Write a C program Using algorithm, flowchart And Expression

- **Algorithm** :- A logical step by step method to solve the problem is called the algorithm.

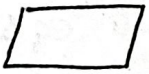
An algorithm includes calculation, reasoning and data processing. It can be presented by natural language, pseudocode, and flowchart. etc.

Flowchart :- It is graphical or pictorial representation of an algorithm with the help of different symbols, shapes & arrows to demostrate a process or program.

Following std symbols are applied in flowchart.

Terminal box - Start / End

Input / Output -



Process / Instruction

Decision

Connector / Arrow

→ If flowchart is a movie, then algorithm is story of that movie.

→ An algorithm is core of flowchart.

→ An algorithm can be expressed & analyzed through a flowchart.

Difference between Algorithm & Flowchart

Algorithm	Flowchart
I) It is procedure of solving Problem	I) It is a graphical representation of a process.
II) The process is shown in step by step instruction	II) The process is shown in block by block information diagram.
III) It is complex & difficult to understand.	III) It is intuitive and easy to understand.
IV) It is convenient to debug error	IV) It is hard to debug error
V) The solution is showcased in natural language.	V) The solution is showcased in pictorial format.
VI) It is somewhat easier to solve complex problem	VI) It is hard to solve complex problem
VII) It costs more time to create an algorithm.	VII) It costs less time to create a flowchart.

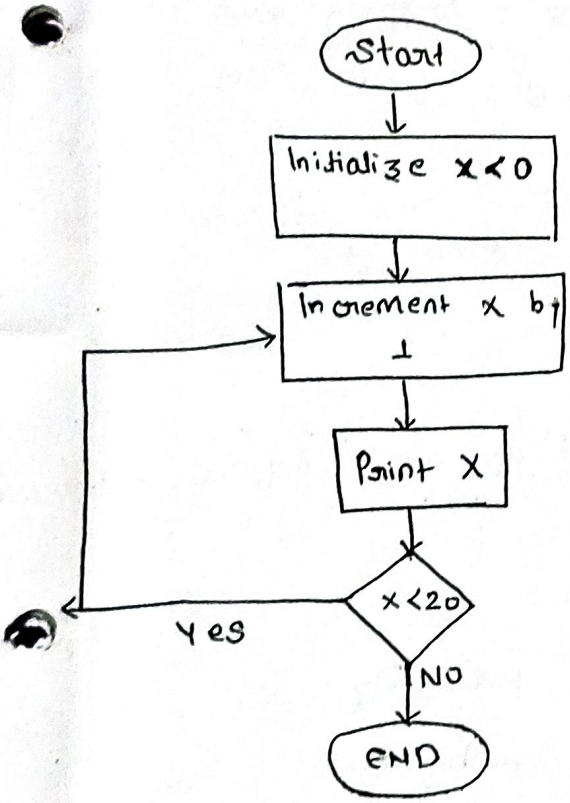
Eg. 1

Print 1 to 20

Algorithm:

- Step 1 : Initialize x as 0
- Step 2 : Increment x by 1
- Step 3 : Print x
- Step 4 : If x is less than 20 then go back to step 2.

Flowchart:



Eg 3

Determine whether a student Passed the Exam or not.

Algorithm:

- Step 1 : I/p grades of 4 courses M_1, M_2, M_3, M_4
- Step 2 : Calculate the average grade with formula = $\left(\frac{m_1+m_2+m_3+m_4}{4}\right)$

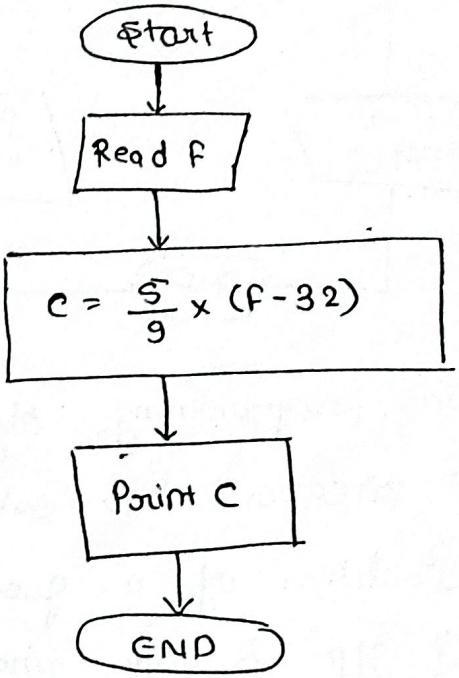
Eg 2

Convert Temp from (°F) to(°C)

