf is a built in function**
2. **It is a input related function**
3. **scanf is used to read data at runtime.**
4. **using scanf we can read any type of data like int, float, char ,string.**

stdio.h

```
scanf(….)
{
   //code
}
```

along with scanf we have to use one operator that operator is

&

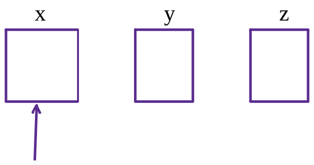
& means addressof operator

For example

Let us see a program

```
#include<stdio.h>
main()
{
   int x,y,z;
   scanf("%d%d",&x,&y);
   z=x+y;
   printf("%d",z);
}
```

we are giving int value
it will allocates 4 bytes of memory



Every location
has some address

address means
a place very
the value is stored

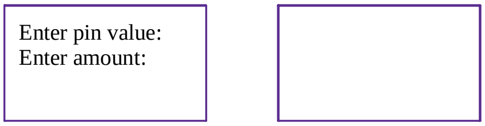
see the difference between the previous program without scanf and this program with scanf

```
#include<stdio.h>
main()
{
  int x=3,y=4,z;
  z=x+y;
  printf("%d",z);
}
```

```
#include<stdio.h>
main()
{
   int x,y,z;
   scanf("%d%d",&x,&y);
   z=x+y;
   printf("%d",z);
}
```

One problem is here, we are the developer we know why the screen is waiting,
But any person saw our program he don’t know why our screen is waiting

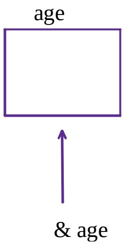
For example if you go to ATM



To tell why the screen is waiting we have to write one statement above scanf

```
#include<stdio.h>
main()
{
    int x,y,z;
    printf("Enter x and y value:");
    scanf("%d%d",&x,&y);
    z=x+y;
    printf("%d",z);
}
```

Write a program to take the age from user and display the age on the screen

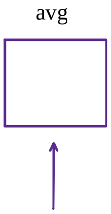


```
#include<stdio.h>
main()
{
    int age;
    scanf("%d",&age);
    printf("Ur age is:%d",age);
}
```

```
#include<stdio.h>
main()
{
    int age;
    printf("Enter your age:");
    scanf("%d",&age);
    printf("Ur age is:%d",age);
}
```

Write a program to take the average from user and display the average on the screen

```
#include<stdio.h>
main()
{
    float avg;
    scanf("%f",&avg);
    printf("Ur avg is:%f",avg);
}
```



As it is a float value it will allocates 4 bytes of memory.

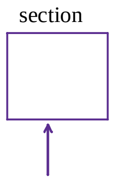
& address of operator
To place the value in the location of avg

```
#include<stdio.h>
main()
{
    float avg;
```

```
   printf("Enter avg:");
   scanf("%f",&avg);
   printf("Ur avg is:%f",avg);
}
```

Write a program to take the section from user and display the section on the screen

```
#include<stdio.h>
main()
{
    char section;
    scanf("%c",&section);
    printf("Ur section is:%c",section);
}
```



As it is a char
value it will
allocates
1 byte of
memory.

& address
of operator
To place the value
in the location of section

```
#include<stdio.h>
main()
{
    char section;
    printf("Enter your section:");
    scanf("%c",&section);
    printf("Ur section is:%c",section);
}
```

Now one twist is there let us see

Write a program to take the name from user and display the name on the screen

Here name means group of characters
group of characters means string
there is no data type for string

we use char data type and mention the size

In c language string must be enclosed in double quotes

```
#include<stdio.h>
main()
{
     char name[20];
     scanf("%s",name);
     printf("Ur Name is:%s",name);
}
```

```
#include<stdio.h>
main()
{
     char name[20];
     printf("Enter your name:");
     scanf("%s",name);
     printf("Ur Name is:%s",name);
}
```

Write a program to take the city from user and display the city name on the screen

```
#include<stdio.h>
main()
{
    char city[20];
    scanf("%s",city);
    printf("Ur city is:%s",city);
}
```

```
#include<stdio.h>
main()
{
    char city[20];
    printf("Enter your city:");
    scanf("%s",city);
    printf("Ur city is:%s",city);
}
```

In c languge

int/int is always int only

4/2=4
6/3=2
5/2=2.5--->float
9/2=4.5---->float

Let us see one example

```
#include<stdio.h>
void main()
{
   printf("%d",4/2);
}
```

```
#include<stdio.h>
void main()
{
   printf("%d",6/2);
}
```

```
#include<stdio.h>
void main()
{
   printf("%f",4/2);
}
```

I will get 0.0000 as a result

```
#include<stdio.h>
main()
{
int x=5,y=2;
printf("%d",x/y);
}
```

5/2=2.5

int--->2.~~5~~ ⇒ 2

I should tell the compiler give the result in float

ఇప్పుడు మనకు అనిపిస్తుంది, ఏంటి, format specifier దగ్గర float అంటే %f అని రాయండి సార్ అని అనిపిస్తుంది, అలా రాసినా నాకు result zero యే వస్తుంది

```
#include<stdio.h>
main()
{
int x=5,y=2;
printf("%f",x/y);
}
```

ఇప్పుడు ఏం చేయాలి?

```
#include<stdio.h>
main()
{
int x=5,y=2;
printf("%f",(float)x/y);
}
```

for example if I wrote like this, what is the output I will get

```
#include<stdio.h>
main()
{
  int x=5;
  printf("%d", x);
}
```

I will get 5 as a output

but I will give x value as 5 and I want output as 5.0

```
#include<stdio.h>
main()
{
  int x=5;
  printf("%f", x);
}
```

you will tell in the format specifer write %f
if I write %f in the format specifier we will get 0

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

5 ⟶ 5.0

```
}
```

5 → 5.0
int   float

```
#include<stdio.h>
main()
{
   float x=5.7;
   printf("%f", x);
}
```

I will get 5.7 as a output

but I will gave x value as 5.7 and I want output as 5

```
#include<stdio.h>
main()
{
   float x=5.7;
   printf("%d", x);
}
```

I will get 0 or wrong output

```
#include<stdio.h>
main()
{
   float x=5.7;
   printf("%d", (int)x);
}
```

