

NAME : Shaheed Falke



SUBJECT : Programming C STANDARD : 1<sup>st</sup> year

## SECTION :

ROLL NO. : 35

# INDEX

SR.NO.	DATE	TITLE	PAGE NO.	REMARKS
		ASCII - American Standard Code for Information Interchange.		
		A-Z → 65-90		
		a-z → 97-122		
		0-9 → 48-57		
		Special symbols → 0-47, 58-64, 91-96, 123-127.		
		ASCII - ordinal	→ In python	
		ASCII - key code	→ In .net	

## \* History:

- Basics of different programming field
- Basic of c - design to computer hardware
- Programming in 'c'
- Decision making statements
- Loop / Iterative statement
- switch case
- Functions & pointers
- Array and String
- Structure & union
- file Handling

## \* Tools -

Application - set of tools

software - set of applications

## # Sequence

Tools → Applications → software

- \* There are no of programming languages.
- \* Every programming language is to develop different types of application.
- \* C / C++ → console application, small-scale games.
- HTML / ASP → static web - application.
- XML, ASP, .Net, PHP → Dynamic web appln.

Imp - Assembly lang. → circuit programming, compiler design.

V.B.net, C#, visual C++ → windows based,  
V.B'6 → GUI application.  
Network-based appln, console application.

Java-script, VB script → To add script into web-application.

PERL → Drug application, Biological application

Python → console appln, GUI application, web-appln, Automation application, Data analysis application.

Java → console appln, windows GUI appln, Network based appln, Linux based application, Web servers.

- # Android = Java + XML
- \* To make software programming language is not important compiler is important.
- \* Compiler is set of tools which can perform specific tasks.
- Compiler tools →
  1. Error checker
  2. Assembler
  3. Linker
  4. Debugger
  5. Interpreter
  6. Translator

Now we know that, numbers of programming lang.

& every programming language is specially to perform different technical tasks for which it was invented.

- \* A programming language is actually set of instruction or code that is understandable to its compiler.
- \* Any prog. language inventor never invents a prog. language, but they actually invent compiler of that language.
- \* A compiler is actually set of tools, that are to perform specific tasks in program. Different tools in compiler in other word compiler of every programming language is different.

For Ex: In C & C++ compiler, we find  
Error checker, Translator, Linker, etc.

- \* In Java: Appletviewer, java c

java

jdb

java doc

java h

java p

tools

- \* Different programming lang. have different features & because of these features a lang. can perform that specific technical tasks only.

- \* To study any programming language we mainly required.

1. Compiler of that language

2. IDE or Editor of that language

IDE - Integrated Development Environment

c/c++ → vs code

code blocks

Java → Net beans  
Eclipse

Text pad

Python → py charm

IntelliJ IDE

Idle Python

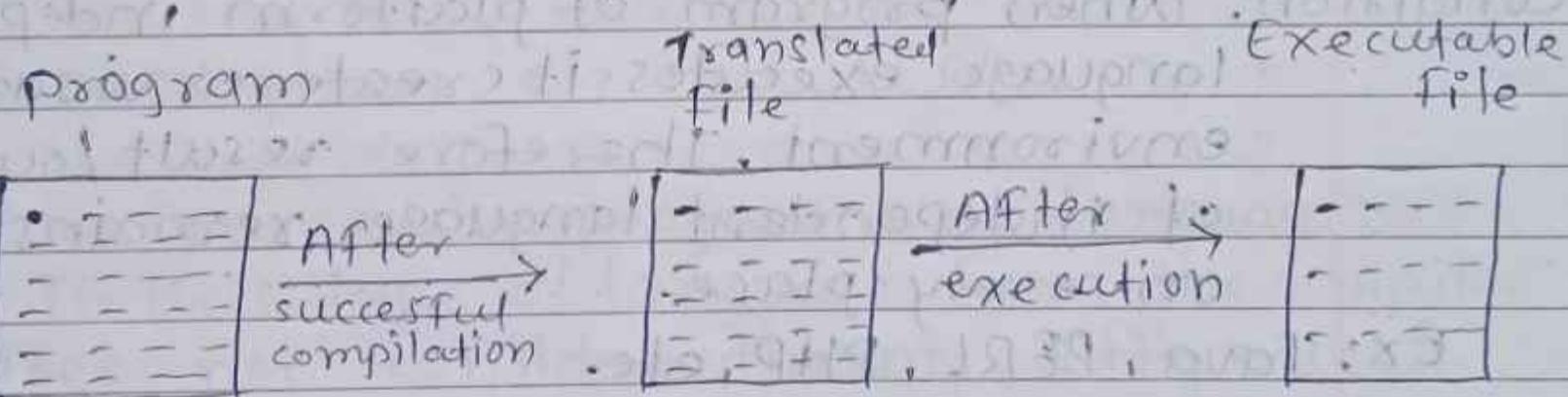
Jupiter Netbook

## # Types of programming language based on platform

There are two types of prog. lang. based on platform

1. Platform dependent

2. Platform independent



one.c /

one.obj

one.exe

There are two types of programming language based on platform.

## 1. Platform dependent programming language:

- This type of languages directly executes on platform & adopts some feature from the platform on which it is executing. Therefore some calculation / result of platform language may vary based on execution platform.

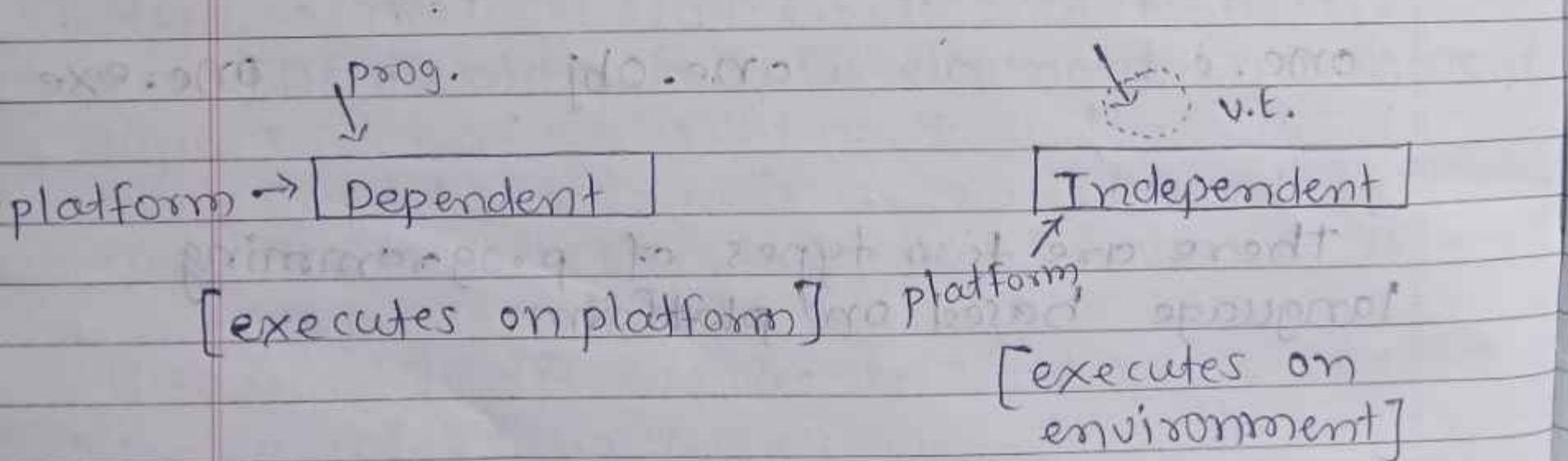
Ex: C, C++, VB = visual Base

## 2. Platform independent programming language:

- This type of language are independent of the platform on which it is executing.

Condition: When program of platform independent language executes it creates its own environment. Therefore result / output of independent language remains same on every place.

Ex: Java, PERL, PHP, etc.



Note: Machine lang. is platform dependent language

# Note: A platform dependent lang. has to land on the platform therefore a platform dependent language must have to create a separate translated machine lang. file & this machine language file executed.

### \* Types of lang. [Based on case-sensitivity]

1. Case sensitive prog. language → Ex. Java, C, C++
2. case insensitive prog. lang. → Ex. HTML, XML, ASP

#### 1. Case sensitive programming language:

In this type of language from the compiler consider the difference between upper case & lower case letters type in program.

Ex: Java, Python, C, C++, PERL

#### 2. Case Inensitive programming language:

In this type of language from the compiler does not consider the any difference between upper case & lower case letters are type in program.

Ex: HTML, XML, ASP

## \* Types of programming language (Based on source code availability)

1. Open source language:-

2. Proprietary language:-

C/C++ → Headers files

Java → Packages

Python → Modules

Android → API

VB, VB.net → Namespace

1. Open source languages:

In this type of language from original source code of their library is freely available form the programmers. We programmers can read, study and can modify that source code on personal level.

Ex: Python, Java, PERL, R, etc.

2. Proprietary languages:

In this type of language's from original source code of their libraries is not freely available. In such language, libraries are provided either in encrypted format or in executable format so that programmers can't read & understand source code.

Ex: C, C++, VB.C, VB.net, etc.

Note!

A library is ultimately a program file that constant pre define code that can be directly useful to programmer for any logical implementation. In simple words a library is pre-written code file almost every programming language provide library to help programmers.

At any place we want to contribute in a C language. Then we should created & share library with other programmers.

Ex: Header file in C/C++ use library for C/C++

\* Types of programming language (Based on compilation & Interpretation process).

There are two types of programming language based on compilation & interpretation process.

1. Compiled language
2. Interpreted language

Compilation includes → Error checking

Linking (Linker)

Assembly

Translating

Interpretation includes → Run | Execute  
(uses JIT)

[JIT - Just In TIME]

## 1. Compiled language:

A programming language in which first entire your program gets compiled and then the entire program executes.

Remember that compilation does not include only error checker but there are other phases that happen during compilation such as Assembling, Linking, Translating.

Also remember that every compile language produces a translated file / encrypted file, after compilation process.

Ex: C, C++, etc.

## 2. Interpreted language:

In interpreted language performs statements by statements compilation & execution. This means each statement individually compiles and runs with the help of JIT & interpreter.

Ex: Python, Java, R, VB.net, etc.

Note:- If we compile an interpreted language it interprets the statement only under controls unless therefore these are changes that some of statements get seppet even if the code was tested multiple time.

Therefore there are changes of buggers in this lang. in opposed to these, the compile lang. check for error completely therefore such program execution only when there are no errors in program.

Compiled lang → Interpreter

Error checker

Assembler

Linker

Translator

Debugger

Interpreted lang → Interpreted

JIT

Monitor

Garbage collector

## # Types of language (Based on program structure)

There are 2 types of language based on program structure.

### 1. Structured language:

In this of language from open statements are not allow every executable statement must be inside body of function in other words from .

## 2. Scripting language:

In this type of language from open statements are allowed that is executable statements can be outside functions body.

In this type of language we define function only for purpose reusability.

Reusability → write once use multiple time

Ex: Python, Java, R, PERL, etc.

# A function is set of statement.

\* These statement under function → body

Function name → Method

→ rational Definition

→ procedure

\* Any word followed by () is function name.

stelio.h → 57

Math.h → 84

Imp → When we use these function its body executes.

## Function

library fun

pre-defined fun  
built-in fun

user defined function

whose body is  
defined by programmers.

whose body is already first define then use  
defined



Ex: main()

Directly use it



Ex: printf(),

scanf(), etc.

- Based on body, there are 2 types of functions

1. Library

2. User defined

\* Body of library function are in specific header file.

∴ To use body of function we have to include that header file.

#include <header\_file\_name.h>

### # Note:

We know that any word followed by parameter is function name and sometimes we use function which has some values in between parenthesis. These values are called as parameters or arguments.

for ex: `printf ("Hello all")` } → parameters  
`scanf ("%d, &a")` } → parameters  
`clear ()` } → parenthesis  
`getch ()` } → parenthesis

### \* Note:

We can also include header file within double quotes (" ") instead of (<>).

Imp. ← `#include <stdio.h>`  
`#include "stdio.h"`

### \* Types of programming language (Based on features)

There are 3 types of lang. based on features

- i) Low level language.
- ii) Middle level language
- iii) High level language.

### i) Low-level programming language:

1. Stores data in binary form.
2. Directly understandable to processors.
3. Don't have datatypes & memory blocks.
4. Don't have any support for permanent data.
5. Most suitable for circuit programming & compiler design.
6. Stores data on electric components.
7. Performs only basic operations.
8. Works on very smaller values only.

### ii) High-level programming language:

1. Stores data in original string form.
2. Works on user values.
3. Have variety of operator and operations.
4. Have variety of library to perform all action application.
5. supports data bases and file for permanent storage.
6. Have variety of data type to work with various types of values.

### iii) Middle level - programming language:

1. Combines from some features low level and some high level language.
2. Store data in binary form.

3. Requires translators that translates into machine language field.
  4. supports only handling for permanent data store. (No data support)
  5. Perform intermediate operation.
  6. Have basic data types.
- \* Types of prog. lang. (Based on Memory Allocation)

there are memory management of c and c++ is above memory matrix were memory blocks, each of 1 byte gets created now every variable we declared occupies memory on that

Based on allocation of memory, there are two approach in c and c++.

1. Top down approach
2. Bottom up approach

### 1. Top down approach:

In top down approach of memory allocation, the variables occupies memory from beginning memory matrix, row wise. C language has top down approach of memory allocation.