Algorithm:

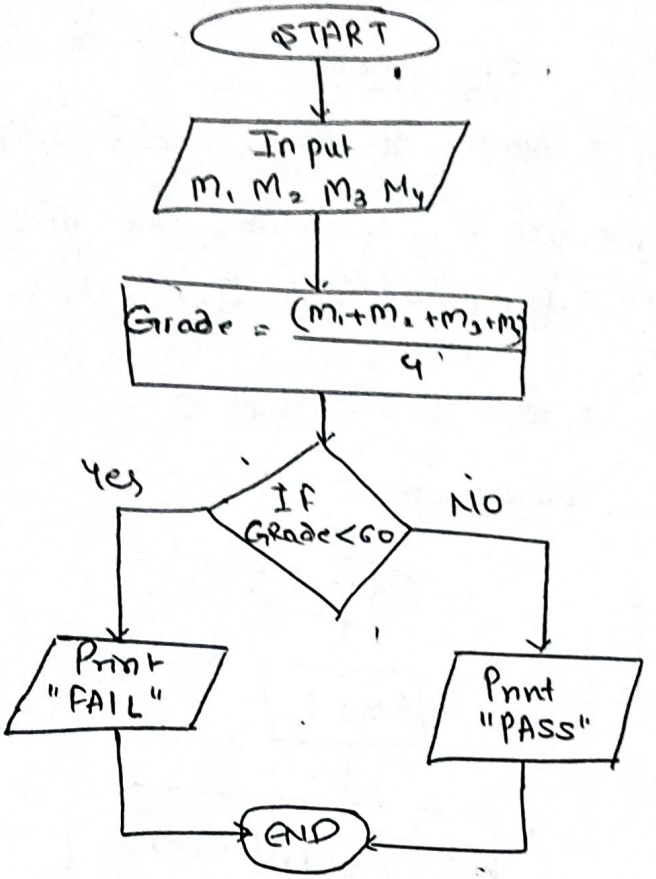
- Step 1 : Read temp in fahrenheit.
- Step 2 : Calculate temp with formula $C = \frac{5}{9} \times (F-32)$
- Step 3 : Print C

Flowchart:



Step 3 : If the average grade is less than 60 print "FAIL" & else print "PASS".

Flowchart :



Write an algorithm to add 2 entered by user

- Step 1 : Start
- Step 2 : Declare variables num1, num2 and sum
- Step 3 : Read values num1 and num2.
- Step 4 : Add num1 & num2 & assign result to sum
- Step 5 : Display sum
- Step 6 : STOP

Def:

→ In programming, algorithm is a set of well defined instructions in sequance to solve the problem.

→ Qualities of a good algorithm

i) I/P & o/p should be defined precisely.

ii) Each step should be clear & unambiguous

iii) It should be most effective among different ways of solving problems

iv) It should not have computer code.

Flowchart

Definition :- Flowchart is a graphical representation of an algorithm. a step by step approach to solve a task.

It makes use of symbols which are connected among them to indicate flow of information & processing.