5.7 → 5
float   int

from this we learnt that
we can convert int value to float value
float value to int value

x=5	5.0	
int	float	(float)x
x=5.7	5	(int)x
float	int	

The conversion of one data type to another data type is called type casting

The most asked interview question

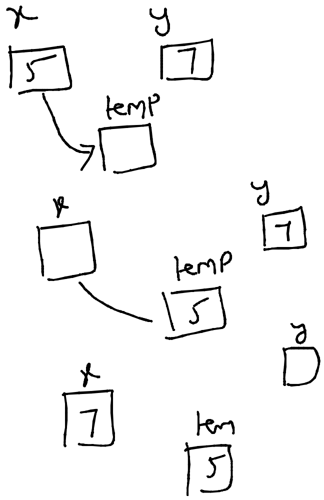
swapping of two numbers

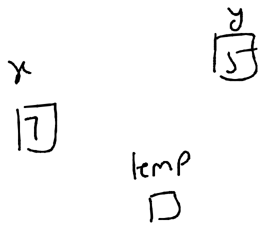
For example, ఈ క్రింద ప్రోగ్రాం చూసి output ఏమి వస్తుందో చెప్పండి

```
#include<stdio.h>
main()
{
   int x=5,y=7;
   printf("x=%d, y=%d",x,y);
 }
```

```
#include<stdio.h>
main()
{
   int x=5,y=7;
   printf("x=%d, y=%d",x,y);
 }
```

ఇప్పుడు నాకు ఈ printf statement దగ్గర x=7, y=5





```
#include<stdio.h>
main()
{
    int x=5, y=7, temp;
    temp=x;
     x=y;
    y=temp;
   printf("x=%d y=%d",x,y);
}
```

Swapping of two numbers without using temp variable

For example

x y

$\boxed{30}$ $\boxed{20}$

What output I should get

x y

$\boxed{20}$ $\boxed{30}$

There is a formula, without using temporary variable , we can use this formula also for swapping

what is the formula is

x=x+y
y=x-y
x=x-y

x y

$\boxed{30}$ $\boxed{20}$

x=x+y
x=30+20
=50

x y

$\boxed{50}$ $\boxed{20}$

y=x-y
y=50-20
=30

x y

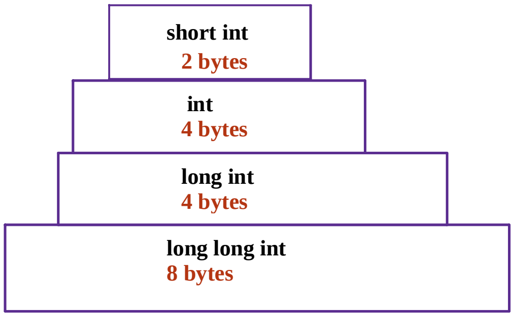
$\boxed{50}$ $\boxed{30}$

```
#include<stdio.h>
```

```
main()
{
   int x=20,y=30;
   x=x+y;
   y=x-y;
   x=x-y;
  printf("x=%d\n y=%d",x,y);
}
```

We know there are three basic data types
in c langauge
1.int
2.float
3.char

int data type is again divided into 4 types



short int
2 bytes

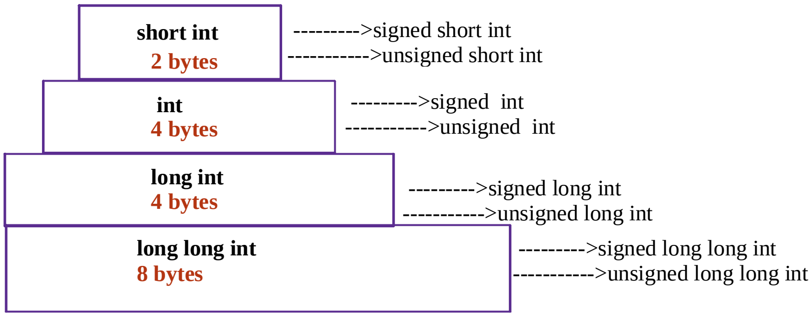
int
4 bytes

long int
4 bytes

long long int
8 bytes

Every type is again divided into two types

signed	+ve & -ve
unsigned	+ve only



short int
2 bytes
--------->signed short int
----------->unsigned short int

int
4 bytes
--------->signed int
----------->unsigned int

long int
4 bytes
--------->signed long int
----------->unsigned long int

long long int
8 bytes
--------->signed long long int
----------->unsigned long long int

9 rows four columns   default ga short int----->signed short int

Data type	**size**	**Range**	**Format Specifier**
short int	2bytes	-32768 to 32767 -2^{15} to $2^{15}-1$	%hd
unsigned short int	2bytes	0 to 65535 0 to $2^{16}-1$	%hu
int	4bytes	-2^{31} to $2^{31}-1$	%d
unsigned int	4bytes	0 to $2^{32}-1$	%u
long int	4bytes	-2^{31} to $2^{31}-1$	%ld
unsigned long int	4bytes	0 to $2^{32}-1$	%lu
long long int	8bytes	-2^{63} to $2^{63}-1$	%lld
unsigned long int	8bytes	0 to $2^{64}-1$	%llu

short int

Let us see an example for range of a short int

range -------->-32768 to 32767
format specifier----->%hd

```
#include<stdio.h>
```

```
main()
{
    short int x;
    scanf("%hd",&x);
    printf("%hd",x);
}
```

```
#include<stdio.h>
main()
{
    short int x=20000,y=30000,z;
    z=x+y;
    printf("%hd",z);
}
```

```
#include<stdio.h>
main()
{
    short int x=10000,y=10000,z;
    z=x+y;
    printf("%hd",z);
}
```

Let us see a example for int data type

range=-2,147,483,648 to 2,147,483,647

```
#include<stdio.h>
main()
{
    int x;
    scanf("%d",&x);
    printf("%d",x);
}
```

Sizeof Operator

Size of operator is used to know the size of the data type or variable.

Syntax

sizeof(<datatype>/<variable>)

If we want to know the size of the data type give the data type name
If we want to know the size of the variable give the variable name

```
#include<stdio.h>
main()
{
    printf("Size of int is:%d bytes\n",sizeof(int));
    printf("Size of float is:%d bytes\n",sizeof(float));
    printf("Size of char is:%d byte\n",sizeof(char));
}
```

Not only datatype we can also find the size of the variable also

```
#include<stdio.h>
main()
{
    int x=3;
    float y=4;
    char z='A';
    printf("Size of x is:%d bytes\n",sizeof(x));
    printf("Size of y is:%d bytes\n",sizeof(y));
    printf("Size of z is:%d byte\n",sizeof(z));
}
```

Why different data types have different ranges?