## 2. Bottom up approach:

In bottom up approach of memory allocation of the variable occupies memory from last block from last at the end C++ has bottom up approach of memory allocation.

Note:

Only C & C++ have such concepts of memory matrix, whereas all high language like Java, Python have concept of "Hashtable" of memory location.

\* c - character set: ~~long ago~~ go until it's

This is a list of valid characters that are allowed in a C program.

Alphabets : A-Z , a-z are called variables

Digits : 0 to 9

special symbols: + , - , \* , / , %  
> , < , ; , = , ? , ,  
.. , # , & , ( ) , [ ]

- \* Any mathematical & scientific constants are not known to any compiler.
  - \* Any mathematical & scientific symbols are not known to any compiler ( $\pi$ ,  $\theta$ ,  $d$ ,  $B$ , etc.).
  - \* Any mathematical & scientific formulae are not known to any compiler.  
(Programmer has to write formulae as statement).
  - \* Any measurement units are not allowed in programs work only on values. Never use units during inputs & formulae.

## Memory Blocks:-

1. Is an area where runtime values are stored.
  2. Memory-blocks get created only during runtime.

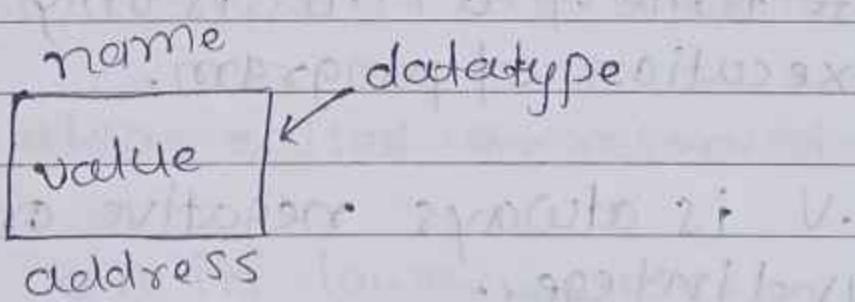
\* A memory-block(s) has 4 attributes:

1. Name (Identifier)
2. Value
3. Datatype
4. Address

Identifiers  $\Rightarrow$  Any name

\* Rules of giving an identifier /name

1. Should be of max 33 and min 1 character language.
2. Space is not allowed.
3. Special symbols are not allowed only under score (-) is allowed.
4. Must begin with alphabet or underscore (-) [must not begin with digit].
5. Keywords are not allowed as identifier.



\* As a programmer we can give name, datatype and value.

\* whereas, address is automatically assigned by compiler.

- \* As a programmer, if address is logically required then we can print or use the address.
- \* But we can't generate or change or delete address.
- \* For any memory disk Block (M.B) name & datatypes are mandatory to give.
- \* If we declare a M.B then a M.B is never generated empty.
- \* A M.B is always have an auto-generated value. This auto-generated value is called as 'Garbage Value'.
- \* When we assign value, our value will overwrite garbage value (G.V).
- \* Life time of a M.B is only upto end of execution of program.
- \* G.V is always negative or positive, non-zero and integer.
- \* Address is always positive integer and non-zero.
- \* During runtime we can not change name or datatype.

\* Based on value, there are two nature of a memory-blocks:

1. Variable

2. Constant

1. Variable: whose value can be changed during runtime.

2. Constant: whose value can not be changed during runtime.

\* Note:

In any programming language by default a memory block is variable.

\* Keywords:

- keywords are the words whose meanings are already known to compiler.
- we cannot use keywords as identifier.
- keywords are also called as reserved words.
- All keywords are in lower-case.
- In 'c' we have 32 keywords.

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

### \* Datatype

- Is to specify which type of value we are going to store in a memory-block.

Numeric → Integer - Numeric value without decimal  
Numeric → Real - Numeric value with decimal point

Alphabetic → character - stores alphabets, special symbols.

[For use only Turbo C]

### Data type

Integer		Real		Character
int	long	short	float	char
-32768 to +32767( $2^{15}$ )	-2147483648 to +2147483647( $2^{31}$ )	-32768 to +32767	34E-17 to 34E+17	double
2 bytes	4 bytes	2 bytes	4 bytes	long double
%d	%ld	%Ld	%f	38E-308 to 38E+308
				1 byte
				%c

\* In C & C++ Range is Cycle.

Datatype			
Integer	Real	Character	Boolean
short -32768 to +32767 2 bytes %ld	int -2147483648 to +2147483647 4 bytes %d	long -9223372036854775808 to +9223372036854775807 8 bytes %ld	No change No change
			bool Stores boolean TRUE boolean FALSE 2 bytes %b
			[for online gcc compiler or any other compilers use]

\* `int a;` → Declaration

`a = 5;` → Assigning

\* `int a = 5;` → Declaration + Assigning = Initialization

\* In case of Integer & Real interpreter stores its binary conversion.

\* In case of character, interpreter stores its ASCII value.

\* You perform any operation on character. It will be ultimately performed on its ASCII value.

\* A character value must be assigned & compared in single inverted comma.

## # Declaring variable and constant.

We know that there are 2 nature/types of a memory blocks:

1. Variable
2. Constant

1. Variable:  
Means whose value can be changed during runtime to declare variable general syntax.

datatype v-name; → `int age;`  
`float avg;`  
`char gen;`

datatype variable-list; → `int roll, age, enroll,`  
`float avg; wt, grade, char gen, minor;`

\* Functions name is also identifier.

Page No. \_\_\_\_\_  
Date \_\_\_\_\_

\* A memory block is by default is variable.

datatype v-name = value; → int age = 19

float pie = 3.14

char gen = 'M'

char minor = 'Y'

## 2. Constant:-

Means whose value can not be changed during runtime. To declare constant general syntax is

const datatype c-name = value;

For ex: const int age = 19;

const float pie = 3.14 ;

const char gen = 'M' ;

\* Here "const" is keyword

\* constant enliene must be initialize.

## # Headerfile

1. Any word followed by parenthesis () is a function name.

2. Every function has body. This body contains set of statements which execute when a function is called.

3. Based on body, there are two types of functions

i) Library function

ii) User defined function.

### i) Library function:

The functions whose body is already defined library function. As a programmer we have to directly call that function.

### ii) User-defined function:

whose body is defined by programmer. As a programmer first define function then call/use that function.

4. The body of library function are defined in specific headerfile. Therefore to use any library function we have to include that header i.e. -

#include < headerfile-name.h >

or #include "headerfile-name.h"

5. In program this statement is called as pre-processor directive or pre-processing directive.

6. Here # is an operator called as pre-processor.

printf()	malloc()	free()
scanf()	calloc()	
gets()	realloc()	
puts()		
fflush()		

clrscr()

getch() } conio.h  
getche()

exit() → process.h

sqrt()	{	strlwr()	}
pow()		strcpy()	
sin()		strwrd()	
cos()		strrev()	
tan()		strncpy()	
ceil()		strcat()	

oval	{	graphics.h
circle		
line		
color		

\* In C we have total 24 header file and mostly we include required header file in beginning of program.

\* Always remember the header file name and its library function in it.

\* Never include unnecessary header file in program.  
for ex. stdio → standard input-output  
(total 57 function)  
conio → console input-output  
(total 38 function)

Note:- A memory block never gets generated during compile, it always get created during runtime.

Page No.  
Date

### \* Types of Memory

#### 1. Heap memory:

- is permanent memory.
- heap memory stay upto end of execution.
- every declared variables, objects, arrays point creates memory into heap memory.

#### 2. Buffer memory:

- is temporary memory.
- buffer memory stay only upto execution of that single statement.
- result of expressions point messages input value initially goes to buffer memory.
- execution of function also happen in buffer memory.

\* In C/C++ default memory is heap memory.

\* In all high-level language default memory is buffer memory.

\* In C/C++ answer/result get generated in buffer memory and hold its copy in a M.B. which is in heap memory.

From above discussion we can say that the ultimate purpose of declaring variable is to create memory blocks in heap memory which hold / store generated answer from buffer memory.

### \* Input - output statements :-

Input statements:

Input statement is to accept / read values externally from input resource.

scanf( ), gets( ), f scanf( ), vf scanf( )

Output statements:

It is display / write message / result into resource.

printf( ), vprintf( ), f printf( )

puts( ),

putch( ),

f puts( ),

Note:- Ampersand

Address specifier operator

Address of operator

Page No.		
Date		

\* Facility

Escape sequence: \ %

Formatting sequences: In, It, Ia, Ifb, -

lw lh lr

Format specifier: %f, %c, %d, %dd  
%s %x %b

\* From above list note that we have many library function for input output operation but among them printf() is general purpose function for output messages and scanf() is generate purpose function for any input.

Notes: if we indicate formatting facility is enable to is that function.

i. Input statement:-

We use scanf as general purpose input statement to use scanf() is-

scanf(".format", &Vname);  
specifier

(2, "b & x = noibba") Matrix

x =

Ex. 1. int age;

scanf ("%d", &age);

2. float avg;

scanf ("%f", &avg);

3. long ac\_num;

scanf ("%ld", &ac\_num);

4. int age;

float avg;

scanf ("%d %f", &age, &avg);

It is not necessary to make input for every declared variable.

2. Output Statement:

We use the output instruction

message resulting values of output

resources we have printf (" ") general purpose output statement. To use printf()

general syntax is

printf ("Message here");

Ex 1. Addition is X

printf ("Addition is %d", c)

2. Addition = X

printf ("Addition = %d", c)

3. X

printf ("%d", c)

4. int a=5 ; float b=9.2

value of a is x & b is x

printf("value of a is %d & b is %f", a,b);

5. x is value of a & x is value of b

printf("%d is value of a & %f is b", a,b);

\* Note:

1. From above example, note that we do not use & (in person) in printf.

2. If we use & (in person) in printf then it will print address of variable, instead of value of.

3. Whenever & is use with variables name that it will deal with address of that variable.

&c -> address of c  
c -> value of c

## \* Operators:-

An operator is single or set of special symbol which has same pre-defined operational meaning.

- In C we have 3 types of operators.

1. Unary operators - which requires single operand to work with.
2. Binary operators - which requires two operand to work with.
3. Ternary operators - which requires three operand to work with.

Ternary  
operator

$\rightarrow$  C - Conditional operator ( $? :$ )

Unary operators { I - Increment operator ( $++$ )

                  D - Decrement operator ( $--$ )

L - Logical operator  $\&\&$  (LOGICAL AND)

$\vee\vee$  (LOGICAL OR)

$\neg\neg$  (LOGICAL NOT)

Binary  
operator

A - Arithmetic operator (+ - \* / %)  
(Modulus)

R - Relational operator ( $> < \leq \geq$ )

$\neq$  (Not Equal)       $=$  (Equal)

B - Bitwise operator  $\ll$  (LEFT SHIFT)

$\gg$  (RIGHT SHIFT)

$\&$  (BITWISE AND),  $\mid$  (BITWISE OR),

$\sim$  (BITWISE NOT),  $\sim$  (BITWISE XNOR)

\* Arithmetic operators:

• Arithmetic operators have to perform arithmetic operators are operands.

Operator	Description
+	To obtain addition
-	To obtain subtraction
*	To obtain multiplication
/	To obtain division
%	(Modulus) To obtain remainder

\* Note:- Every input/output function that ends with letter f allows formating during their operation. For any formating we have 3 options.

1. Format specifier
2. Escape sequence
3. Formating character

\* Escape sequence:- In C, we have 2 special symbols for escaping. They are \ and %. Both these escape sequences stops the special symbols to perform their respective operations.

\* For Ex.1. we use \ to prevent operations of formating characters in it.

For Ex: 2) `printf("we use \n for newline");`  
 Here, \n will work as newline symbol  
 And it will break output line.  
 But it will not get printed.

For Ex: (ii) `printf("we use \\n for newline")`  
 Here, \n gets printed. Because F.s \ will  
 stop \n to work.

Ex. 2. If we want to print "\n message, then  
 also we have to use escape sequence \ before ("\\n").

Ex. (i) `printf("My friend "Amit" is good-guy");`  
 will show CT error  
 ii) `printf("My friend \"Amit\" is good-guy");`  
 I will stop " to be considered as closing".  
 Hence, "Amit" gets printed.

Ex. 3. In case if we want to print any  
 format specifier like %d, %f, then we  
 have to use \ before format specifier to  
 prevent its working.

a) `printf("We use %d for int");`  
 will print any random value but %d will  
 not get printed.  
 b) `printf("We use \\%d for int");`  
 Here %d get printed.

Note:- Remember that any other operator has no operational meaning in `printf (" ")`. Anything we can write within `printf()` will be as message.

Ex: `printf ("c = a+b;");`

This will not perform any addition operation but this will get printed as message.

\* Note: In programming always copy/paste not cut/paste.

\* Note: Integer divide by integer answer is integer.