Importance of flowchart :-

i) Flowchart are better way of communicating the logic of a system.

ii) Flowchart act as a guide for blueprint during program design

iii) Flowchart help in debugging process

iv) With the help of flowchart program can be easily analysed

Disadvantages :-

i) It is difficult to draw flowchart for large & complex program

ii) It is difficult to modify the flowchart

iii) It is difficult to reproduce flowchart

Symbols used in flowchart :-

	Symbol	Purpose	Description
1)	→	Flow line	Indicates the flow of logic by connecting symbols.
2)	⬭	Terminal (Start / Stop)	Represents the start and end of a flowchart
3)	▱	Input / Output	Used for i/p & o/p operation
4)	▭	Processing	Used for arithmatic operation & data representation
5)	◇	Decision	Used for decision making between two or more alternatives.
6)	○	On page connector.	Used to connect two or more parts of a flowchart, which are on the same page
7)	⬠	Off page connector	Used to connect two parts of a flowchart which are to be read over different pages

- Guidelines for developing flowchart :-

i) Flowchart can have only one start & one stop symbol

ii) On page connectors are referenced using number

iii) Off page connectors are referenced using alphabets.

General flow of process is top to bottom or left to right. Arrows should not cross each other.

- Pseudo codes :- Pseudocode is an informal high-level description of a computer program or algorithm.

→ It is written in symbolic code which must be translated into a programming language before it can be executed.

e.g. → Code to check if the user entered number is odd/e

```
num : INPUT  " Enter a number"
IF  num  MOD 2 == 0
        print  " EVEN Number"
ELSE
        print  " Odd number"
```

- Guidelines for writing Pseudocode :-

i) Use capital letter for reserved commands or keyword
   eg. IF ELSE

ii) Write only one statement per line.

iii) Use plain english to provide particular description

e.g. →
```
age = INPUT : " Enter your age"
IF age is greater than 18
    PRINT " adult"
ELSE PRINT " Under age"   ENDIF
```

```
FOR   i -> 0 to 20
PRINT i
ENDFOR
```

- **Advantage** :-

i) Increase code readability

ii) Provides best way to write algorithm

iii) Best approach to represent how the actual program will be written.

Basic Structure of C Program

C Program is divided into different sections. There are main sections to a basic C program.

Documentation Section :-

→ It consists of a set of comment lines giving the details like Name of program, The author name and other details like time of coding & description.

e.g. -
```
/* File Name : Helloworld C
/* Author    : Mohan
```

2) Link Section :-

→ It consist of declaration of header files that will be used in program.

→ This section provides instruction to compiler to link header files to the system libraries.

e.g. - `#include <stdio.h>`

3) Definition Section :-

→ This section defines all symbolic constants. The keyword define is used in this part.

Eg ⇒ `#define Pi = 3.14`

4) Global declaration section :-

→ In this section of code global variables are declared

→ The variables which are used in more than one function are known as global variable.

→ This section also declares user-defined functions.

```
float area(float r);
int a=7;
```

5) Main function section :-

→ Every C program must have one main() function section.

→ This section contains two parts : Declaration part, Execution part

→ Declaration part declares all variables used in execution part

→ There must be atleast one statement in execution part

→ These two parts remain enclosed in curly brackets. The opening of bracket indicates start of program & closing of bracket logically indicates end of program.

→ All statements in declaration and execution section end with a semicolon(;).

eg →
```
int main (void)
{
    int a=10;
    Printf ("%d", a);
    return 0; }
```

6) Sub-Program section :-

→ This section contains all used defined function which are called in main function.

eg →
```
int add (int a, int b)
{ return a+b; }
```

* All sections except main section may remain absent when they are not required.

Sample Program

```
/*  File Name : area of circle
/*  Author    : Mohan
/*  description : A program to calculate
                  area of a circle
#include <stdio.h>
#define pi = 3.14;
float area (float r);

int main()
{
    float r;
    Printf (" Enter the radius:n");
    scanf ("%f", &r);
    Printf ("the area is: %f, area(r));
    return 0;
}
float area (floatr)
{ return  pi*r*r;
}
```

Documentation Section

// link Section
// definition section
// global declaration Se.
// main() function section
// start of program

// Declaration Part

Execution part

// & user defined function
// sub program

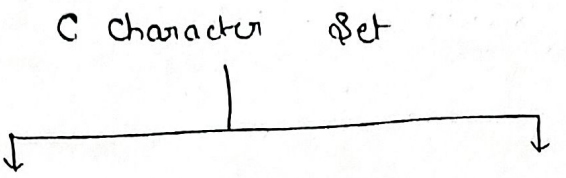
Output

```
Enter the radius :
7
The area IS : 153.860001
--------------------------
Program exited with code : 0
Press any key to continue . . .
```

Data Concept

Character Set :- Character set is fundamental raw material of any language and they are used to represent information.