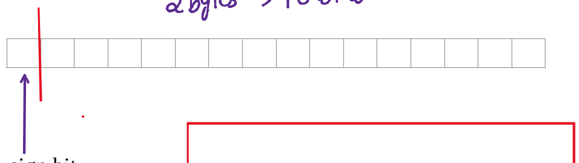
For example,

short int

Short int means by default it takes both -ve value & +ve value

It has a range of -32768 to 32767, size is 2 bytes

bit -> 0 (or) 1
1 byte -> 8 bits
2 bytes -> 16 bits



sign bit

+ve ---->0
-ve--->1

32767--->111 1111 1111 1111
32768--->1000 0000 0000 0000

Unsigned short int

unsigned short int means it takes only **+ve value**

It has a range 0 to 65535, size is 2 bytes

2bytes means 16 bits

65535--->1111111111111111
65536--->1 0000 0000 0000 0000

Float Data Type

Float data type is again divided into 3 types

1.float
2.double
3.long double

DataType	Size	Precision[No of decimal places]	Format Specifier
float	**4bytes**	**6**	**%f**
double	**8bytes**	**15**	**%lf**
long double	**16bytes**	**18**	**%Lf**

```
#include<stdio.h>
main()
{
    float a=6.5;
    printf("%f",a);
}
```

Number System

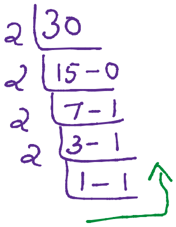
They are four numbers systems

1.Decimal Number System
2.Binary Number System
3.Octal Number system
4.Hexa Decimal Number System

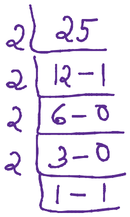
For example let us take 30
If we see in scientific calculator

programmer calculator

The binary value of 30 is 11110

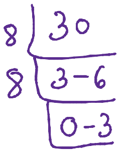


The binary value of 25 is 11001



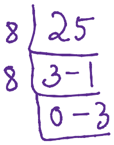
Let us convert 30 to octal value

Divide with 8



The octal value of 30 is 36

For example take another number
now convert
25 into octal



31

The octal value of 25 is 31

Now let us convert 30 to hexa decimal
format

In Hexa decimal

10-->A
11-->B
12-->C
13--->D
14-->E
15-->F

we have to divide the number with 16

16 | 30

16 | 1 – 14

So we got

1 14

1E

So the hexadecimal Format of 30 is 1E

Now let us see the hexadecimal format For 25

16 | 25

| 1 – 9

19

So the hexadecimal Format of 25 is 19

Now let us see how to convert from one format to another format in C language

In c language to convert Decimal number into octal and Hexa decimal we should use the following format specifier

octal	**%o**
Hexadecimal	**%x or %X**

For example

```
#include<stdio.h>
main()
{
   int a=30;
   printf("%o",a);
}
```

```
#include<stdio.h>
main()
{
   int a=30;
   printf("%x",a);
}
```

```
#include<stdio.h>
main()
{
   int a=30;
   printf("%X",a);
}
```

I have a Question?

Can we print like this

```
#include<stdio.h>
main()
{
    printf("You have 25% salary increase");
}
```

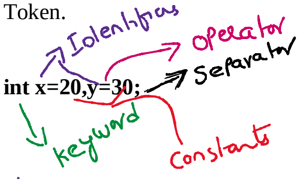
No we will get special characters in place of %
symbol

instead of it we have to write like this

```
#include<stdio.h>
main()
{
    printf("You have 25%% salary increase");
}
```

C language Tokens

Every smallest individual part of the C-Language is called as
Token.



int x=20,y=30;

C language Tokens are divided into

1.Identifier
2.keyword
3.constant
4.Operator
5.Separator
6.Comment

Identifier

Names of the variables, functions or any user defined name is called as identifier

Rules for Naming Identifiers

1.Identifiers are made up of
  a.Letters
  b.Digits
  c. Underscore(__)
  d.Dollar($)

Other than above, No other characters are not valid for naming identifier

int marks	**valid**
int total marks	**Invalid**
int totalmarks	**valid**
int total_marks	**valid**
int total@marks	**Invalid**
int total$marks	**Valid**

```
#include<stdio.h>
main()
{
   int total$marks;
}
```

2.Identifier shouldnot start with a digit.

3.Identifier can be of any length.

```
#include<stdio.h>
main()
{
   int afdfdsfdsfjdsfewfdsfjdsfdsfdsfsdfdsfjdsfdsfsdfdsfdsafdsafjsdfjdsafjdsfjdsajfdsjfdsa=100;
   printf("%d",afdfdsfdsfjdsfewfdsfjdsfdsfdsfsdfdsfjdsfdsfsdfdsfdsafdsafjsdfjdsafjdsfjdsajfdsjfdsa);
}
```

4.Identifiers are case Sensitive. Upper
Case and Lower characters will be

treated as different

```
#include<stdio.h>
main()
{
    int x=20,X=30;
    printf("%d",x);
    printf("%d",X);
}
```

5.Reserved words should not be used as identifiers

```
int float;----------->valid
int if;------------>invalid
int for;---------->Invalid
int INT  ------->valid
```

Keywords

1.Reserved Words are called as Keywords

For example : If, For, else, while….

2. Every Keyword keyword has a specific meaning and that can perform particular functionality

```
#include<stdio.h>
main()
{
    int x=5;
    if(x==5)
    {
     printf("Hai");
    }
    else
    {
     printf("Hello");
    }
}
```

Constants

Constants are values assigned to a variable.