### • Arithmetic operation.

I. % (Modulus)

Ex: i) `a = 15 % 3;`

$$a = 0$$

ii) `b = 15 % 2;`

$$b = 1$$

iii) `c = 2 % 15;`

$$c = 2$$

iv) `f = 1 % 10;`

$$f = 1$$

v)  $e = 2793 \% 10;$

$e = 3$ , value will be lost, answer will be 0

(iv) Trailing of precision loss in floating point

(v) vi)  $y = 2793 \% 100;$  answer will be 93

$$y = 93$$

vii)  $j = 'A' \% 5;$

$j = 0$  because ASCII value of 'A' is 65

so answer will be 0 before the ASCII code

viii)  $k = 'Z' \% 4;$

$k = 2$  because ASCII value of 'Z' is 90

so answer will be 2

\* The % operator doesn't work on real values. (floating - point)

1. `int k = 7293 / 10;`

$$k = 729$$

2. `float d = 7293 / 10;`

$$d = 729.0000$$

3. `float m = 7293 / 100;`

$$m = 72.93$$

4. `int m = 7293 / 10.0;`

$$m = 729$$

### \* Increment operator:

- is to increase value of any variable by 1.

exp:

int a=5;  
a++

a int  
[56] ↗  
xxx

float b=9.2;

b++ float  
[9.2] ↗

char gen='M';

gen++ char

[M] ↗

xxx

a++;

++a;

a=a+1;

### \* Decrement operator:

- is to decrease value of any variable by 1.

a int

int a=5; a sub  
[5] ↗

xyz

float b=9.2;

b++; float

[9.2] ↗

xyz

char gen = 'M'  
gen --

gen  
PTL

char

a--;  
--a;  
a = a - 1;  
a - = 1;

~~at + (Post increment): first use/assign as it is then increase the value by 1.~~

~~+ + a (Pre increment): first increase value by 1 then use value.~~

~~a - - (Post decrement): first use / assign as it is then decrease value by 1.~~

~~-- a (Pre-decrement): first decrease value by 1 then use value.~~

\* Assignment operator :- (=)

1. Operator assigns value of right operand to the variable of left - operand.

2. Here, expression and value's must be on RHS and target variable must be of L.H.S.

Exp: c = a + b

\* The assignment operator has least / minimum precedence among all other operators.

\* Bitwise operators:-

- These operators works on binary - bits of operator.
- Both operands must be integer (Doesn't work on real values).

① LEFT shift operator ( $<<$ ):-

ex:  $c = a << b;$

This operator shift binary value of left operand towards left as per the value of Right operand.

② RIGHT shift operator ( $>>$ ):-

ex:  $c = a >> b;$

This operator shift binary value of left operand towards right as per value of Right operand.

③ Bitwise AND operator ( $\&$ ):-

ex:-

$$a = 75 \quad \text{&} \quad 155$$

$$= [a = 11]$$

128	64	32	16	8	4	2	1	
0	1	0	0	1	0	1	1	$\rightarrow 75$
1	0	0	1	1	0	1	1	$\rightarrow 155$

(दोनों पर्याप्त असेल तर एक थेणार)

This operator works on binary bits of both operands. If value of both operand bits is bit-1 then it results bit-1 otherwise results bit-0. Refer following table:-

LEFT operand bit	RIGHT operand bit	Result bit
1	1	1
0	1	0
1	0	0
0	0	0

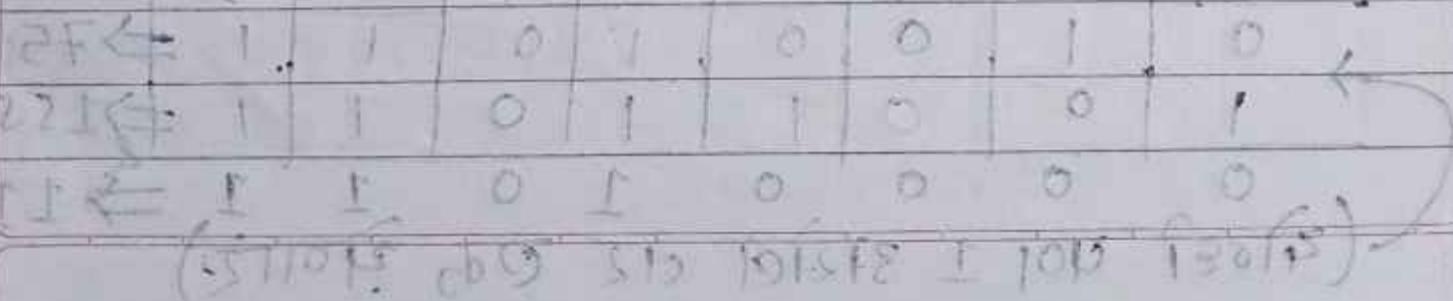
#### ④ Bitwise OR operator (|):-

LEFT operand bit : (<<)	RIGHT operand bit >	Result bit
1	<< 0 = 1	1
0	0	0

ex:

$$a = 75, 155 ;$$

128	64	32	16	8	4	2	1	
0	1	0	0	1	0	1	1	$\Rightarrow 75$
1	0	0	1	1	0	1	1	$\Rightarrow 155$
1	1	0	1	1	0	1	1	



This operator works on binary bits of both operands. This operator results bit -1 in any 1 operand bit is 1? otherwise results bit -0.

### ⑤ Bitwise NOT operator ( $\sim$ ):- (tilde)

$$\begin{array}{l} a = 75 \\ \boxed{a = 52} \end{array}$$

128	64	32	16	8	4	2	1	$\Rightarrow 75$
1	0	0	1	0	1	1	1	
0	0	1	0	0	1	0	0	

This operator is a unary operator. This operator inverts binary bits of operand.

Operand bit	Result bit
1	0
0	1

Q.1. Obtain binary equivalent of following:-

$$1) 135 (10000111) \rightarrow 0011 - (s \ll 0000) = d$$

$\rightarrow$

256	128	64	32	16	8	4	2	1	$\Rightarrow 135$
1	0	0	0	0	1	1	1	1	

2)  $27651(101011001101)$

 $\rightarrow$ 

2,048	1024	512	256	128	64	32	16	8	4	2	1
1	0	1	0	1	1	0	0	1	1	0	1

3)  $993(1111100001)$

 $\rightarrow$ 

1,024	512	256	128	64	32	16	8	4	2	1
1	1	1	1	1	1	0	0	0	0	1

4)  $500(111110100)$

 $\rightarrow$ 

512	256	128	64	32	16	8	4	2	1
1	1	1	1	1	1	0	1	0	0

Q.2. Evaluate following expressions:

①  $a = (135 \ll 2) - 80; (10000111)$

 $\rightarrow$ 

512	256	128	64	32	16	8	4	2	1
1	0	1	0	0	0	0	1	1	1
1	0	1	0	0	1	1	1	0	0

$540 - 80 \Rightarrow 460$

②  $b = (200 \gg 2) - (100 \ll 2) (000011)$

512	256	128	64	32	16	8	4	2	1
1	0	1	0	0	0	1	0	0	0
1	1	1	0	1	1	0	0	1	0
1	1	0	0	1	0	0	1	0	0

$56 - 400 \Rightarrow 350$

400

3)  $c = a \& b;$

$\rightarrow$

256	128	64	32	16	8	4	2	1
1	1	1	0	0	1	1	0	0
1	0	1	0	1	1	1	0	0
1	0	1	0	0	1	1	0	0

$d = (100 << 2) \& (199 + 25);$

$\rightarrow$

256	128	64	32	16	8	4	2	1
		1	1	0	0	1	0	0
		1	0	0	0	0	0	0

$\Rightarrow 400 + 224 = 624$

⑥ Bitwise XOR operator ( $\wedge$ ) ~~or~~ Exclusive OR operator  
 This operator is a binary operator &  
 works on binary bits of both operand.  
 This operator results bit-1 if both operand  
 bits are opposite, otherwise results bit-0.

LEFT operand bit	RIGHT operand bit	Result bit
1	0	0
0	1	1
1	1	0
0	0	0

For ex:  $a = 175 \wedge 200$

512	256	128	64	32	16	8	4	2	1
		1	0	1	0	1	1	1	0
		1	1	0	0	1	0	0	0
		0	1	1	0	0	1	1	1

## \* Skeleton of C program:-

include required header files

void main()

    declare variables

    input - values

    logical operations & expressions

    display resulting values

}

Note: ① C is a structured language.

② This means, every executable statement must be within body of a function.

③ In structured languages the execution controls jumps from one function's body to another function's body.

∴ In C, it is mandatory to define at least one function.

④ Therefore, interpreter has decided/ fixed one name of function. i.e. main.

⑤ That 'void' is one of return-type.

Note:- A program without main() method will get compiled but will not run/ execute.

- ① Write a program to print "Hello all" message.

```
#include <stdio.h>
#include <conio.h>
void main()
{
    clrscr();
    printf ("Hello all");
    getch ();
}
```

- ② Write a program to print your name, college name, city. all in new lines.

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

```
void main()
{
```

```
    clrscr();
```

```
    printf ("Sharad Falke \n");
```

```
    getch ();
```

```
    printf ("Masthwaada Institute of Technolo-
```

```
\n gy \n");
```

```
    getch ();
```

```
    printf ("Chhatrapati Sambhaji Nagar");
```

```
    getch ();
```

```
}
```

- ③ write a program to print your exact address. Use proper formating.

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

```
void main()
```

```
{
```

```
clrscr();
```

```
printf("At. Bhagwati, Tq. Vaijapur, In,  
Dist. Chhrapati Sambhajinagar")
```

```
getch();
```

- ④ write a program to accept age of student & display it with proper message.

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

```
#include <conio.h>
```

```
void main()
```

```
{
```

```
int age;
```

```
clrscr();
```

```
printf("Enter age of student : ");
```

```
scanf("%d", &age);
```

```
printf("The student is %d years old", age);
```

```
getch();
```

```
}
```

Comments:- In any programming language, comments is the section that neither compile nor execute/run.

Writing comments is always a good habit & it is sign of good programmer.

In a program, as a comment we can write

- short explanation of code
- alternate logic
- opening & closing of logical block.
- Any path over password of files.

Almost every programming language supports commenting & there are two types of comments -

1. single-line comment
2. Multi-line comment

1. Single line comment:- We have // as single-line comment symbol and the effect of this operator is only up to that single line.  
For ex:-

(1) // This is variable to store age of student

// This is single-line comment

// end of while loop.

2. Multi-line comment :- We have /\* \*/ as multi-line comment operator. The effect of multi-line comment is in between opening & closing of this operator.

For ex.  
① /\* here I write loop to check for valid single entry of value in array.

② This is previous correct logic

This is previous work.  
stable ( $\epsilon = 1$ ) versus fractional.

```
while (true) {  
    if (input == "q") {  
        break;  
    }  
    cout << "You entered " << input << endl;  
}
```

$\rightarrow$  transcription

1) *What do you think about the new school?*

foros de la Corte Suprema de Justicia de la Nación.

\* | *unpublished* *recording* *2011-11-11*

⑤ write a program to accept average marks of a student & display it with proper message-

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

```
#include <conio.h> // ①
```

frustración, ansiedad, tristeza, miedo, etc.

~~void main() { int a; } }~~

1

float average;

class();

```
printf("Enter average marks\n");
```

```

scanf ("%d", &average);
printf ("student's average marks = %.2f",
       average);
getch();
}

```

- ⑥ W.A.P to accept age, roll no & gender letter of student & display it with proper message.

```

#include <stdio.h>
#include <conio.h>
void main()
{
    int sage;
    float savg;
    char sgen;
    clrscr();
    printf ("Enter age of student\n");
    scanf ("%d", &sage);
    printf ("Enter average marks\n");
    scanf ("%f", &savg);
    printf ("Enter Gender letter\n");
    scanf ("%c", &sgen);
    printf ("Average marks = %f\n",
           savg);
    printf ("Gender = %f\n", sgen);
    getch();
}

```

- ⑦ Write a program to perform addition of two integers.

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

void main()
{
    int num1, num2, ad;
    clrscr();
    printf("Enter 2 values\n");
    scanf("%d %d", &num1, &num2);
    ad = num1 + num2;
    printf("Addition is %d", ad);
    getch();
}
```

Note:- *to avoid this kind of error* (3)

- Always prefer to use fflush() before taking character input or string input.
  - This becomes necessary if any other input is taken before character input or string input.
  - When any input is given, we press ENTER  to submit that input.
  - This ENTER also remains into buffer memory hence it stops next character input and considers previous ENTER as input value for character variable.
- ∴ By using fflush (stdin) we clear the buffer memory that removes previous [ENTER]. Hence the scanf() stops for character input.*

Note:-

- If we make habit of using fflush() before accepting every input, then also it is OK/valid. But, before character input and string input fflush is necessary.
- The parameter (which is written in () bracket) of fflush() is not of type string.

*∴ Do not use Double quotes (" ") for stdin.*

- ⑧ W.A.P to accept a number from the user & print its square and cube.

```
#include <stdio.h> // header file
#include <conio.h> // header file
void main()
{
    int a, sq, cube;
    printf("Enter any number\n");
    scanf("%d", &a);
    sq = a*a;
    printf("The square of no. is : %d", sq);
    cube = a*a*a;
    printf("The cube of no. is : %d", cube);
    getch();
}
```

- ⑨ W.A.P to accept marks of five subjects of a student . calculate and printing total marks & average marks.

```
#include <stdio.h> // header file
#include <conio.h> // header file
void main()
{
    int sub1, sub2, sub3, sub4, sub5, Total;
    float avg;
    printf("Enter physics marks :\n");
    scanf("%d", &sub1);
    printf("Enter chemistry marks :\n");

```

```

scanf("%d", &sub1);
printf("Enter JAVA marks :\n");
scanf("%d", &sub2);
printf("Enter DSA marks :\n");
scanf("%d", &sub3);
printf("Enter NCC marks :\n");
scanf("%d", &sub4);
scanf("%d", &sub5);
total = sub1 + sub2 + sub3 + sub4 + sub5;
printf("Total marks :%d", total);
avg = total / 5;
printf("Average marks :%f", avg);
getch();
}

```

- ⑩ W.A.P to accept a number from user and print its last digit.