The characters in C are grouped in following categories



Source Character Set

a) Alphabets A-Z, a-z
b) digits 0,1,2,3,4,5,6,7,8,9
c) Special character
d) White space

Execution character set

e) Escape sequance

Special character

~	tilde	\	back slash	?	question mark
%	Percent	(	left parenthesis	.	dot operator
|	Vertical bar	*	asterisk	}	right flower brace
@	at symbol	)	right parenthesis		
+	plus	'	apostrophe		
<	less than	:	colon		
_	underscore	[	left bracket		
-	minus	"	quotation mark		
>	greater than	;	semicolon		
^	caret	]	right bracket		
#	number sign	!	exclamation mark		
=	equal to	,	comma		
&	ampersand	{	left flower brace		
$	dollar sign				
/	slash				

White space Character

\b	blank space
\t	horizontal tab
\v	vertical tab
\r	carriage return
\f	form feed
\n	new line
\\	Back slash
\'	single quote
\"	double quote

\? question mark

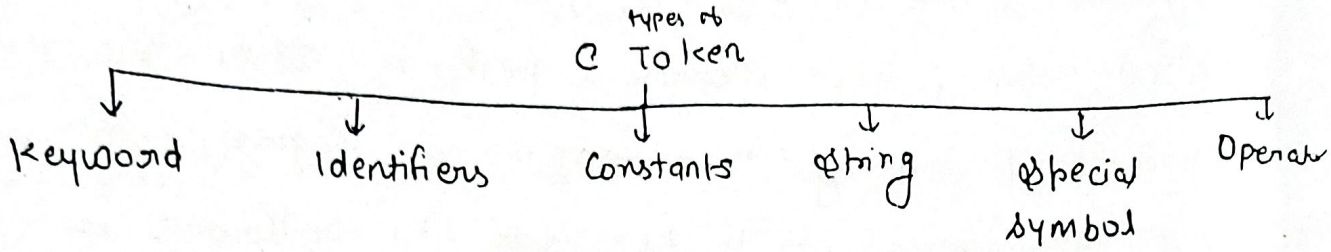
\0 Null

\a alarm (bell)

- Execution character set :- These character set don't Print ~~from~~ ASCII value, these characters perform other function. e.g. backspacing, moving to a newline, or ringing a bell. These are represented by a backslash (\) followed by a character or a combination of digit, this combination is known as escape sequance

Escape sequance	ASCII Value	character Result
Null	000	\0
Beep sound	007	\a
moves previous position	008	\b
moves next horizontal tab	009	\t
moves next line	010	\n
moves vertical tab	011	\v
moves initial position of next page	012	\f
moves beginning of line	013	\r
Present double quotes	034	\"
Present apostrophe	039	\'
Present ques mark	063	\?
Present back slash	092	\\
Octal		\ooo
hexadecimal number	\x	

C Token :- Smallest individual ~~word~~ element (word, punctuation) in C is known as C token. This is building block for creating a program in C language



- Keyword :- Every C word is classified as either a keyword or an identifier. All ~~the~~ keywords have fixed meaning that can not be changed. Keywords are always written in lower case. C language supports 32 keywords.

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

- Identifier :- Identifiers refer to the name of variables, functions and arrays. These are user defined and consist of alphabet, digits, underscore.