Constants are:

Integer Constants	**Ex:73,22,34,45**
Float Constants	**Ex:34.5,56.6,77.7**
Char Constants	**Ex: 'A','C','D'**
String Constants	**Ex:'Rajesh','Kavya'**

Integer Constants

Binary Constants	**Prefixed with 0b**
Octal Constants	**Prefix with O**
Hexa Decimal Constants	**Prefix with 0x**

```
#include<stdio.h>
main()
{
   int x=111;
   printf("%d",x);
}
```

```
#include<stdio.h>
main()
{
   int x=0b111;
   printf("%d",x);
}
```

```
#include<stdio.h>
main()
{
   int x=036;
   printf("%d",x);
}
```

Operators

Operator is a symbol that is used to perform operations like arthimetic, or logical operations

Arthimetic: +,-,*,/

Ex: Adding two numbers, multiplying two numbers

Logical: >,<,==
5>6

```
#include<stdio.h>
main()
{
   int a=5,b=6;
   printf("%d", a>b);
}
```

we will get 0 as the answer false means 0, 1 means true

Separators

Separator can be used in c programming to separate variables, statements, blocks

,	**Variable Separator**
;	**Statement Separator**
{}	**Block Separator**
[]	**Size/Subscript Separator**
()	**Expression Separator**

```
#include<stdio.h>
main()
{
    int a=5,b=5,c=0,d;
    d=(a+b)+(c+a);
    printf("%d",d);
}
```

```
#include<stdio.h>
main()
{
   char name[5]="ravi";
   printf("%s",name);
}
```

6.Comment

To improve redability to the other users we write comment lines
In C programming for single we use //
In C programming for multi line comments we use /*   */

```
// This programming is used to print a string
#include<stdio.h>
```

```
main()
{
   char name[5]="ravi";
   printf("%s",name);
}
```

```
/*This programming is used to print a string*/
#include<stdio.h>
main()
{
   char name[5]="ravi";
   printf("%s",name);
}
```

Operators

Operator:Operator is a symbol that is used to perform operations like artithemtic or logical operations

Operand: The variable that is participating in operation is called as "Operand"

Expression: Expression is a combination of Operators and Operand

Ex:

x+y

x,y ---> Operands
+------>Operator
x+y----> Expression

We can divide Operands into two ways based

1. Number of Operands
2. Based on Purpose

Based on Number Operands, we divide operators into 3 types

1.Unary Operator

If one operand is participated in operation then it is called "Unary Operator"

Ex:

x++------->increment Operator
x-- -------->Decrement Operator
-x----------> Unary Operator

```
#include<stdio.h>
main()
{
   int x=10;
   x++;
   printf("%d",x);
}
```

```
#include<stdio.h>
main()
{
   int x=10;
```

```
  x--;
  printf("%d",x);
}
```

Binary Operator:

If two Operands participates in Operation then it is called "Binary operator.

Ex:

x+y--->addition
x*y------>Multiplication

Ternary Operator

If three Operands Participate in the Operation then it is called as "Ternary Operator.

Ex: a>b? a:c===>

Based on the purpose we can categorize Operators into five types

Arthimetic Operators	**+ - *** **Addition Substraction Multiplication** **/ %** **Division Modulo Division**
Relational or Comparision Operators	**>, <, >=, <=, == !=** **Equals Not Equal**
Logical Operators	**&& || |** **AND OR NOT**
Assignment Operator	**=**
Bitwise Operators	**& | (pipe) ^(Caret/cap)** **b/w AND B/W OR B/W XOR** **<< >> ~(Tilde)** **Left Shift Right Shift Complement**

Arthimetic Operators

Arithmetic operators are used to perform arthimetic operations

Following arthimetic operations are provided by c-language

+	**Addition**	**x+y=>10+2=12**
-	**Substraction**	**x-y=>10-2=8**
*	**Multiplication**	**x*y=>10*2=20**
/	**Division**	**x/y=>10/2=5 [Quotient]**
%	**Modulo Division**	**x%y=>10%2=0 [Remainder]**

```
#include<stdio.h>
main()
{
  int a=10, b=5;
  printf("%d",a/b);
}
```

output

```
2
```

```
#include<stdio.h>
```

```
main()
{
   int a=10, b=5;
   printf("%d",a%b);
}
```

/	**division Operator gives quotient output**
%	**modulo operator gives remainder as output**

```
#include<stdio.h>
main()
{
   int a=5, b=2;
   printf("%d",a/b);
}
```

5/2=2.5
5%2=1

5/2	**2**	**int/int=int**
5%2	**1**	**int%int=int**
5+2	**7**	**int+int=int**
5-2	**3**	**int-int=int**
5*2	**10**	**int*int=int**

5.0/2	**2.5**	**float/int=float**
5.0%2	**Error**	**float%int=Error**
5.0+2	**7.0**	**float+int=float**
5.0-2	**3.0**	**float-int=float**
5.0*2	**10.0**	**float*int=float**

% Modulus operator can be used on 2 integer Operands only

```
#include<stdio.h>
main()
{
   int a=5;
   float b=2.0;
   printf("%f",a%b);
}
```

5%2	1
5.0%2	Error
5%2.0	Error
5.0%2.0	Error

5/2	2
5.0/2	2.5
5/2.0	2.5
5.0/2.0	2.5

Program to demonstrate Arthimetic operators

```
#include<stdio.h>
main()
{
  int x=5, y=2;
  printf("%d",x+y)
  printf("%d",x-y)
  printf("%d",x*y)
  printf("%d",x/y)
  printf("%d",x%y)
}
```

```
#include<stdio.h>
main()
{
  int x=5, y=2;
  printf("sum=%d\n",x+y);
  printf("diff=%d\n",x-y);
  printf("product=%d\n",x*y);
  printf("Quotient=%d\n",x/y);
  printf("remainder=%d\n
```

Control Structures

Sequential Structures

```
#include<stdio.h>
main()
{
   printf("Hi\n");
   printf("Welcome\n");
   printf("bye\n");
}
```

Control Structures

```
#include<stdio.h>
main()
{
   int x=5;
   if(x>5)
     printf("Hi\n");
   if(x<5)
    printf("Welcome\n");
  if(x==5)
    printf("bye\n");
}
```

Control Structures:

1. Control Structures are used to control the flow of execution of statements
2. It changes sequential execution
3. Normally, C-Program gets executed sequentially. To change the sequential execution, to transfer the control to our desired location we use control structures

C-Language provides the following control structures

Conditional	if
	if else
	if else if
	nested if

Multi way conditional	switch

Looping/Iterative	while
	do while
	for

Jumping	goto
	break
	continue
	return

We must write in small letters only

As of now we are going to discuss now

Conditional Control Structures

Conditional Control structures get executed based on the condtion.