```

#include <stdio.h>
#include <conio.h>
void main()
{
    int a, l;
    printf("Enter any number:");
    scanf("%d", &a);
    l = a % 10;
    printf("Last digit is :%d", l);
    getch();
}

```

- (11) w.A.P. to accept 2 numbers from user and print their sum of last digits.

```
#include < stdio.h >
#include < conio.h >
void main()
{
    int a, b, l, m, sum;
    printf("Enter 2 numbers");
    scanf("%d %d", &a, &b);
    l = a % 10; b = a / 10;
    m = b % 10;
    sum = l + m;
    printf("sum is %d", sum);
    getch();
}
```

- (12) w.A.P to accept from user and print its previous value and next value. must use increment, decrement operator.

```
#include < stdio.h >
#include < conio.h >
void main()
{
    int a, p, n;
    clrscr();
    printf("Enter any number\n");
    scanf("%d", &a);
    p = --a;
```

```

    printf("Previous number : %d \n", p);
    n = a + 2; // or n = ++a or a++;
    printf("Next number : %d ", c);
    getch();
}

```

- (13) W.A.P to accept to perform all arithmetic operations on 2 integers.

```

#include <stdio.h> // header file
#include <conio.h> // header file
void main()
{
    clrscr();
    int num1, num2;
    printf("Enter 2 values\n");
    scanf("%d %d", &num1, &num2);
    getch();
}

```

- (14) W.A.P to accept to perform all add, sub, multi & division operations.

```

#include <stdio.h> // header file
#include <conio.h> // header file
void main()
{
    clrscr();
    int n1, n2;
    int add, sub, multi;
    float div;

```

```

    clrscr();
    printf("Enter 2 values\n");
    scanf("%d%d", &n1, &n2);
    add = n1 + n2;
    sub = n1 - n2;
    multi = n1 * n2;
    div = n1 / (float)n2;
    printf("In Addition is %d", add);
    printf("In Subtraction is %d", sub);
    printf("In Multiplication is %d", multi);
    printf("In Division is %f", div);
    getch();
}

```

- 15) W.A.P to accept numbers of hours to travel to destination. convert the same hours, into numbers of minutes or seconds.

```

#include <stdio.h>
#include <conio.h>
void main()
{
    int hr;
    long min, sec;
    clrscr();
    printf("Enter number of hours");
    scanf("%d", &hr);
    min = hr * 60;
    sec = hr * 3600;
}

```

```

printf("%d hours is equal to %ld minutes\n"
      ":In", hr, min);
printf("%d hours is equal to %ld seconds\n"
      ":In", hr, sec);
getch();
}

```

- (16) W.A.P to accept radius of a circle & calculate area & circumference of that circle.

```

#include<stdio.h>
#include<conio.h>
void main()
{
    float rad, ar;
    clrscr();
    printf("Enter radius of circle\n");
    scanf("%f", &rad);
    ar = 3.14 * rad * rad;
    rad = 3.14 * rad * rad;
    printf("Area is %.f", ar);
    printf("Circumference is %.f", rad);
    getch();
}

```

F=3      P=1      C=3  
 F=2      P=2      C=2

(17) W.A.P. to swap values of two integers.

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int a, b, c;
    clrscr();
    printf("Enter 2 values in %d\n");
    scanf("%d%d", &a, &b);
    printf("In Before swapping, a=%d &\n"
           "b=%d", a, b);
```

c = a;

a = b;

b = c;

```
printf("In After swapping, a=%d &\n"
      "b=%d", a, b);
```

getch();

}

num = 175

a = num % 10; : a  $\Rightarrow$  5

b = num / 10; : b  $\Rightarrow$  17

c = b % 10; : c  $\Rightarrow$  7

s = a + c; : s  $\Rightarrow$  12

18) W.A.P. to accept a four digit number from the user & print sum of all its digits.

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int num, l, sl, tl, f, a, sum, b, c;
    clrscr();
    printf("Enter 4 digit number : \n");
    scanf("%d", &num);
    l = num % 10;
    a = num / 10;
    sl = a % 10;
    b = num / 100;
    tl = b % 10;
    c = num / 1000;
    f = c % 10;
    sum = l + sl + tl + f;
}
```

```
printf("sum of all 4 digits is %d", sum);
getch();
```

(Note)

19) W.A.P. to accept a four digit number from the user & print sum of first & last digit only.

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int num, l, sl, tl, f, a, sum, b, c;
    clrscr();
    printf("Enter a 4 digit number : ");
    scanf("%d", &num);
    l = num % 10;
    a = num / 10;
    sl = a % 10;
    b = num / 100;
    tl = b % 10;
    c = num / 1000;
    f = c % 10;
    sum = l + f;
    printf("Sum of first & last digit is %d", sum);
    getch();
}
```

(20) W.A.P. to accept any number from the user & print sum of its last 2 digits.

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int num, l, sl, a, sum;
    clrscr();
    printf("Enter any number : \n");
    scanf("%d", &num);
    l = num % 10; // last digit
    a = num / 10;
    sl = a % 10; // second last digit
    sum = l + sl;
    printf("sum of last two digits is %d", sum);
    getch();
}
```

Q21) W.A.P. to accept a 4 digit number from the user and reverse that number.

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int num, a, b, c, s;
    clrscr();
    printf("Enter any 4 digit value : ");
    scanf("%d", &num);
    a = num % 10;
    num = num / 10;
    b = num % 10;
    num = num / 10;
    c = num % 10;
    num = num / 10;
    s = (a * 1000) + (b * 100) + (c * 10) + num;
    printf("Reverse the number is %d", s);
    getch();
}
```

(22) Write a P to accept numbers of seconds and convert it into equivalent numbers of minutes.

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int min, sec;
    clrscr();
    printf("Enter seconds\n");
    scanf("%d", &sec);
    min = sec / 60;
    sec = sec % 60;
    printf("%d : %d", min, sec);
    getch();
}
```

- Q3) W.A.P. to accept a 5 digit number from the user & print sum of first and last digit.

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int f, l, sum;
    long num;
    clrscr();
    printf("Enter the five digit number : ");
    scanf("%ld", &num);
    l = num % 10;
    f = num / 10000;
    sum = l + f;
    printf("sum of first and last digit
    is %d", sum);
    getch();
}
```

Q. Write any 15 invalid variable names:-

- i) int a/c - number;
- ii) long papa's mob number;
- iii) @ name;
- iv) my - variable;
- v) user & name;
- vi) int #n1;
- vii) int age +;
- viii) long sum;
- viiii) float bank - acc - no;
- x) long double stel - mo - no;
- xii) char gender later;
- xiiii) long adhar - card no;
- xiii) int enroll =;
- xiv) int sec =;
- xv) long fred's no;

### \* Rules of giving an identifier:

- 1) Maximum 33 characters and minimum 1 character.
- 2) Space is not allowed.
- 3) Special symbols are not allowed. Only Under score (-) is allowed.
- 4) Must begin with alphabet or underscore. [must not begin with digit]
- 5) Keywords are not allowed as identifier.

(25)

A cashier has currency notes of 100, 50 & 10. W.A.P to accept amount to withdraw by a customer & print how many cashier notes of each denomination the cashier should give.

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

```
#include <conio.h>
```

```
void main()
```

```
{
```

```
int amount, hundreds, fifties, tens;
```

```
printf("Enter the amount withdraw : ");
```

```
scanf("%d", &amount);
```

```
hundreds = amount / 100;
```

```
amount % 100 = 100;
```

```
fifties = amount / 50;
```

```
amount % 50 = 50;
```

```
tens = amount / 10;
```

```
printf("Num of 100 notes : %d\n", hundreds);
```

```
printf("Num of .50 notes : %d\n", fifties);
```

```
printf("Num of 10 notes : %d\n", tens);
```

```
getch();
```

```
}
```

num → 4516  
O/p → 5627

Page No.

Date

- (26) W.A.P. to accept a 4 digit number from the user & print the result in number by adding 1 in each digit.

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

Void main()
{
    int num, a, b, c, s;
    clrscr();
    printf("Enter any 4 digit value\n");
    scanf("%d", &num);
    a = num % 10;
    num = num / 10;
    b = num % 10;
    num = num / 10;
    c = num % 10;
    num = num / 10;
    a = ++a;
    b = ++b;
    c = ++c;
    num = ++num;
    s = (num * 1000) + (c * 100) + (b * 10) + a;
    printf("Increase each digit by 1 = %.d\n");
    getch();
}
```

(27)

W.A.P to accept distance betn two cities in km & convert the same distance into metre / feet.

$$1 \text{ km} = 1000 \text{ metre}$$

$$1 \text{ m} = 3.38 \text{ feet}$$

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

```
#include <conio.h>
```

```
void main()
```

```
{
```

```
int k, m;
```

```
float f;
```

```
clrscr();
```

```
printf("Enter distance in kilometers\n");
scanf("%d", &k);
```

```
m = k * 1000;
```

```
f = m * 3.28;
```

```
printf("%d kilometers\n %d meters\n
      In %d feet:", k, m, f);
```

```
getch();
```

```
}
```

int a = 5, b = 10, c = 15;  
 cout << a + b + c;

int d = 20, e = 30, f = 40;  
 cout << d + e + f;

int g = 50, h = 60, i = 70;  
 cout << g + h + i;

int j = 80, k = 90, l = 100;  
 cout << j + k + l;

(28) W.A.P. to accept temperature of city. in degree celsius & print its equivalent temperature in degree fahrenheit. (0 pt)

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

```
void main()
{
```

```
    int d;
```

```
    float f;
```

```
    clrscr();
```

```
    printf("Enter temperature in degree
```

```
celcius\n");
```

```
    scanf("%d", &d);
```

```
    f = (d * 9 / 5) + 32;
```

```
    printf("%d degree celcius in %f
```

```
fahrenheit", d, f);
```

```
getch();
```

- (29) Write a program to accept a 4 digit number from the user & print the result all numbers by 1 and 9 to 0.

#include <stdio.h>

int main()

{

int number, result = 0;

clrscr();

printf("Enter a 4-digit number: ");

scanf("%d", &number);

if (number >= 1000 && number <= 9999) {

int t = (number / 1000 + 1) % 10;

int h = ((number / 100) % 10 + 1) % 10;

int tens = ((number / 10) % 10 + 1) % 10;

int o = number % 10 + 1) % 10;

result = t \* 1000 + h \* 100 + tens \* 10 + o;

printf("Output is %d\n", result);

} else {

printf("Please enter a valid four-digits number.\n");

}

## Decision Making Statements

**Relational operator :-** Relational operator are to check conditions. We use these conditions to check correctness of any situation.

Operator	Description
> (greater than)	Results TRUE if value of left operand is greater than value of right - operand ; otherwise results FALSE.
< (less than)	Results TRUE if value of left operand is less than value of right operand ; otherwise results FALSE.
!= (not equals)	Results TRUE if value of left operand is not equals to value of Right - operand ; otherwise results FALSE.
==	Results TRUE if value of left operand is equal to value of right - operand ; otherwise results FALSE.

a) if statements;

if (condition)

{  
statements;  
statements;

- Here if is a keyword.

if condition is prepared by using  
relational operator.

\* if specified condition is TRUE, then  
only block of statement executes.

\* if specified condition is FALSE, then  
block of statement get skipped.

(30) W.A.P to accept total bill amount from the user & print new bill amount after giving 15% discount.

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

void main()
{
    float totalBillamount, discount, newBillamount;
    clrscr(); clrscr();
    printf("Enter the total Bill amount:");
    scanf("%f", &totalBillamount);
    discount = 0.15 * totalBillamount;
    newBillamount = totalBillamount - (totalBillamount * discount);
    printf("New bill after applying a 15% discount is %.2f", newBillamount);
    getch();
}
```

(31) W.A.P. to swap two integers without using third variable.