Rules for constructing identifier in C are given below

i) The 1st character of an identifier should be either an alphabet or an underscore followed by any character set

ii) In identifier both uppercase & lowercase are distinct i.e case sensitive

iii) Commas or blank space cannot be used

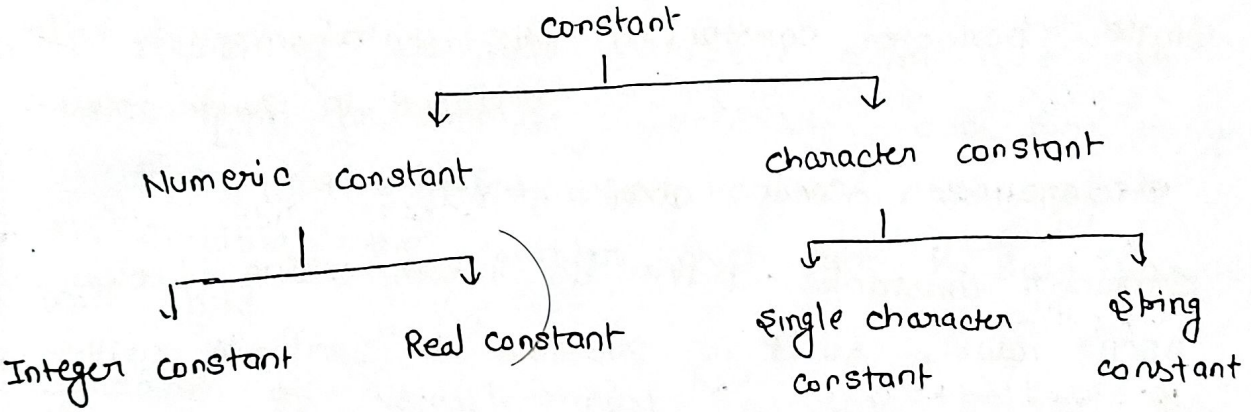
iv) keywords cannot be represented as identifier

v) Length of identifier should not be more than characters.

vi) It should be short, meaningful & readable.

- **Constant** :- In C constants are fixed values that cannot be changed during the execution of Program.

Syntax Declaration - const data_type variable_name = value;



① **Integer Constant** :- Integer constant includes sequance of digits. Integer can be decimal (0-9) octal and hexadecimal. (0-9, A-F) (0-7)

Declaration - const int value = 400;
const int oct = 040;
const int hex = 0X4F;

* space, commas and non-digit character are not permitted between digits.

Real / float constant :- These quantities or includes numbers containing fractional part like 17.548. Such numbers are called real (floating point) constant. This constant is required to represent continously varing quantity like temp, distance, price. These numbers can be expressed as -

215.65 — 2.1565e2
malissa e(E) exponent

→ declaration :- const float pi = 3.14;

→ Always exponent must be integer.

③ Single character constant :- This contains single character enclosed in single quote.

declaration const char gender = 'F';

character constants have an integer value known as ASCII value, so it is possible to perform arithmatic operation using this.

④ String constant :- A string constant is a sequance of character enclosed in double quote. The character may be letter, number, special character, blank space.

declaration const char name[] = "Data-Analysis!!"

• Variable :- A variable is a name given to a storage area that our program can manipulate. Each variable in C has a specific type

which determines the size & layout of variable m... the range of value that can be stored within that memory and set of operation that can be applied the variable. The variable name can be composed of letters, numbers & underscore.

Declaration type variable_name = value;

int x_val = 0;

- Declaration of variable :-

→ After designing suitable variable name we must declare them to compiler. Declaration does two things -

i) It tells the compiler what variable name is.

ii) It specifies the compiler which type of data variable will hold.

→ Declaration of variable must be done before they are used in program.

Primary type declaration

syntax data-type variable_name;

int count;

float ratio;

User defined type declaration

Syntax - typedef type identifier;

type → existing data type

identifier → new name given to data type

typedef int unit

unit batch1, batch2;

Data type declaration Instruction in C

Instruction :-

- Program statements are called instructions
- Instructions are commands
- Types of instructions
    1) Data type declaration instruction
    2) I/O instruction
    3) Arithmatic instruction
    4) Control instruction

- **Data type :-**

→ Keywords : int char float double void

→ The data type which are keywords are primitive data type.