C-Language provides the following conditional Control structures

1.if
2.if else
3.if else if
4.nested if

we must write if in small letters only

in front of if we have to write condition in parenthesis

if

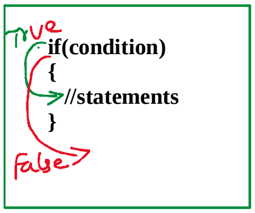
Syntax

```
if(condition)
{
   //statements
}
```

it is used to perform a task based on condition.
First it checks the condition, if the condition is True , statements get excuted.
if condition is FALSE statements will not be executed

The statements in "if" block get executed when the condition is true

```
#include<stdio.h>
main()
{
   int x=6;
   if(x>5)
   printf("The value is greater than 5");
}
```



```
if(condition)
{
//statements
}
```

```
#include<stdio.h>
main()
{
    int x=6;
    if(x>5)
    printf("The value is greater than 5");
    if(x<5)
    printf("The value is less than 5");
}
```

```
#include<stdio.h>
main()
{
    int x=4;
    if(x>5)
    printf("The value is greater than 5");
    if(x<5)
    printf("The value is less than 5");
}
```

Ko

```
#include<stdio.h>
main()
{
    int marks=40;
    if(marks>=40)
    printf("You have passed");
    if(marks<40)
    printf("You have failed");
}
```

```
#include<stdio.h>
main()
```

```
{
    int marks=40;
    if(marks>=40)
    printf("You have passed");
    printf("Congratulations");
    if(marks<40)
    printf("You have failed");
    printf("You have to write Supplementary exam");
}
```

```
#include<stdio.h>
main()
{
    int marks=40;
    if(marks>=40)
    {
    printf("You have passed");
    printf("Congratulations");
    }
    if(marks<40)
    {
    printf("You have failed");
    printf("You have to write Supplementary exam");
    }
}
```

If one statement is there in "if" block
we have no need to write curly
braces{}

The problem with the above program
is two time condition is checked . so
the execution time is wasted.

So the alternative is if else

Syntax

```
if(condition)
{
    //statements
}
else
{
   //statements
}
```

```
#include<stdio.h>
main()
{
    int marks=40;
    if(marks>=40)
    printf("You have passed");
    else
    printf("You have failed");
}
```

```
#include<stdio.h>
main()
{
    int marks=40;
    if(marks>=40)
    {
    printf("You have passed");
    printf("Congratulations");
    }
    else
    {
    printf("You have failed");
    printf("You have to write Supplementary exam");
    }
}
```

```
#include<stdio.h>
main()
{
    int marks=40;
    if(marks!=40)
    printf("You have Failed");
    else
    printf("You have Passed");
}
```

To read the marks at the run time

```
#include<stdio.h>
main()
{
   int marks;
   scanf("%d",&marks);
   if(marks!=40)
   {
       printf("You have Failed");
   }
   else
   {
       printf("You have passed");
    }
}
```

```
#include<stdio.h>
main()
{
   int marks;
   printf("Enter marks:");
   scanf("%d",&marks);
   if(marks!=40)
   {
       printf("You have Failed");
   }
   else
   {
       printf("You have passed");
    }
}
```

```
#include<stdio.h>
main()
{
    int marks;
    printf("Enter your marks:");
    scanf("%d",&marks);
    if(marks>=35)
    {
        printf("PASS");
     }
     if(marks<35)
     {
         printf("FAIL");
      }
}
```

The problem with the above program is two time condition is checked . so the execution time is wasted.

so the above program we can write like this also

```
#include<stdio.h>
main()
{
    int marks;
    printf("Enter your marks:");
    scanf("%d",&marks);
    if(marks>=35)
    {
        printf("PASS");
     }
     else
    {
         printf("FAIL");
      }
}
```

Write a program to check the given number is even or odd

even ===>2,4,6,8,10
odd===>3,9,15,21

To get remainder value which value we have to use modulus operator

```
#include<stdio.h>
main()
{
    int n;
    printf("Enter a number:");
    scanf("%d",&n);
    if(n%2!=0)
    {
    printf("Odd Number");
    }
    else
   {
    printf("Even number");
    }
}
```

Write a program to check whether the given number is divisible by 7 or not

```
#include<stdio.h>
main()
{
    int n;
    printf("Enter a number:");
    scanf("%d",&n);
    if(n%7==0)
    {
    printf("Divisible by 7");
    }
    else
   {
    printf("Not Divisible by 7");
    }
}
```

write a program to check whether a person is eligible for vote or not

```
#include<stdio.h>
main()
{
    int age;
    printf("Enter your age:");
    scanf("%d",&age);
    if(age>=18)
    {
    printf("You are eligible for vote");
    }
    else
   {
    printf("Not eligible for Vote");
    }
}
```

Logical Operators

Logical operators are used on multiple conditions

In c language there are 3 logical operators

&&	Logical AND
||	Logical OR
!	Logical NOT

&&	All conditions should be satisfied
||	At least one condition must be satisfied

Truth Table

AND Truth Table

Conditon-1	Condition-2	Condition-1 &&Condition-2
T	T	T
T	F	F
F	T	F
F	F	F

AND means if all values are true then the result is true

OR Truth Table

Conditon-1	Condition-2	Condition-1 ||Condition-2
T	T	T
T	F	T
F	T	T
F	F	F

OR means if atleast one value is true then the result is true otherwise False

For example to check whether the person is eligible for army exam or not

age==> between 18 and 25

```
Enter age:23
Eligible
```

```
Enter age:30
Not Eligible
```

```
Enter age:16
Not Eligible
```

```
if(age>=18 &&  age<=25)
```

```
#include<stdio.h>
main()
{
  int age;
  printf("Enter your age:");
  scanf("%d",&age);
  if(age>=18 &&  age<=25)
  {
    printf("You are eligible for Army Exam");
   }
   else
   {
     printf("You are not eligible for Army Exam");
   }
}
```

In case of && [AND],
If first condition is FALSE, it never checks remaining conditions

|| or operator

In case of OR atleast one condition is true the result is true

Write a C program to check whether the given number is divisible by 3 or 7

```
#include<stdio.h>
main()
{
   int n;
   printf("Enter n value:");
   scanf("%d",&n);
   if(n%3==0 || n%7==0)
   {
      printf("The number is divisible by 3 or 7");
    }
    else
    {
       printf("The number is not divisible by 3 or 7");
    }
}
```

In case of || [OR],
If first condition is TRUE, it never checks remaining conditions