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

void main()
{
    int a, b;
    clrscr();
    printf("Enter any 2 values\n");
    scanf("%d %d", &a, &b);

    printf("Before swapping. a=%d and
           b=%d", a, b);

    a = b - a;
    b = a + b;
    a = b - a;

    printf("After swapping, a=%d and
           b=%d", a, b);
}
```

v.v.Imp

- \* C & C++ are loosely datatype language.
- \* Java is strongly datatype language.

Data		

\* Type conversion:- Type conversion is a process in which we convert value of 1 datatype into another datatype.  
Based on different logical situation, there are 2 types of type conversion -

- 1) Implicit type conversion
- 2) Explicit type conversion

① Implicit type conversion : In this type conversion value of 1 datatype automatically gets converted into another datatype. Therefore it is also called as automatic type conversion.

for example:-

int a=9.2;

a  $\Rightarrow$  9

float b=35;

b = 35.0000

int c='A';

c  $\Rightarrow$  65

char d='z';

z  $\Rightarrow$  90

char d='A';

d  $\Rightarrow$  65

\* Here, type conversion happens based on datatype of target variable.

② Explicit type conversion : Explicit type conversion In this type of type conversion, we have manually specify the datatype to be converted into. Mostly we use explicit type conversion in expressions & logical statements.

For example:-

```
[int b=(int) 75/3.0;]
```

```
[float c=a/(float)b;]
```

\* Explicit type conversion is also called type casting.

\* Logical possibilities of if-statement:-

1) Simple if-statement or Single if-statement:-

```
if (condition)
```

```
{
```

```
statements;
```

```
statements;
```

```
}
```

2) Multiple if-statement or if-ladder:-

`if (condition-1) {  
 Statement;  
}`

`? } (Condition-2)  
if (condition-2)  
{  
 Statement;  
}`

`? } (Condition-3)  
if (condition-3)  
{  
 Statement;  
}`

`? } (Condition-4)  
if (condition-4)  
{  
 Statement;  
}`

\* In multiple if-statements all if-statements are independent.

3) Nested if-statement examples:

`if (condition-1){  
 ?  
}`

`? if (condition-2)  
? {  
 ?`

`? if (condition-3)  
? {  
 ?`

`? } (Condition-2)  
? } (Condition-3)  
? } (Condition-4)`

`Statement;  
} (Condition-1)`

If their execution of statements are dependent on correctness of multiple conditions, then prefer nested if-statement.

4) Nested multiple if-statement:-

```
if (condition-1)
{
    if (condition-2)
    {
        statements;
    }
    if (condition-3)
    {
        statements;
    }
    if (condition-n)
    {
        statements;
    }
}
```

5) Multiple nested if-statements:-

```
if (condition-1)
{
    if (condition-2)
    {
        statements;
    }
}
```

```

if (condition - 3)
{
    if (condition - 4)
    {
        if (condition - 5)
        {
            statements;
        }
    }
}

```

- (32) W.A.P to accept a number from the user & print whether it is greater than 10 or less than 10.

```

#include <stdio.h> //subr. f
#include <conio.h> //atm. b/w
void main()
{
    int a;
    clrscr(); //clear screen
    printf("Enter any number\n");
    scanf("%d", &a);
    if (a > 10) //if
    {
        printf("a is greater than 10");
    }
    if (a < 10) //if
    {
        printf("a is less than 10");
    }
}

```

```

if (a == 10)
{
    printf(" a is 10");
}
getch();
}

```

- (33) W.A.P to accept age of a student & print whether student is eligible for driving license or not. Student is eligible for driving licence if age is greater than or equal to 18, otherwise not eligible.

```

#include <stdio.h>
#include <conio.h>
void main()
{
    int sage;
    clrscr();
    printf("Enter student age\n");
    scanf("%d", &sage);
    if (sage >= 18)
    {
        printf("Student is eligible for driving license");
    }
    if (sage <= 18)
    {
        printf("Student is not eligible for driving license");
    }
}

```

getch();

- Q4) W.A.P to accept average marks of student  
is pass or fail.

```
#include <stdio.h>
#include <conio.h>
void main()
{
    float avg;
    clrscr();
    printf("Enter the student marks");
    scanf("%f",&avg);
    if (avg > 40.00)
    {
        printf("The student is pass");
    }
    if (avg < 40.00)
    {
        printf("The student is fail");
    }
    getch();
}
```

(35) W.A.P to accept two integers from the user & print the largest of them.

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int num1, num2;
    clrscr();
    printf("Enter the first number : ");
    scanf("%d", &num1);

    printf("Enter the second number : ");
    scanf("%d", &num2);

    if (num1 > num2)
        printf("%d is greatest\n");
    else if (num2 > num1)
        printf("%d is greatest\n");
    else
        printf("Both numbers are equal\n");

    getch();
}
```

- (36) W.A.P. to accept total bill amount from the user & give 15% discount if total bill amount more than 1000 rupees.

```
#include <stdio.h>
#include <conio.h>
void main()
{
    float billamount;
    clrscr();
    printf("Enter bill amount: ");
    scanf("%f", &billamount);

    if (billamount < 1000)
        printf("Amount is less than 1000\n");
    else
        printf("Amount is 1000 or greater\n");

    getch();
}
```

(37)

W.A.P. to accept a number from the user & check whether it is positive or negative or zero.

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int num;
    clrscr();
    printf("Enter any number\n");
    scanf("%d", &num);
    if (num > 0)
        printf("Number is positive");
    if (num < 0)
        printf("Number is negative");
    if (num == 0)
        printf("Number is zero");
    getch();
}
```

(38) W.A.P. to accept a number from the user & check whether that number is EVEN or ODD.

```
#include < stdio.h >
#include < conio.h >
void main()
{
    int num;
    clrscr();
    printf("Enter any number\n");
    scanf("%d", &num);

    if (num % 2 == 0)
    {
        printf("Number is EVEN");
    }
    if (num % 2 == 1)
    {
        printf("Number is ODD");
    }
    getch();
}
```

Q39) W.A.P. to accept a number from the user & check whether it is multiple of 11 or not.

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int num;
    clrscr();
    printf("Enter any number\n");
    scanf("%d",&num);

    if (num % 11 == 0)
    {
        printf("Number is divisible by 11");
    }
    if (num % 11 != 0)
    {
        printf("Number is not divisible
               by 11");
    }
    getch();
}
```

Q) W.A.P to accept a four digit number from the user & print whether it is a palindrome number or not.

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int num, a, b, c, rev, temp;
    clrscr();
    printf("Enter a 4 digit number\n");
    scanf("%d", &num);
    temp = num;
    a = num % 10;
    num = num / 10;
    b = num % 10;
    num = num / 10;
    c = num % 10;
    num = num / 10;
    rev = (a * 1000) + (b * 100) + (c * 10) + num;
    if (temp == rev)
    {
        printf("Number is Palindrome");
    }
}
```

```

if (temp != rev)
{
    printf("Number is not palindrome");
}
getch();
}

```

- (41) W.A.P to accept 3 numbers from the user & print largest of them.

```

#include <stdio.h>
#include <conio.h>
void main()
{
    int n1, n2, n3;
    printf("Enter 3 numbers\n");
    scanf("%d %d %d", &n1, &n2, &n3);

    if (n1 > n2)
    {
        if (n1 > n3)
            printf("%d is largest.", n1);
        else
            printf("%d is largest.", n3);
    }
    else
    {
        if (n2 > n1)
            if (n2 > n3)
                printf("%d is largest.", n2);
            else
                printf("%d is largest.", n3);
    }
}

```

```

        printf(" %d is largest ", n2);
}
if (n3 > n1)
{
    if (n3 > n2)
    {
        printf(" %d is largest ", n3);
    }
}
getch();
}

```

- (42) W.A.P to accept to find a largest 4 number from the user & print largest of them.

```

#include < stdio.h >
#include < conio.h >
void main()
{
    int a, b, c, d;
    printf(" Enter 4 integers \n ");
    scanf(" %d %d %d %d ", &a, &b, &c, &d);
    if (a > b)
    {
        if (a > c)
        {
            if (a > d)

```

printf("%d is largest number\n", a);

if (b > a)

if (b > c)

if (b > d)

printf("%d is largest number\n", b);

if (c > a)

if (c > b)

if (c > d) // b, c, d, a for

{ if (operator is not >) then,

printf("%d is largest number\n",  
c);

}

(d < 0) . i;

```

if (d>a)
{
    if (d>b)
    {
        if (d>c)
        {
            printf("d is largest number in a, b, c");
        }
    }
    getch();
}

```

- (43) W.A.P to accept a three digit number from the user & check whether from the user & it is an Armstrong number or not.

```

#include <stdio.h>
#include <conio.h>
void main()
{
    int a, b, c, d, n;
    clrscr();
    printf("Enter a three digit number");
    scanf("%d", &d);
    a = (d % 10);
    b = (d / 10) % 10;
    c = (d / 100) % 10;
    n = (a * a * a) + (b * b * b) + (c * c * c);
    if (d == n);
}

```

```

    {
        printf("It is an Armstrong number");
    }
    getch();
}

```

- (44) W.A.P to accept 3 angles of a triangle & print whether the triangle is valid or not.  
A triangle is valid if sum of three angles is 180.

```

#include <stdio.h>
#include <conio.h>
void main()
{
    int a, b, c;
    clrscr();
    printf("Enter sides of a triangle\n");
    scanf("%d %d %d", &a, &b, &c);
    if (a+b+c == 180)
        printf("Triangle is valid");
    else
        printf("Triangle is not valid");
    getch();
}

```

(45) W.A.P to accept marks of 5 subjects of a student & print total count of numbers of subjects in which student is pass. To pass a subject marks must be  $\geq 40$ .

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int s1, s2, s3, s4, s5, a, b, c, d, e, sum;
    clrscr();
    printf("Enter 5 subjects marks\n");
    scanf("%d %d %d %d %d", &s1, &s2,
          &s3, &s4, &s5);
```

```
a=0;
b=0;
c=0;
d=0;
e=0;
```

```
a=1;
```

```
if (s2 >= 40)
{
    b=1;
}
```

```
if (CS3 >= 40)
```

```
{
```

```
c = 1;
```

```
}
```

```
if (CS4 >= 40)
```

```
{
```

```
d = 1;
```

```
}
```

```
if (CS5 >= 40)
```

```
{
```

```
e = 1;
```

```
}
```

```
sum = a + b + c + d + e;
```

```
printf("Student Pass in %d subjects", sum);
```

```
getch();
```

```
}
```

- Q46 W.A.P to accept a character from the user & print Capital "YES" if that character is an upper case Alphabet otherwise print "NO".

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

```
#include <conio.h>
```

```
void main()
```

```
{
```

```
char ch;
```

```
int flag = 25;
```

```

clrscr();
printf("Enter any character: ");
scanf("%c", &ch);

if (ch >= 'A') {
    if (ch >= 'G')
        printf("YES");
    flag = 900;
}
else if (ch <= 'Z' & ch >= 'A')
    if (flag == 25)
        printf("NO");
getch();
}

```

- (Q7) W.A.P. to accept any day of current year & print day number on the day.

```

#include <stdio.h>
#include <conio.h>
void main()
{
    int day, month;
    clrscr();
    printf("Enter day ");

```

```
scanf("%d", &day);
printf("Enter month number\n");
scanf("%d", &month);
if (m==1)
{
    printf("Total days = %d", d=d+31);
}
if (m==2)
{
    printf("Total days = %d", d=d+59);
}
if (m==3)
{
    printf("Total days = %d", d=d+90);
}
if (m==4)
{
    printf("Total days = %d", d=d+120);
}
if (m==5)
{
    printf("Total days = %d", d=d+151);
}
if (m==6)
{
    printf("Total days = %d", d=d+181);
}
if (m==7)
{
    printf("Total days = %d", d=d+212);
```

```
if (m == 8)
{
    printf("Total days = %d ", d = d + 212);
}
if (m == 9)
{
    printf("Total days = %d ", d = d + 242);
}
if (m == 10)
{
    printf("Total days = %d ", d = d + 273);
}
if (m == 11)
{
    printf("Total days = %d ", d = d + 303);
}
if (m == 12)
{
    printf("Total days = %d ", d = d + 334);
}
getch();
```

- ④ If 1<sup>st</sup> January 2023 was Sunday. Then  
W.A.P to accept any day of year 2023  
& print the weak day on that day.

15-06-11  
Saw a small bird which was very similar to "Mallard".  
I think it was a female "Mallard".

15-06-11  
Saw a small bird which was very similar to "Mallard".

15-06-11  
Saw a small bird which was very similar to "Mallard".

15-06-11  
Saw a small bird which was very similar to "Mallard".

15-06-11  
Saw a small bird which was very similar to "Mallard".

15-06-11  
Saw a small bird which was very similar to "Mallard".

15-06-11  
Saw a small bird which was very similar to "Mallard".

15-06-11  
Saw a small bird which was very similar to "Mallard".

15-06-11  
Saw a small bird which was very similar to "Mallard".

15-06-11  
Saw a small bird which was very similar to "Mallard".

15-06-11  
Saw a small bird which was very similar to "Mallard".

15-06-11  
Saw a small bird which was very similar to "Mallard".

15-06-11  
Saw a small bird which was very similar to "Mallard".

15-06-11  
Saw a small bird which was very similar to "Mallard".

15-06-11  
Saw a small bird which was very similar to "Mallard".

15-06-11  
Saw a small bird which was very similar to "Mallard".

15-06-11  
Saw a small bird which was very similar to "Mallard".

15-06-11  
Saw a small bird which was very similar to "Mallard".

15-06-11  
Saw a small bird which was very similar to "Mallard".

15-06-11  
Saw a small bird which was very similar to "Mallard".

15-06-11  
Saw a small bird which was very similar to "Mallard".

15-06-11  
Saw a small bird which was very similar to "Mallard".

15-06-11  
Saw a small bird which was very similar to "Mallard".

15-06-11  
Saw a small bird which was very similar to "Mallard".

15-06-11  
Saw a small bird which was very similar to "Mallard".

15-06-11  
Saw a small bird which was very similar to "Mallard".

15-06-11  
Saw a small bird which was very similar to "Mallard".

④ W.A.P to accept a character from the user & print whether it is an uppercase & lowercase letter or digit or sp. symbol.

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int flag = 1;
    char ch;
    clrscr();
    printf("Enter any character\n");
    scanf("%c", &ch);

    if (ch >= 65)
    {
        if (ch <= 90)
            printf("It is an Uppercase
letter");
        flag = 2;
    }
    if (ch >= 97)
    {
        if (ch <= 122)
            printf("It is an lowercase
letter");
        flag = 2;
    }
}
```

```

if (ch >= 48) { // part of global (0)
{
    if (ch <= 57)
    {
        printf("It is a digit");
        flag = 1; // local
    }
}

if (flag == 1):
{
    printf("It is a special symbol");
}

getch();
}

```

### \* Logical operator :-

- are to check multiple conditions.

Operator	Description
&& (AND)	Results TRUE if both operand conditions are TRUE ; otherwise results FALSE.
(OR)	Results TRUE if any one Operand condition is TRUE ; otherwise results FALSE.
! (NOT)	is to invert/reverse result of any condition.

(50)

W.A.P to find smallest of 3 number  
Use logical operator.