- **Declaration statement :-**

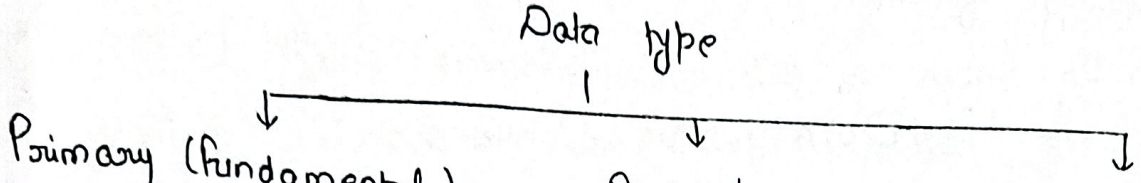
```
int    a, b = 5;     (b assigned 5)     a [2B]  b [2B]    dos based architecture
float  k;                               k [4B]
char   ch, m;                           ch [1B]  m [1B]
double d1;                              d1 [8B]
```

These declaration statements are for compiler, so that compiler can read the variables name.

Declaration statement should always be before action statement

Data type

In C programming, data types are declaration for variables. This determines type & size of data associated with variable.



- Primary (fundamental) Data type
  - integer (int)
  - character (char)
  - floating point (float)
  - double precision floating point (double)
  - void
  - long int
  - long double
- Derived data type
- User defined data type
  - Array
  - function
  - structure
  - Pointer

Size and range of data type in 16-bit machine

Type	Size (bits)	Range
char or signed char	8	-128 to 127
unsigned char	8	0 to 255
int or signed int	16	-32,768 to 32,767
unsigned int	16	0 to 65535
short int | signed short int	8	-128 to 127
long int | signed long int	32	-2,147,483,648 to 2,147,483,647
unsigned long int	32	0 to 4,294,967,295
float	32	3.4E-38 to 3.4E+38
double	64	1.7E-308 to 1.7E+30

long double 80 3.4E-4932 to 1.1E+4932

- Void :- This type has no value. This is used to specify type of functions. The type of a function is said to be void type if it does not return any value to the calling function.

Data Type Conversion

Data type conversion is basically conversion of one data type to another. There are two types of conversion

1) Implicit Type Conversion :- This is also known as 'automatic type conversion'. It is done by compiler without user trigger.
This takes place when in an expression more than one data type is present., in such condition, type conversion takes place to avoid loss of data. All the data types of variable are upgraded to the data type of variable with largest data type.. It is possible for implicit conversions to lose information, signs can be lost. (when signed is implicitly converted to unsigned).

eg
```
#include <stdio.h>
int main()
{
    int  x = 10;
    char y = 'a';
    x = x + y;    // y is implicitly converted to int type
```

float z = x + 1.0 ; // x is implicitly converted to float type

bool < char < short int < int < unsigned int < long < unsigned long < long long < float < double < long double

- Explicit type conversion :- This is also called type casting and it is user defined. Here the user can type cast the result to make it a particular data type.

Syntax : (type) expression

e.g. :-

```
int main()
{
    double x = 1.2;
    int sum = (int)x + 1; // Explicit conversion from double to int
    Printf ("sum = %d", sum);
    return 0;
}
```

⇒ Output :-

sum = 2

Arithmatic Instruction in C language

Operator :- Perform operation with operand (data)

3 + 4

Operand operator

- Arithmatic Instruction :- An instruction which is used to manipulate data using operator is known as Arithmatic instruction

3+4 * 5 = 23 By rule of precedence (tells which operator first)

BODMAS - Bracket of Division Multiplication Add Sub

There is no BODMAS in C language

- Operator Groups :-

Precedence Order

Unary Operator :- Need only one operand
Arithmatic operator
Bitwise operator
Relational operator
Logical operator
Conditional operator
Assignment operator

- Unary Operator :-

Need only one operand

+, - → indicates sign of number
++ → increment operator
-- → Decrement operator

Operator

→ An operator is a symbol that tells the computer to perform certain mathematical or logical operation on values or variables. eg. +, −, /, *, <, >

→ An expression is a sequance of operands and operators

e.g. 10 + 15

→ C operators can be classified into a no. of categories : They

1) Arithmatic Operators :- C supports all arithmatic operators.