In a office there are managers, analyst, clerk, salesman . Office Management decides to increase the salary to only Managers and Clerks Write a c program to print the statement as"Your salary is increased" if the designation is either clerk or Manager.

```
manager==>m
clerk==>c
analyst==>a
salesman==>s

if (job=='m'   job=='c')
```

```
#include<stdio.h>
main()
{
   char designation;
   printf("Enter your designation:");
   scanf("%c",&designation);
   if(designation=='m' || designation=='c')
   {
       printf("Your salary is increased");
    }
    else
    {
         printf("Your salary is not increased");
     }
}
```

A Office Management decides to increase the salary to only Managers whose salary is above 10000 Write a c program to print the statement as"Your salary is increased" if the designation is manager and salary is 10000

```
#include<stdio.h>
main()
{
   char designation;
   int salary;
   printf("Enter your designation:");
   scanf("%c",&designation);
   printf("Enter your salary:");
   scanf("%d",&salary);
   if(designation=='m' && salary>=10000)
   {
      printf("Your salary is increased");
   }
   else
   {
        printf("Your salary is not increased");
    }
}
```

! Operator

Logical Not

Truth Table

Condition	! (Condition)
T	!(T)--->F
F	!(F)---->T

```
#include<stdio.h>
main()
{
   int x=5;
   printf("%d", x>3);
}
```

```
#include<stdio.h>
main()
{
   int x=5;
   printf("%d", !(x>3));
}
```

Assignment Operators

C language Provides the following Assignment Operators

=	Simple Assignment
+=	Addition Assignment
-=	Substraction Assignment
*=	Multiplication Assignment
/=	Division Assignment
%=	Modulus Assignment

Example:

int x=5;

x+=20; ===>x=x+20==>5+20

int x=7;

x-=4==>x=x-4==>7-4==>3

$\cancel{5}$
25

$\cancel{7}$
3

int x=7;

x*=4==>x=x*4==>7*4==>28

$\cancel{7}$
28

int x=6;

x/=3==>x=x/3==>6/3==>2

x/=3==>x=x/3==>6/3==>2

~~6~~
2

int x=6;

x%=3==>x=x%3==>6%3==>0

~~6~~
0

When the variable is same only we can use the Assignment Operator

x=y+z //No

x=x+10;===>x+=10
x=x+y===>x+=y

```
#include<stdio.h>
main()
{
   int x=20;
   x+=10;
   printf("%d",x);
}
```

```
#include<stdio.h>
main()
{
   int x=20;
   x/=10;
   printf("%d",x);
}
```

Unary Minus Operator

1.Symbol: -
2. It is a operator. Only one Operand participates in the operation
3. It is used to convert +ve to -ve or -ve to +ve
4. Here we multiply operand with -.

Example:

Program to demonstrate unary minus operator

```
#include<stdio.h>
main()
{
   int x=15;
   x=-x;
   printf("%d\n",x);
   x=-x;
   printf("%d\n",x);
}
```

Ternary Operator/Conditional Operator:

Symbol: ?:

It is used to execute the expression based on the condition.

Syntax:
<condition>? <true_expression>:<false_expression>;

Write a program to find Biggest of two numbers

```
#include<stdio.h>
main()
{
   int x,y;
   printf("Enter  two numbers:");
   scanf("%d%d",&x,&y);
   if(x>y)
   {
     printf(" X is greater than Y");
   }
   else
   {
     printf("Y is greater than X");
   }
}
```

```
#include<stdio.h>
main()
{
   int x,y;
   printf("Enter  two numbers:");
   scanf("%d%d",&x,&y);
   if(x>y)
   {
     printf(" X is greater than Y");
   }
   else
   {
     printf("Y is greater than X");
   }
}
```

x>y?printf("x is greater than y"):printf("y is greater than x")

```
#include<stdio.h>
main()
{
   int x,y;
   printf("Enter  two numbers:");
   scanf("%d%d",&x,&y);
   x>y?printf("x is greater than y"):printf("y is greater than x");
}
```

```
#include<stdio.h>
main()
{
   int n;
   printf("Enter a number:");
   scanf("%d",&n);
   if(n%2==0)
   {
     printf(" The number is even number");
   }
   else
   {
     printf("The number is odd number");
   }
}
}
```

```
#include<stdio.h>
main()
{
   int n;
   printf("Enter a number:");
   scanf("%d",&n);
   n%2==0?printf("The number is even number"):printf("The number is odd number");
}
```

<u>Increment Operator</u>

1.Symbol: ++
2.It is a unary Operator. one operand participates in Operation
3.Used to increase by 1 value of the variable
4.It can be used in 2 ways
a. Post Increment/Postfix Increment[x++]
b. Pre Increment/Prefix Increment[++x]

We are using ++ because one value will be increased

```
#include<stdio.h>
main()
{
   int x=12;
   printf("%d\n",x++);→13
   printf("%d",x);
```

x

~~12~~ 13

Print ① ② increment

→ 12

```
#include<stdio.h>
main()
{
   int x=12;
   printf("%d\n",++x);13
   printf("%d",x);13
```

Increment ① ② Print

Let us see another Example

```
#include<stdio.h>
main()
{
  int x=5, y=3,z;
  z=(x++)+y----------->8
  printf("%d",z);
}
```

```
#include<stdio.h>
main()
{
  int x=5, y=3,z;
  z=(++x)+y----------->8
  printf("%d",z);
}
```

Bitwise Operators

0,1===>bits

Bitwise Operator are used to perform operations on bits

C language provides following bitwise Operators

&	**Bitwise And**
|	**Bitwise Or**
^	**Bitwise XOR**
<<	**Left shift Operator**
>>	**Right Shift Operator**
~	**Bitwise Not/Complement**

In C language

T means 1
F means 0

First let us see the truth table for Bitwise AND, Bitwise OR, Bitwise XOR

x	y	x&y	x|y	x^y
1	1	1	1	0
1	0	0	1	1
0	1	0	1	1
0	0	0	0	0

1. In case AND if both bits are 1 then the result is 1 otherwise the result is 0

2. In case OR if atleast one bit is 1 the result is 1 other wise the result is 0

3. In case of XOR if opposite bits are there then the result is one if similar bits are there then the result is zero

x=12 y=13

let us perform bitwise AND operation

let us convert 12 into binary language

let us perform bitwise AND operation

let us convert 12 into binary language

$$\begin{array}{r|l} 2 & 12 \\ \hline 2 & 6-0 \\ \hline 2 & 3-0 \\ \hline & 1-1 \end{array}$$

$2 \Rightarrow 1100$

let us convert 13 into binary language

$$\begin{array}{r|l} 2 & 13 \\ \hline 2 & 6-1 \\ \hline 2 & 3-0 \\ \hline & 1-1 \end{array}$$

$3 \Rightarrow 1101$

.

x==>12===>1100
y==>13===>1101

x&y=====>1100

$2^3 2^2 2^1 2^0$
1 1 0 0

$1\times2^3+1\times2^2+0\times2^1+0\times2^0$

$8+4+0+0=12$.