```
#include <stdio.h>
#include <conio.h>
void main()
{
    clrscr();
    int a, b, c; // = - 2017.0
    printf("Enter any value\n");
    scanf("%d%d%d", &a, &b, &c);
    if (a > b & a > c)
    {
        printf("A is largest.");
    }
    if (c > b) if (c > a)
    {
        printf("B is largest.");
    }
    if (c > a & c > b)
        printf("C is largest.");
}
```

- (51) A library charged fine for every book written late. W.A.P. to accept numbers of days a student's is late to return the book & print find the following table.

No. of days	Fine in Rs.
1 - 7	1 RS./ day
8 - 19	2 RS./ day
20 - 29	3 RS./ day
28 - 30	5 RS./ day
> 30	Membership is cancelled

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int days, fine;
    clrscr();
    printf("Enter number of days\n");
    scanf("%d", &days);
    if (days >= 1 && days <= 7)
    {
        fine = days * 1;
        printf("fine is Rs. %d", fine);
    }
    if (days >= 8 && days <= 19)
    {
        fine = days * 2;
        printf("fine is Rs. %d", fine);
    }
}
```

```
if (days >= 20 & days <= 24)
```

```
{  
    fine = days * 3;  
    printf("fine is Rs. %d ", fine);
```

```
if (days >= 25 & days <= 30)
```

```
{  
    fine = days * 5;  
    printf("fine is Rs. %d ", fine);
```

```
if (days >= 31)
```

```
{  
    fine = days * Membership is cancelled;  
    printf("fine is Rs. %d ", fine);
```

```
getch();
```

- (52) WAP to accept number of units consumed by a consumer of an electricity meter. Calculate & print total bill amount from consumed units. Refer following table.

Units consumed	charges per unit
1 - 150	2.50 / unit
151 - 300	2.85 / unit
301 - 500	3.15 / unit
> 500	3.45 / unit

Perform cumulative / slab wise calculation of bill. Also add 150 Rs./- in final bill as rent of electricity meter.

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

```
#include <conio.h>
```

```
void main()
{
```

```
    float n, tb;
```

```
    clrscr();
```

```
    printf("Enter total units consumed (n)");
```

```
    scanf("%f", &n);
```

```
    if (n >= 1 && n <= 150)
```

```
    {
```

```
        tb = n * 2.5;
```

```
    if (n >= 151 && n <= 300)
```

```
        tb = 375 + (n - 150) * 2.85;
```

```
    if (n >= 301 && n <= 500)
```

```
        tb = 375 + 427.5 + (n - 300) * 3.75;
```

```
(375 + 427.5 + 630 + (n - 500) * 3.75);
```

```
if (n > 500)
```

```
tb = 375 + 427.5 + 630 + (n - 500) * 3.45;
```

```
printf("Total bill amount = %d", tb);
```

```
getch();
```

(53)

W.A.P to accept 3 sides of triangle & print whether it is an equilateral triangle or isosceles triangle or scalar triangle or a right angled triangle.

```

#include <stdio.h>
#include <conio.h>
void main()
{
    float s1, s2, s3;
    clrscr();
    printf("Enter three sides of triangle : ");
    scanf("%f %f %f", &s1, &s2, &s3);

    if (s1 == s2 && s2 == s3)
    {
        printf("It is a Equilateral triangle\n");
    }
    if (s1 == s2 && s2 == s3 || s2 == s3)
    {
        printf("It is a Isosceles triangle\n");
    }
    if (s1 * s1 == s2 * s2 + s3 * s3 || s2 * s2
        == s1 * s1 + s3 * s3 || s3 * s3 == s1 * s1 + s2 * s2)
    {
        printf("It is a Right angled triangle\n");
    }
    else
    {
        printf("It is a Scalene triangle\n");
    }
    getch();
}

```

- (54) W.A.P. to accept fees amount & gender letter of a student & give 200 Rs. discount in fees , if gender is male , otherwise give 150 Rs discount.

```
#include <stdio.h>
#include <conio.h>
void main()
{
    clrscr();
    int d;
    char g;
    printf("Enter gender of student in ");
    scanf("%c", &g);
    if (g == 'M' || g == 'm')
        printf("Enter fees of student in ");
    scanf("%d", &d);

    if (g == 'M' || g == 'm')
    {
        d = d - 200;
        printf("Total fees amount with
               discount is %d Rs ", d);
    }
    else
        d = d - 150;
    printf("Total fees amount with
           discount is %d Rs ", d);
}
getch();
```

(55) W.A.P to accept month number from the user & print the number of days that month has.

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int month;
    clrscr();
    printf("Enter the month number\n");
    scanf("%d", &month);
    if (month == 1 || month == 3 || month == 5
        || month == 7 || month == 8 || month ==
        10 || month == 12)
    {
        printf("This month have 31 days");
    }
    if (month == 2)
        printf("This month have 28 & 29 days");
    if (month == 4 || month == 6 || month == 9
        || month == 11)
        printf("This month have 30 days");
    getch();
```

(56)

A company decides salary of employee from the following table.

Age	Gender	Work experience	Salary
19-23	M	0-3	125,000
19-23	F	0-3	28,000
24-30	M	4-10	40,000
24-30	F	4-10	45,000
31-37	M	11-19	55,000
31-37	F	11-19	55,000
>37	M/F	>19	75,000
>45	M/F	>25	90,000

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

```
#include <conio.h>
```

```
void main()
```

```
{
```

```
int a, w-e? salary;
```

```
char G;
```

```
clrscr();
```

```
printf("Enter gender :");
```

```
scanf("%c", &G);
```

```
printf("Enter age :");
```

```
scanf("%d", &a);
```

```
printf("Work experience :");
```

```
scanf("%d", &w-e);
```

```
if (a >= 19 && a <= 23 && G == 'M') &&  
w-e >= 0 && w-e <= 3)
```

```
{
```

```
printf("Salary is : 25000");
```

```
}
```

```
if (a >= 19 && a <= 23 && G == 'F' && w_e  
    >= 0 && w_e <= 3) { program A (2)  
    printf("Salary is : 28000");  
}
```

```
if (a >= 24 && a <= 30 && G == 'M' && w_e  
    >= 0 && w_e <= 10) {
```

```
    printf("Salary is : 40000");
```

```
if (a >= 24 && a <= 30 && G == 'F' && w_e >= 6  
    && w_e <= 10) {
```

```
    printf("Salary is : 45000");
```

```
if (a >= 31 && a <= 37 && w_e >= 11 &&  
    w_e <= 19) { min max  
    printf("Salary is : 55000");
```

```
if (a > 37 && w_e > 19 && a <= 45) {
```

```
    if (G == 'M') { G == 'F' } else
```

```
    printf("Salary is : 75000");
```

```
if (a < 45 && w_e > 25) {
```

```
    if (G == 'M') { G == 'F' } else
```

```
    printf("Salary is : 90000");
```

```
, getch();
```

(57) A steel manufacturing company decides grade of steel from following conditions.

- 1) Carbon content must be less than 0.8
- 2) Hardness must be greater than 11.2
- 3) Tensile strength must be greater than 14.6

This company develops one batch of a steel lock. W.A.P to accept value of carbon content, hardness & tensile strength of that batch decide grade of lock from following criteria.

Grade is A if all 3 conditions are met.

Grade is B if condition 1 & 2 are met.

Grade is C if condition 2 & 3 are met.

Grade is D if condition 1 & 3 are met.

Grade is E if any 1 condition is met.

Grade is F if no conditions are met.

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

```
((5.5)) #include <conio.h>
```

```
void main()
```

```
:((5.7)) {
```

```
int c, h, t;
```

```
clrscr();
```

```
printf("Enter the carbon amount\n");
```

```
scanf("%d", &c);
```

```
printf("Enter the hardness\n");
```

```
scanf("%d", &h);
```

```
printf("Enter the tensile strength\n");
```

```
scanf("%d", &t);
```

```
if ((c < 0.8) && (h > 11.2) && (t > 14.6))
```

```
{ printf("Grade of material is A\n"); }
```

```
if ((c < 0.8) && (h > 11.2) && (t > 14.6))
```

```
{ printf("Grade of material is B\n"); }
```

```
if ((c > 0.8) && (h > 11.2) && (t > 14.6))
```

```
{ printf("Grade of material is C\n"); }
```

```
if ((c < 0.8) && (h < 11.2) && (t < 14.6))
```

```
{
```

```
printf("Grade of material is D\n"); }
```

```
if ((c < 0.8) && (h < 11.2) && (t < 14.6))
```

```
((c > 0.8) && (h > 11.2) && (t > 14.6))
```

```
((c > 0.8) && (h > 11.2) && (t < 14.6))
```

```
{ printf("Grade of material is E\n"); }
```

```
if ((c > 0.8) && (h < 11.2) && (t < 14.6))
```

```
{
```

```
printf("Grade of material is F\n"); }
```

```
getch();
```

```
}
```

- (58) W.A.P to accept 3 sides of triangle from the user & print whether the triangle is equilateral or not. (conditional if)

```
#include < stdio.h >
#include < conio.h >
void main()
{
    int s1, s2, s3;
    clrscr();
    printf("Enter 3 sides of triangle : \n");
    scanf("%d %d %d", &s1, &s2, &s3);

    if (s1 == s2) && (s2 == s3) && (s3 == s1)
    {
        printf("Triangle is Equilateral \n");
    }
    else
    {
        printf("Triangle is not equilateral \n");
    }
    getch();
}
```

b) If...else Statement: A program of Q.3a

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

\* Here 'if' & 'else' are key words.

\* If specified condition is True, then if block executes.

\* If specified condition is false, then else block executes.

- (59) W.A.P. to accept three sides of triangle & check whether it is equilateral triangle or not.

Output is shown in

```
#include < stdio.h> // about it
```

```
#include < conio.h> (Driver file)
```

```
void main()
```

```
{
```

```
int s1, s2, s3;
```

```
printf("Enter 3 sides of triangle");
```

```
scanf("%d%d%d", &s1, &s2, &s3);
```

```
if (s1 == s2) && (s2 == s3) && (s3 == s1)
```

```
{
```

```
    printf("Triangle is Equilateral\n");
```

```
}
```

```
else
```

```
{
```

```
    printf("Triangle is not equilateral");
```

```
}
```

```
getch();
```

```
}
```

(60)

W.A.P. to check whether enter a number is even or odd.

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int n;
    clrscr();
    printf("Enter a number\n");
    scanf("%d", &n);

    if (n % 2 == 0)
    {
        printf("Number is Even");
    }
    else
    {
        printf("Number is Odd");
    }
    getch();
}
```

(61) W.A.P. to accept age of student & print whether he is eligible for driving licence or not.

```
#include < stdio.h >
#include < conio.h >
void main()
{
    int age;
    clrscr();
    printf("Enter the student's age\n");
    scanf("%d", &age);
    if (age >= 18)
        printf("Student is Eligible for Driving
               licence");
    else
        printf("Student is not Eligible for
               Driving licence");
    getch();
}
```

(62) W.A.P to accept an alphabet from the user and print whether it is an uppercase letter & lowercase letter use if...else.

```
#include < stdio.h> // header file
```

```
#include < conio.h> // header file
```

```
void main() // driver function
```

```
{
```

```
int flag=1; // flag variable
```

```
char ch; // character variable
```

```
clrscr(); // clear screen
```

```
printf("Enter any character\n");
```

```
scanf("%c", &ch);
```

```
if (ch>=65 && ch<=90)
```

```
{ // condition satisfied
```

```
printf("It is an Uppercase letter:\n");
```

```
flag=2;
```

```
}
```

```
else
```

```
if (ch>=97 && ch<=122)
```

```
{ // condition satisfied
```

```
printf("It is an Lowercase letter:\n");
```

```
flag=2;
```

```
}
```

```
getch();
```

```
}
```

\* Logical possibilities of if...else. 97/9

① Simple if...else or single if...else:-

if (condition)

{

statements;

=

else

{

statements;

=

{

97/9

② Multiple if...else statements; - 97/9 -

if (condition-1)

{

statements;

=

{

else

{

statements;

=

{

if (condition-2)

{

statements;

=

{

else { } is for writing logic \*

Statement ;

}

### ③ Nested if...else statements

(A) if (condition-1) {

    if (condition-2) {

        statements ;

=

}

    else ;

{

        statements ;

=

}

    else

{

        statements ;

=

}

(B) if (condition-1) {

    statements ;

:

    else

{

    if (condition-2) {

{

        statements ;

=

:

        statements ;

=

:

        statements ;

=

:

    } if (condition-2) {

{

        statements ;

=

:

    } if (condition-2) {

{

- ⑥ W.A.P to accept the number from the user & print whether it is greater than 10, or less than or equal to 10 . Use only if.. else. Single if not allowed.