Operator	Description
+	adds two operands
−	subtract second operand from 1st
*	Multiply two operands
/	Divide numerator by denominator
%	modulo division

Let a and b are variables known as operand. & a=4, b=6 then,

a−b = −2
a+b = 10
a*b = 24
a/b = 0 — [if both a & b are int type]
a%b = 4 ↳ a/b = 0.66... [If any of a & b is float type]

Program :- Use integer arithmatic to convert a given number of days into months & days.

```
main ()
int months, days;
printf ("Enter day \n");
scanf ("%d", &days);
months = days / 30;
days = days %30;
Printf ("Months = %d  Days = %d", months, days); }
```

Output :-

```
Enter days.
265
months = 8  Days = 25
```

(2) Relational Operator :-

This operator is used to compare two operands and take certain decision depending on relation.

Operator	Meaning
<	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

Relational Operator Complements

>	is complement of	<=
<	is complement of	>=
==	is complement of	!=

(3) Logical Operator :-

The logical operator && and || are used when we want to test more than one condition & make decision. C has 3 logical operators

Symbol	Meaning
&&	logical AND
||	logical OR
!	logical NOT

Truth Table

Operand-1	Operand - 2	Value of Expression Op-1 && Op-2	Op1 || Op-2
Non-zero	Non-zero	1	1
Non-zero	0	0	1
0	Non-zero	0	1
0	0	0	0

Relative precendence of the relational & logical operator is as follows :

Highest !

>>= <<=

== !=

&&

Lowest ||

(4) Assignment Operator :-

The assignment operator assigns the result of an expression to a variable.

There are some shorthand assignment operators along with assignment operator '='.

assignment statement v = v op exp; [here v is a variable op= is shorthand operator] is equal to

v = v op exp

e.g. x += (y+1); is equal to x = x +(y+1);

Statement with simple assignment Operator	Statement with shorthand operator
a = a+1	a += 1
a = a-1.	a -= 1
a = a * (n+1)	a *= (n+1)
a = a/(n+1)	a /= (n+1)
a = a%b	a %= b

⑥ Increment & Decrement Operator :-

Increment operator (++) adds 1 to the operand while decrement operator (--) subtracts 1. They are represented as

++m or m++
--m or m--

where m is a variable

++m ; is equivalent to m += 1;
--m ; is equivalent to m -= 1;

A prefix operator (++m or --m) first add 1 to the operand and then assigns the result to variable on left.

```
m = 5;
y = ++m;
```

Result : y = 6, m = 6

A postfix operator (m-- or m++) first assigns the value to variable on left and then increments the operand.

```
m = 5;
y = m++;
```

Result : y = 5, m = 6;

(7) Conditional Operator :-

A ternary operator pair "? :" is available in C to construct conditional expression of the form

exp1 ? exp2 : exp3

where exp1, exp2, exp3 are expressions.

Here exp1 is evaluated first, if it is true (non zero) then exp2 is evaluated and becomes value of expression. If exp1 is false (zero), exp3 is evaluated and becomes value of expression.

Let a = 10;
b = 15;

x = (a > b) ? a : b;

Result : x = b

This can also be achieved using if else statement.

(8) Bitwise Operator :-

For manipulation of data on bit level bitwise operator is used.

Operator	Meaning
&	bitwise AND
|	bitwise OR
^	bitwise exclusive OR
<<	shift left
>>	shift Right

Special Operators :-

C supports some special operators like sizeof, pointer, member selection :, comma operator.

(i) The comma operator can be used to link related expressions. the expression as evaluated from left to right.

```
value = ( x = 10, y = 5, x+y );
```

comman operator has lowest precedence of all operators. So the parenthesis are necessary.

(ii) sizeof operator returns the no of bytes occupied by operand. Operand may be constant, variable, or data type qualifier.

```
m = sizeof (sum);
```

Input / Output functions

* The main() is a special function used by C system to tell the computer where the program starts
The empty pair of parethesis () after main indicates that the function has no argument.