Let us see how to convert a number into binary format and binary number into decimal format

$12 \rightarrow$

$$\begin{array}{r|l} 2 & 12 \\ \hline 2 & 6-0 \\ \hline 2 & 3-0 \\ \hline & 1-1 \end{array}$$

$\rightarrow 1100$

$\Rightarrow \overset{2^3}{1}\ \overset{2^2}{1}\ \overset{2^1}{0}\ \overset{2^0}{0}$

$\Rightarrow 1\times2^3+1\times2^2+0\times2^1+0\times2^0$

$\Rightarrow 8+4+0+0$

$=12$

$13 \rightarrow$

$$\begin{array}{r|l} 2 & 13 \\ \hline 2 & 6-1 \\ \hline 2 & 3-0 \\ \hline & 1-1 \end{array}$$

$\Rightarrow 1101$

$\Rightarrow \overset{2^3}{1}\ \overset{2^2}{1}\ \overset{2^1}{0}\ \overset{2^0}{1}$

$\Rightarrow 8+4+0+1$

$=13$

$9 \rightarrow$

$$\begin{array}{r|l} 2 & 9 \\ \hline 2 & 4-1 \\ \hline 2 & 2-0 \\ \hline & 1-0 \end{array}$$

1001

$\overset{2^3}{1}\ \overset{2^2}{0}\ \overset{2^1}{0}\ \overset{2^0}{1}$

$=1\times2^3+0\times2^2+0\times2^1+1\times2^0$

$$1001$$
$$= 1\times2^3+0\times2^2+0\times2^1+1\times2^0$$
$$= 8+0+0+1$$
$$=9$$

$$17 \Rightarrow \begin{array}{r|l} 2 & 17 \\ 2 & 8-1 \\ 2 & 4-0 \\ 2 & 2-0 \\ & 1-0 \end{array}$$

$$\rightarrow 10001$$
$$\rightarrow \overset{2^4}{1}\overset{2^3}{0}\overset{2^2}{0}\overset{2^1}{0}\overset{2^0}{1}$$
$$= 1\times2^4+0\times2^3+0\times2^2+0\times2^1+1\times2^0$$
$$= 16+1$$
$$=17$$

x==>12===>1100
y==>13===>1101

x&y=====>1100

$$\rightarrow \overset{2^3}{1}\overset{2^2}{1}\overset{2^1}{0}\overset{2^0}{0}$$
$$\Rightarrow 8+4+0+0$$
$$= 12$$

```
#include<stdio.h>
main()
{
    int x=12,y=13,z;
    z=x&y;
    printf("%d",z);
}
```

x==>13===>1101
y==>9===>1001
x&y=====>1001

$$\overset{2^3}{1}\overset{2^2}{0}\overset{2^1}{0}\overset{2^0}{1}$$
$$\Rightarrow 1\times2^3+0\times2^2+0\times2^1+1\times2^0$$
$$\Rightarrow 8+0+0+1$$
$$= 9$$

```
#include<stdio.h>
main()
{
    int x=13,y=9,z;
    z=x&y;
    printf("%d",z);
}
```

x==>13===> 01101
y==>17===>10001
x&y=====>00001

```
#include<stdio.h>
main()
{
    int x=13,y=17,z;
    z=x&y;
    printf("%d",z);
}
```

$$12 \rightarrow \begin{array}{r|l} 2 & 12 \\ 2 & 6-0 \end{array}$$

12 → 2|12
2|6 − 0
2|3 − 0
|1 − 1

→ 1100
$2^3\ 2^2\ 2^1\ 2^0$
⇒ 1 1 0 0
⇒ $1\times2^3+1\times2^2+0\times2^1+0\times2^0$
⇒ 8+4+0+0
= 12

13 → 2|13
2|6 − 1
2|3 − 0
|1 − 1

⇒ 1101
$2^3\ 2^2\ 2^1\ 2^0$
⇒ 1101
⇒ 8+4+0+1
= 13

9 → 2|9
2|4 − 1
2|2 − 0
|1 − 0

1001
$2^3\ 2^2\ 2^1\ 2^0$
1 0 0 1
= $1\times2^3+0\times2^2+0\times2^1+1\times2^0$
= 8+0+0+1
=9

17 → 2|17
2|8 − 1
2|4 − 0
2|2 − 0
|1 − 0

→ 10001
$2^4\ 2^3\ 2^2\ 2^1\ 2^0$
→ 10001
= $1\times2^4+0\times2^3+0\times2^2+0\times2^1+1\times2^0$
= 16+1
=17

Bitwise OR (|)
x==>12===>1100
y==>13===>1101

x|y=====>1101

$2^3\ 2^2\ 2^1\ 2^0$
1 1 0 1
8+4+1
⇒ 13

```
#include<stdio.h>
main()
{
   int x=12,y=13,z;
   z=x|y;
   printf("%d",z);
}
```

$2^3\ 2^2\ 2^1\ 2^0$
1 1 0 1
8+4+1
= 13

x==>13===>1101
y==>9===>1001
x|y=====> 1101

#include<stdio.h>

```
main()
{
   int x=12,y=13,z;
   z=x|y;
   printf("%d",z);
}
```

```
x==>13===>  01101
  y==>17===>10001
 x&y=====>11101
```

```
#include<stdio.h>
main()
{
   int x=13,y=17,z;
   z=x|y;
   printf("%d",z);
}
```

$2^4 2^3 2^2 2^1 2^0$

1 1 1 0 1

16+8+4+1

29

Left shift Operator

Symbol: <<(two less than symbols)

left shift operator means multiply with 2

```
x=12
x<<2

x=12--->x*2=12*2=24*2=48
```

```
#include<stdio.h>
main()
{
   int x=12,z;
   z=x<<2;
   printf("%d",z);
}
```

```
x=12
x<<3
x=12--->x*2=12*2=24*2=48*2=96
```

```
#include<stdio.h>
main()
{
   int x=12,z;
   z=x<<3;
   printf("%d",z);
}
```

```
x=10
x<<4
x=10--->x*2=10*2=20*2=40*2=80*2=160
```

```
#include<stdio.h>
main()
{
   int x=10,z;
   z=x<<4;
   printf("%d",z);
}
```