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int a;
    clrscr();
    printf("Enter any number\n");
    scanf("%d", &a);

    if(a>10)
    {
        printf("A is greater than 10");
    }
    else if(a<10)
    {
        printf("A is less than 10");
    }
    else if(a==10)
    {
        printf("A is zero");
    }
    getch();
}
```

- 64) WAP to accept a number from the user & print whether it is positive or negative or zero.

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int a;
    clrscr();
    printf("Enter any number\n");
    scanf("%d", &a);
    if (a>0)
        printf("Number is positive");
    else if (a<0)
        printf("Number is Negative");
    else if (a==0)
        printf("Number is zero");
    getch();
}
```

(65) W.A.P to accept two numbers and find largest of two numbers. Using if-else-if.

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

```
void main()
```

```
{
```

```
    int num1, num2; // input binary
```

```
    clrscr();
```

```
    printf("Enter 1 number\n");
```

```
    scanf("%d", &num1);
```

```
    printf("Enter 2 number\n");
```

```
    scanf("%d", &num2);
```

```
    if (num1 > num2)
```

```
    {
```

```
        printf("Num1 is largest\n");
```

```
    }
```

```
    else if (num1 < num2)
```

```
    {
```

```
        printf("Num 2 is largest\n");
```

```
    }
```

```
    else
```

```
    {
```

```
        printf("Both numbers are Equal ");
```

```
    getch();
```

```
}
```

Q66 W.A.P to accept 3 numbers & find largest of three  
number. Using if...else... .

- Without using logical operators.
- By using logical operators.

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int num1, num2, num3;
    clrscr();
    printf("Enter 3 numbers");
    scanf("%d %d %d", &num1, &num2, &num3);

    if (num1 > num2 && num1 > num3 &&
        num3 > num1)
    {
        printf("%d is the largest number", num1);
    }
    if (num2 > num1 && num2 > num3)
    {
        printf("%d is the largest number", num2);
    }
    if (num3 > num1 && num3 > num2)
    {
        printf("%d is the largest number", num3);
    }
    getch();
}
```

b.

```

#include <stdio.h>
#include <conio.h>
void main()
{
    int num1, num2, num3;
    clrscr();
    printf("Enter 3 numbers\n");
    scanf("%d%d%d", &num1, &num2, &num3);

    if (num1 > num2)
    {
        if (num1 > num3)
            printf("num1 is largest");
        else
            printf("num3 is largest");
    }
    else
    {
        if (num2 > num3)
            printf("num2 is largest");
        else
            printf("num3 is largest");
    }
    getch();
}

```

(67)

W.A.P to find largest of four numbers.  
Using if-else.

- a. by using logical operators
- b. without using logical operators.

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int n1, n2, n3, n4;
    clrscr();
    printf("Enter 4 numbers\n");
    scanf("%d %d %d %d", &n1, &n2, &n3,
        &n4);
    if (n1 > n2 && n1 > n3 && n1 > n4)
    {
        printf("n1 is largest");
    }
    else
    {
        if (n2 > n3 && n2 > n4)
        {
            printf("n2 is largest");
        }
        else
        {
            if (n3 > n4)
            {
                printf("n3 is largest");
            }
            else
            {
                printf("n4 is largest");
            }
        }
    }
}
```

```

    {
        printf("%d is largest");
    }
}

```

```
getch();
```

b. without using logical operator :-

```

#include < stdio.h >
#include < conio.h >
void main()
{

```

```
    int n1, n2, n3, n4;
    clrscr();

```

```
    printf("Enter 4 numbers\n");

```

```
    scanf("%d %d %d %d", &n1, &n2, &n3,
          &n4);

```

```
    if (n1 > n2) && n2 > n3 && n3 > n4 && n1 > n4)

```

```
        printf("%d is largest", n1);
    }

```

```
    if (n2 > n1 && n2 > n3 && n3 > n4)

```

```
        printf("%d is largest", n2);
    }

```

```
    if (n3 > n1 && n3 > n2 && n3 > n4)

```

```
        printf("%d is largest", n3);
    }
}
```

```

if(n1>n4 & n2>n4 & n3>n4)
{
    printf("%d is largest", n4);
}
getch();
}

```

- (68) WAP to find largest of five numbers using if...else. (without using logical operator.)

```

#include <stdio.h>
#include <conio.h>
void main()
{
    int n1, n2, n3, n4, n5;
    clrscr();
    printf("Enter 5 numbers\n");
    scanf("%d%d%d%d%d", &n1, &n2, &n3,
          &n4, &n5);
    if(n1>n2 & n1>n3 & n1>n4 & n1>n5)
    {
        printf("n1 is largest\n");
    }
    else
    {
        if(n2>n1 & n2>n3 & n2>n4 & n2>n5)
        {
            printf("n2 is largest\n");
        }
        else
        {
            printf("n3 is largest\n");
        }
    }
}

```

```

if(n3>n1 && n3>n2 && n3>n4 && n3>n5)
{
    printf("n3 is largest\n");
}
else
if(n4>n1 && n4>n2 && n4>n3 && n4>n5)
{
    printf("n4 is largest\n");
}
else
{
    printf("n5 is largest\n");
}
getch();
}

```

else.. if statement or else.. if ladder:-

In case of multiple if-statements all if-statements are independent.

It checks/ tests all conditions.

∴ It wastes execution time.

**Ladder = multiple**

if condition-1)

{

statements;

=

}

else if condition-2)

{

statements;

=

}

else if condition-3)

{

statements;

=

}

else if condition-n)

{

statements;

=

}

else

{

statements;

=

}

statements;

=

- \* In else-if statements, if any one condition appears TRUE then all remaining else-if blocks gets skipped.
  - \* Last else-block executes when no conditions gets satisfied.
  - \* we can use this last else-block to do any default operation.
  - \* Last else-block is optional.
- (Q) W.A.P to accept the numbers from the user & print whether it is positive or negative or zero. Using else-if.

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int num;
    clrscr();
    printf("Enter any number\n");
    scanf("%d", &num);
    if (num > 0)
        printf("It is positive\n");
    else if (num < 0)
        printf("It is negative\n");
}
```

```

else
{
    printf("It is zero"); getch();
}
}

```

- (70) W.A.P to accept age of the person & gender of the person & print whether he is eligible for driving licence or not. Refer following table.

Age	Gender	Decision
$\geq 16$	M	eligible
$\geq 18$	F	eligible
In all other cases		not eligible

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

```
#include <conio.h>
```

```
void main()
```

```
int age; char gen;
```

```
clrscr();
```

```
printf("Enter gen of the applicant\n");
```

```
:scanf("%c", &gen);
```

```
printf("Enter age of the applicant\n");
```

```
:scanf("%d", &age);
```

```
if (age  $\geq 16$  & gen == 'M')?
```

```
{ printf("Eligible for Driving
```

```
printf("Eligible for Driving licence");}
```

Page No.	
Date	

```

else
if (age >= 18 && gen == 'F')
{
    printf("Eligible for driving licence");
}
else
{
    printf("Not eligible for driving licence");
}
getch();
}

```

- Q7) A company decides efficiency of employee based on the numbers of days taken by an employee to complete the task. W.A.P. to accept numbers of days taken by an employee to complete the task. & print proper message from following table.

No. of days	Message
1-9	Excellent performance
10-12	Acceptable performance
13-15	Needs training
16-17	Needs to improve & training
>17	Warning & training at local or remote centre.

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

```
#include <conio.h>
```

```
void main()
```

```
{ clrscr();
```

```
int days;
```

```
clrscr();
```

```
printf("Enter the days taken by
```

```
employee\n");
```

```
scanf("%d", &days);
```

```
if (days >= 1 && days <= 9)
```

```
{ printf("Excellent performance"); }
```

```
else if (days >= 10 && days <= 12)
```

```
{ printf("Acceptable performance"); }
```

```
else if (days >= 13 && days <= 15)
```

```
{ printf("Needs training"); }
```

```
else if (days >= 16 && days <= 17)
```

```
{ printf("Needs training at local centre"); }
```

```
else
```

if (days >= 17) {  
 {

    printf("Learning & training at remote  
 centre");

} // if (days >= 17) instead of (days >= 17)  
 getch();

}

- 72) An insurance company decides policy approval & premium payment amount from following table.

Age (M/F)	gen (C/U)	lives-in (H/S)	health (C/L)	status (S/E/P)	maximum policy amount	premium for month
25- M	C	S	C	E	2,00,000	156 RS/
35	C	S	C	S	1,50,000	1000 RS
25- F	C	S	C	E	1,75,000	84 RS/
35	C	S	C	S	1,40,000	1000 RS
25- M	C	S	C	E	1,40,000	51 RS/
40	C	S	C	S	1,00,000	1000 RS

In all other cases application is not eligible for policy.

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

```
#include < conio.h > // header
```

```
void main()
```

```
{
```

```
    long amt;
```

```
    int pre, age;
    char gen, lives, hs;
```

```
class();

```

```
printf("Enter age of applicant\n"); if  
scanf("%d", &age);
```

```
fflush(stdin);
```

```
printf("Enter gender of applicant\n");  
scanf("%c", &gen);
```

```
fflush(stdin);
```

```
printf("Enter lives-in\n");  
scanf("%c", &lives);
```

```
fflush(stdin);
```

```
printf("Enter Health state\n");  
scanf("%c", &hs);
```

```
if (age >= 28 && age <= 35) && gen == 'M' &&  
lives == 'C' && hs == 'E'
```

```
{ printf("Applicant is eligible for policy\n");  
printf("Enter requested policy amount\n");  
scanf("%d", &amt);
```

```
if (amt <= 2,00,000) {
```

```
pre = (amt * 56) / 1000;
```

```
printf("Premium amount from requested  
policy amount : %d\n", pre);
```

```
else
```

```
{ printf("Requested policy amount exceeds\n"); }
```

else

if  $\text{age} \geq 25 \& \& \text{age} \leq 35 \& \& \text{gen} == 'F' \& \&$   
 $\text{lives} == 'M' \& \& \text{hs} == 'E'$

{

printf("The Applicant is eligible for policy\n");

printf("Enter requested policy amount\n");

scanf("%ld", &amt);

if ( $\text{amt} \leq 1,75,000$ )

{

pre =  $(\text{amt} * 54) / 1000$ ;

printf("Premium amount from requested policy  
amount : %.2f", pre);

else

{

printf("Requested policy amount exceeds ");

}

else

if  $\text{age} \geq 25 \& \& \text{age} \leq 40 \& \& \text{gen} == 'M' \& \&$   
 $\text{lives} == 'U' \& \& \text{hs} == 'E'$

{

printf("The Applicant is eligible for policy\n");

printf("Enter requested policy amount\n");

scanf("%ld", &amt);

if ( $\text{amt} \leq 1,40,000$ )

{ if ( $\text{amt} \geq 1,40,000$ )

pre =  $(\text{amt} * 51) / 1000$ ;

printf("Premium amount from requested  
policy amount : %.2f", pre);

}

```

else
{
    printf("Requested policy amount exceeds");
}
else if (eligible)
{
    printf("Applicant is not Eligible for policy");
}
getch();
}

```

- 73) W.A.P to accept cost price & selling price of a product & print whether the shopkeeper has made profit or loss. Also print the amount he earned or loss.

```

#include <stdio.h>
#include <conio.h>
void main()
{
    float cp, sp, diff;
    clrscr();
    printf("Enter the cost price\n");
    scanf("%f", &cp);
    fflush(stdin);
    printf("Enter the selling price\n");
    scanf("%f", &sp);
}

```

74)

current  
>1  
10001 -  
8000 -  
In a

```
if (cp > sp)
{
```

$$\text{diff} = cp - sp;$$

```
printf("The shopkeeper has made loss of Rs. %f\n", diff);
```

```
}
```

```
else if (sp > cp)
```

```
{
```

$$\text{diff} = sp - cp;$$

```
printf("The shopkeeper has made profit of Rs. %f\n", diff);
```

```
}
```

```
else if (cp == sp)
```

```
{
```

```
printf("No profit or No loss");
```

```
}
```

```
getch();
```

```
}
```

- 74) A shopkeeper demands for goods to a company out late. The company in out late decide whether to deliver the goods for not from following table.

current purchase order	previous balance amount	relation- ship years	Decision
>18000	<2000	>10	Deliver
10001 - 18000	<1200	>6 & <10	-11-
8000 - 10000	<700	>3 & <6	-11-
In all other cases			does not -11-

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int amt, pamt, ryear;
    clrscr();
    printf("Enter Amount : \n");
    scanf("%d", &amt);
    printf("Enter previous amount : \n");
    scanf("%d", &pamt);
    printf("Enter relationship years : \n");
    scanf("%d", &ryear);

    if (amt > 18000 && pamt < 2000 && ryear > 10)
        printf("Deliver\n");
    else if (amt < 10000 && amt > 18000 && pamt < 1200 && ryear > 6)
        printf("Deliver\n");
    else if (pamt < 8000 && amt > 10000 && pamt < 700 && ryear > 3)
        printf("Deliver\n");
    else
        printf("Does not deliver ");
    getch();
}
```

## \* Conditional Operator:-

`condition ? statement-1 : statement-2 ;`

This operator is also called as ternary operator. This operator is shortcut of if-else statement.

To use this operator general syntax is

onward upward in box.

- \* If specified condition is TRUE, then statement -1 executes.
- \* If -specified condition is FALSE, then statement -2 executes.
- \* Here, only single statement is allowed as TRUE part & FALSE part.

(+) W.A.P. to accept the number from the user & print whether it is even or odd. Use conditional operator!