- Each program that uses a standard input / output function must contain the statement.

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

This tells the compiler to search for a file named stdio.h and place its contents at this point in the program.
Abbrevation for stdio.h is Standard input output header.
stdio.h is not necessary for printf and scanf input, output function as they have been defined as a part of C language

- **Printf() :-** printf() function is one of the main output library function in C programming which prints captions and numerical results on screen.
To print int type data at output %d is used.
Similarly to print float %f, ~~float~~ double %lf.
character %c, format specifier is used.

```
#include <stdio.h>
int main()
{
printf("C Programming");
int x = 6; float y = 13.5; char chr = 'a';
Printf("x = %d", x);
Printf("y = %f", y);
Printf("character = %c", chr);
return 0; }
```

o/p
C Programming
x = 5
y = 13.5
character = a

scanf() :-

scanf is a function which reads formatted input from the std. input device as keyboard.

e.g:-

```
# include <stdio.h>
int main()
{ int testInteger;
  Printf ("Enter an integer: ");
  scanf ("%d", &testInteger);
  Printf ("Number = %d", testInteger);
  return 0;
}
```

O/P ⇒

```
Enter an integer : 4
Number = 4
```

→ Here %d is a format specifier to take int type input & &testInteger gets the address of testInteger. and the value entered by user is stored in that address.

→ %f, %lf, %c are format specifier for float, double and character respectively.

```
#include <stdio.h>          // Program to print char & ASCII value
int main()
{ char chr;
  Printf ("Enter a character : ");
  scanf ("%c", &chr);
  printf ("You entered %c. \n", chr);
  Printf ("ASCII Value is %d.", chr);
  return 0; }
```

O/P :-

```
Enter a character : g
your entered g.
```

- getch() :-

→ getch() is a predefined non-standard function that defined in conio.h header file.

→ It is mostly used by MS-DOS's compiler like Turbo C

→ getch() is used -

to hold screen until the user passes a single value to exit from console screen.

to read a single byte character or string from keyboard & then print.

We can hide input character provided by the user in ATM PIN, password etc.

→ Syntax : int getch(void);

→ Parameter : The getch() function does not accept any parameter from user.

→ Return value : It returns the ASCII value of the key Pressed by the user as input.

→ It does not hold any parameter. It has no buffer area to store the input character in program.

★ Write a simple program to display the character entered by the user using the getch() function in c.

```
#include <stdio.h>
#include <conio.h>
int main()
{   printf("Passed value by the user is : %c", getch());
    return 0;   }
```

Passed value by the user is : P

* Write a simple program to hold the console screen until the user gives a key to exit

```
# include <stdio.h>
# include <conio.h>
int main()
{  printf ("Enter a key to exit the console screen. \n");
   getch();
   return 0; }
```

o/p - Enter a key to exit the console screen.

* Write a program to use the getch() function to accept the string from the user.

```
# include <stdio.h>
# include <conio.h>
int main()
{  char ch[6] = {0}; int x;
for (x=0; x<5 ; x++)
{  ch[x] = getch(); }
   printf ("Received 5 character input : %s\n" ch);
   return 0;  }
```

o/p :- Received 5 character input : hello;

* Write a program to use the getch() function to accept the hidden password from the user

putch() function :-

It is used for printing a character to a screen at current cursor location. It is unformatted character o/p functions It is defined in header file conio.h

Syntax : putch(Character);

```
main()
{  char ch;
   clrscr();
   printf(" Press an character: ");
   ch = getch();
   printf("\nPressed character is : ");
   putch(ch);
   getch(); }
```

putch() prints only one character at a time

```
void main()
{  char ch='a';
   putch(ch)
}
```

- **Getchar() :-** Getchar() is a function in C programming language that reads single character from std i/p and returns it to the program

Syntax int getchar(void);

```
# include <stdio.h>
  int main()
{
   printf("%c", getchar());
```