Right shift Operator

Symbol: >> (two greater than symbol)

Left shift operator means divide with 2

```
x=12
```

x>>2

x=12--->x/2=12/2=6/2=3

```
#include<stdio.h>
main()
{
    int x=12,z;
    z=x>>2;
    printf("%d",z);
}
```

x=18
x>>3
x=18--->x/2=18/2=9/2=4/2=2

```
#include<stdio.h>
main()
{
    int x=18,z;
    z=x>>3;
    printf("%d",z);
}
```

x=20
x>>4
x=20--->x/2=20/2=10/2=5/2=2/2=1

```
#include<stdio.h>
main()
{
    int x=20,z;
    z=x>>4;
    printf("%d",z);
}
```

Bitwise Complement/ Bitwise NOT

It performs NOT Operation

x	~x
1	0
0	1

Take the weights of 0s with -sign and add them and also add -1

$12 \rightarrow$

$$\begin{array}{r|l} 2 & 12 \\ 2 & 6-0 \\ 2 & 3-0 \\ & 1-1 \end{array}$$

$\rightarrow 1100$

$\Rightarrow \overset{2^3}{1}\,\overset{2^2}{1}\,\overset{2^1}{0}\,\overset{2^0}{0}$

$\Rightarrow 1\times2^3+1\times2^2+0\times2^1+0\times2^0$

$\Rightarrow 8+4+0+0$

$=12$

$12 \rightarrow 1100$

$\sim12 \rightarrow \overset{2^3}{0}\,\overset{2^2}{0}\,\overset{2^1}{1}\,\overset{2^0}{1}$

-2^3-2^2-1

$\Rightarrow -8-4-1$

$=-13$

$13 \rightarrow$

$$\begin{array}{r|l} 2 & 13 \\ 2 & 6-1 \\ 2 & 3-0 \\ & 1-1 \end{array}$$

$\Rightarrow 1101$

$\Rightarrow \overset{2^3}{1}\,\overset{2^2}{1}\,\overset{2^1}{0}\,\overset{2^0}{1}$

~13

1101

$\overset{2^3}{0}\,\overset{2^2}{0}\,\overset{2^1}{1}\,\overset{2^0}{0}$

$2 \lfloor \underline{1-1}$

$\rightarrow 1101$

$2^3 2^2 2^1 2^0$

$\Rightarrow 1101$

$\Rightarrow 8+4+0+1$

$= 13$

~ 13

1101

$2^3 2^2 2^1 2^0$

0010

$-8-4-1-1$

$-12-1-1$

$= -13-1 = -14$

$9 \rightarrow 2 \lfloor \underline{9}$

$2 \lfloor \underline{4-1}$

$2 \lfloor \underline{2-0}$

$\lfloor \underline{1-0}$

1001

$2^3 2^2 2^1 2^0$

1001

$= 1\times2^3+0\times2^2+0\times2^1+1\times2^0$

$= 8+0+0+1$

$=9$

$9 \rightarrow 1001$

$2^3 2^2 2^1 2^0$

$\sim 9 \rightarrow 0110$

$-8-1-1$

$= -10$

$17 \rightarrow 2 \lfloor \underline{17}$

$2 \lfloor \underline{8-1}$

$2 \lfloor \underline{4-0}$

$2 \lfloor \underline{2-0}$

$\lfloor \underline{1-0}$

$\rightarrow 10001$

$2^4 2^3 2^2 2^1 20$

$\rightarrow 10001$

$= 1\times2^4+0\times2^3+0\times2^2+0\times2^1+1\times2^0$

$= 16+1$

$=17$

$17 \rightarrow 10001$

$2^4 2^3 2^2 2^1 2^0$

$\sim 17 \rightarrow 01110$

$\Rightarrow -16-1-1$

$\Rightarrow -17 \quad -1$

-18

```
#include<stdio.h>
main()
{
   int x=12,z;
   z=~12;
   printf("%d",z);
}
```

```
#include<stdio.h>
main()
{
   int x=13,z;
   z=~13;
   printf("%d",z);
}
```

```
#include<stdio.h>
main()
{
   int x=9,z;
   z=~9;
   printf("%d",z);
}
```

```
#include<stdio.h>
main()
{
   int x=17,z;
   z=~17;
   printf("%d",z);
}
```

Operator Precedence and Associativity

In mathematics we have BODMAS rule

In C language we have Operator Precedence. Operator precedence tells the order of performing Operations .

Ex:

```
#include<stdio.h>
main()
```

In C language we have Operator Precedence. Operator precedence tells the order of performing Operations .

Ex:

~~**1+2*3**~~
~~**3*3**~~
~~**9**~~

1+2*3
1+6
7 ✓

2-10/3
2-3
-1

(1+2)*3
3*3
9

```
#include<stdio.h>
main()
{
    int a=1,b=2,c=3,d;
    d=a+b*c;
    printf("%d",d);
}
```

```
#include<stdio.h>
main()
{
    int a=2,b=10,c=3,d;
    d=a-b/3;
    printf("%d",d);
}
```

```
#include<stdio.h>
main()
{
    int a=1,b=2,c=3,d;
    d=(a+b)*3;
    printf("%d",d);
}
```

()	high
*,/,%	
+,-	low

(1+2)*3 ⇒ 3*3=9
① ②

2+3*4-6/3
③ ① ④ ②

4-6/3+7*3
③ ① ④ ②

Precedence	Associativity
*,/,%	Left to Right
+ -	Left to Right

2+12-6/3
2+12-2
14-2
12

```
#include<stdio.h>
main()
{
    int a=2,b=3,c=4,d=6,e;
    e=a+b*c-d/b;
    printf("%d",e);
}
```

4-6/3+7*3
4-2+21
2+21
23

```
#include<stdio.h>
main()
{
    int a=4,b=6,c=3,d=7,e;
    e=a-b/c+d*c;
    printf("%d",e);
}
```

Associativity

1.When multiple operators have same level priority[precedence] then associativity will be used

2.Associativity is used in two formats

- **a. left to right**
- **b. right to left**