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int num;
    clrscr();
    printf("Enter any number: \n");
    scanf("%d", &num);
    (num % 2 == 0) ? printf("EVEN") : printf("ODD");
    getch();
}
```

76 W.A.P to accept two numbers & print largest of them (using conditional operator)

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int a, b, lar;
    clrscr();
    printf("Enter 2 numbers\n");
    scanf("%d%d", &a, &b);
    (a>b) ? (lar = a) : (lar = b);
    printf("Largest is %d\n", lar);
    getch();
}
```

77 W.A.P to accept the check whether entered year is a leap year or not. A year is leap year  
 ① if it is divisible by 4  
 ② if it is divisible by 100 but not divisible by 400. In all other cases not leap years.

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int year;
    clrscr();
    printf("Enter the year\n");
    scanf("%d", &year);
```

```

if ((year % 4 == 0 & year % 100 != 0) ||  

    (year % 400 == 0))  

{  

    printf("%d is leap year\n", year);  

}  

else {  

    printf("%d is not leap year\n", year);  

}  

getch();

```

\* Largest of two (use conditional operator):-

```

#include < stdio.h>  

#include < conio.h>  

void main()  

{  

    int a, b, c;  

    scanf(&a, &b);  

    c = a > b ? a : b;
}

```

OR

```

c = a > b ? a : b;

```

```

printf("%d is largest\n", c);

```

```

getch();

```

condition

True

- (A) condition-1 ? condition-2 ? statement-1 :  
 statement-2 ? statement-3 ;  
 False

- (B) condition-2 ? statement-1 : condition-2 ?  
 statement-2 : statement-3 ;  
 F

- (C) condition-1 ? condition-2 ? statement-1 :  
 C  
 statement-2 : condition-3 ? statement-3 :  
 T  
 Statement-4 ;

- Q8 W.A.P to check whether entered number is positive, negative or zero. Use conditional operator.

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

```
void main()
{
```

```
    int number;
```

```
    clrscr();
```

```
    printf("Enter a number : \n");
```

```
    scanf("%d", &number);
```

```
    number > 0 ? printf("is positive\n") : number < 0  

    ? printf("is negative\n") : printf("is zero\n");
```

```
    getch();
```

```
}
```

- (Q) Write a P to obtain the largest of three numbers using conditional operator among 3 lines  
 (1) by using logical operators  
 (2) without using logical operators.

(1) #include <stdio.h>  
 #include <conio.h>  
 void main()

```
int num1, num2, num3, largest;
clrscr();
printf("Enter 3 numbers : \n");
scanf("%d %d %d", &num1, &num2, &num3);

```

```
larg = (num1 > num2 && num1 > num3) ? num1 : (num2 > num3) ? num2 : num3;
```

```
printf("%d is largest", larg);
getch();
```

```
getch();
```

(2)

#include <stdio.h>

#include <conio.h>

void main() : { clrscr();

int num1, num2, num3, largest;

clrscr();

printf("Enter 3 numbers : \n");

```
scanf("%d %d %d", &num1, &num2, &num3);
```

```

largest = (num1 > num2) ? ((num1 > num3) ?
    num1 : num3) : ((num2 > num3) ? num2 :
    num3);
printf("%d is largest\n", largest);
getch();
}

```

(8) W.A.P to obtain the largest of four numbers using conditional operator.

- ① by using logical operator
- ② without using logical operator

```

① #include <stdio.h>
#include <conio.h> source: main() = m1
void main()
{
    int num1, num2, num3, num4, largest;
    clrscr();
    printf("Enter 4 numbers: \n");
    scanf("%d %d %d %d", &num1, &num2, &num3, &num4);

    largest = (num1 > num2 && num1 > num3 &&
    num1 > num4) ? num1 : (num2 > num3 &&
    num2 > num4) ? num2 : (num3 > num4) ?
    num3 : num4;
}
```

```

printf("%d is largest", largest);
getch();
}

```

② #include <stdio.h>  
#include <conio.h>  
void main()  
{  
 int num1, num2, num3, num4, largest;  
 clrscr();  
 printf("Enter 4 numbers : ");  
 scanf("%d %d %d %d", &num1, &num2, &num3, &num4);  
 largest = (num1 > num2) ? ((num1 > num3) ?  
 ((num1 > num4) ? num1 : num4) : ((num3 >  
 num4) ? num3 : num4)) : ((num2 > num3) ?  
 ((num2 > num4) ? num2 : num4) : ((num3 >  
 num4) ? num3 : num4));  
 printf("%d is largest ", largest);  
 getch();  
}

$i \rightarrow$  loop counter

i stands for loop counter

\* Loops or Iterative statements or

Repetitive statements:-

Page No.

Date

Every program that we execute till date, executes from top to bottom. Additionally, we placed with decision making statements to keep the execution based on condition.

But if a block of statement needs to execute iteratively/repetitively then we have to use loops.

A loop is block of statement that iteratively executes certain number of times/until a condition becomes false/until a flag changes.

\* C language provide 3 loops:

1) for loop

2) while loop

3) do-while loop

1) for loop:-

Practically, all three loops iteratively performed same operation & therefore a program implemented in one loop can be also implemented use in other to. but professionally all three loops have specific purpose of implementation.

① Prefer for loop when numbers of iterations are all ready known.

② Prefer while loop when numbers of iterations are not known but it depends on any internal situation/value.

③ Prefer do-while loop when at least 1 iteration is required specially for user-driven program.

\* General Syntax :-

for (initialization; condition; incr/decr)

    {

        statements;

        —

        —

        —

}

Q81) W.A.P to print "Hello all" message 10 times.

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

```
#include <conio.h>
```

```
void main()
```

```
{
```

```
    int i;
```

```
    clrscr();
```

```
    for (i=1; i<=10; i=i+1)
```

```
{
```

```
        printf("Hello all\n");
```

```
}
```

```
    getch();
```

```
}
```

①

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int i;
    clrscr();
    for (i=1; i<=20; i=i+2)
    {
        printf("Hello all\n");
    }
    getch();
}
```

②

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int i;
    clrscr();
    for (i=1; i<=30; i=i+3)
    {
        printf("Hello all\n");
    }
    getch();
}
```

③ #include < stdio.h>  
#include < conio.h>  
void main ()  
{  
 int i;  
 clrscr();  
 for (i=1; i<=40; i=i+4)  
 {  
 printf ("Hello all\n");  
 }  
 getch();  
}

④ #include < stdio.h>  
#include < conio.h>  
void main ()  
{  
 int i;  
 clrscr();  
 for (i=78; i<=85; i=i+1)  
 {  
 printf ("Hello all\n");  
 }  
 getch();  
}

(5) #include < stdio.h >  
 #include < conio.h >  
 void main()  
 {  
 int i;  
 clrscr();  
 for (i = 26; i <= 45; i = i + 2)  
 {  
 printf("Hello %c\n");  
 }  
 getch();  
 }

\* void main()  
 {  
 int i;  
 clrscr();  
 printf("main starts\n");  
 for (i = 75; i <= 85; i++)  
 {  
 printf("%d\n", i);  
 }  
 printf("main ends");  
 getch();  
 }

(82) WAP - to print all the numbers from 1 to 10.

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int i;
    clrscr();
    for (i=1; i<=10; i++)
    {
        printf("%d\n", i);
    }
    getch();
}
```

or

```
for (i=11; i<=20; i++)
{
    a = i-10;
    printf("%d\n", a);
}
```

(i.e. 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20)

(i.e. 1, 2, 3, 4, 5, 6, 7, 8, 9, 10)

}

(i.e. 1, 2, 3, 4, 5, 6, 7, 8, 9, 10)

}

(83) WAP to print all the even numbers from 1 to 100.

```
#include < stdio.h >
#include < conio.h >
void main()
{
    int i;
    clrscr();
    for (i=2; i<=100; i+=2)
    {
        printf("%d\n", i);
    }
    getch();
}
```

or

```
for (i=2; i<=100; i++)
{
    printf("%d\n", i);
    i++;
}
```

or

```
for (i=1; i<=100; i++)
{
    if (i % 2 == 0)
    {
        printf("%d\n", i);
    }
}
```

(84) WAP to print the all number from 10 to 1.

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int i;
    clrscr();
    for (i=10; i>=1; i=i-1)
    {
        printf("%d\n", i);
    }
    getch();
}
```

(85) WAP all the numbers. from 75 to 150.

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int i;
    clrscr();
    for (i=75; i<=150; i=i+1)
    {
        printf ("%d\n", i);
    }
    getch();
}
```

- 86) WAP to print square & cube of all the numbers from 1 to 10. The output should be in following format. Use it.

number	square	cube
1	1	1
2	4	8
3	9	27
:	:	:
;	(i-1)*i*i*(i+1)	i*i*i
10	100	1000

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int i, number, square, cube;
    clrscr();
    printf("Number\t Square\t Cube\n");
    for (i=1; i<=10; i++)
    {
        square = i * i;
        cube = i * i * i;
        printf("%d\t %d\t %d\n", i, square, cube);
    }
    getch();
}
```

- (87) W.A.P to print all the numbers from 1 to n , where n is entered by the user.

(Output box)

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int i, n;
    clrscr();
    printf("Enter the value of n");
    scanf("%d", &n);
    for (i=1; i<=n; i++)
    {
        printf("%d\n", i);
    }
    getch();
}
```

- (89) W.A.P to print multiplication table of entered number . The table should be in following format  
Enter the number : 2

2

4

6

;

20

(Output box)

$i = i + 2$

```

#include <stdio.h> // Header file of stdio
#include <conio.h> // Header file of conio
void main()
{
    int num, i, m;
    clrscr();
    printf("Enter any number:\n");
    scanf("%d", &num);
    for (i=1; i<=10; i++)
    {
        m = num * i;
        printf("%d = %d * %d\n", num, i, m);
    }
    getch();
}

```

Q) Write a program to obtain sum of all the numbers from 1 to 10.

```

#include <stdio.h> // Header file of stdio
#include <conio.h> // Header file of conio
void main()
{
    int i, sum=0;
    clrscr();
    for (i=1; i<=10; i++)
    {
        sum += i;
    }
}

```

printf(" sum of all numbers 1 to 10 is %d  
\\n ", sum);

getchar();

Q1) WAP to obtain following output

```
* # * # * *
# * # * # *
* # * # * *
# * # * # *
* # * # * *
```

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

```
#include <conio.h>
```

```
void main()
{
```

```
int i;
```

```
clrscr();
```

```
for (i=1; i<=25; i++)
```

```
{ if (i%5 == 1) .....
```

```
{
```

```
printf ("\\n");
```

```
}
```

```
if (i%2 == 0)
```

```
{
```

```
printf ("#\\t");
```

```
}
```

```
else
```

```
{
```

```
printf ("*\\t");
```

```
}
```

```
getch();
```

- (92) WAP to print sum of all the numbers from 1 to n . where n is entered by the user.

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int i, n, sum = 0;
    clrscr();
    printf("Enter any number: ");
    scanf("%d", &n);
    for (i=1; i<=n; i++)
    {
        sum += i;
    }
    printf("Sum of all numbers from 1 to %d\n", n);
    getch();
}
```

(93) WAP to obtain sum of all digits 100 to 999  
the output should be in following format.

```
#include < stdio.h >
#include < conio.h >
void main()
{
    int sum = 0, i, number;
    clrscr();
    for (i = 100; i <= 999; i++)
    {
        number = i;
        while (number > 0)
        {
            sum += number % 10;
            number /= 10;
        }
    }
    printf("Sum of all digits from 100 to 999 is
    "%d\n", sum);
    getch();
}
```

Armstrong number → Sum of cubes of each digit is equal to original number.

Date		

- Q4) WAP to obtain following output

1 2 3 4 5  
6 7 8 9 10  
11 12 13 14 15  
16 17 18 19 20  
21 22 23 24 25

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int i;
    clrscr();
    for (i=1; i<=25; i++)
    {
        printf("%d\t", i);
        if (i%5 == 0)
        {
            printf("\n");
        }
    }
    getch();
}
```

- Q5) WAP to print all Armstrong numbers from 100 to 999.

It is written down from mobile app / compiler (C Coding App)

```
#include <stdio.h> // function of QACW (a)
#include <math.h> // In to move taking
#include <conio.h>
void main()
{
    int num, originalnum, remainder, result = 0;
    clrscr();
    printf("Armstrong numbers between 100 & 999
    are : \n");
    for (num = 100; num <= 999; num++)
    {
        originalnum = num;
        result = 0;
        while (originalnum != 0)
        {
            remainder = originalnum % 10;
            result += pow(remainder, 3);
            originalnum /= 10;
        }
        if (result == num)
        {
            printf("%d\n", num);
        }
    }
    getch();
}
```

The output should be in below:- 153, 370,  
371, 407

- 96) WAP to accept 20 numbers from the user & print sum of all entered numbers.

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int num, i, sum = 0;
    clrscr();
    printf("Enter 20 numbers :\n");
    for (i = 1; i <= 20; i++)
    {
        scanf("%d", &num);
        sum += num;
    }
    printf("Sum of all entered numbers is %d\n", sum);
    getch();
}
```

- 97) WAP to accept 20 numbers from the user & at the end print total count of even numbers & odd number entered by the user.

```

#include <stdio.h>
#include <conio.h>
void main()
{
    int i, num, ecount = 0, ocount = 0;
    clrscr();
    printf("Enter 20 numbers\n");
    for (i=1; i<=20; i++)
    {
        scanf("%d", &num);
        if (num % 2 == 0)
            ecount++;
        else
            ocount++;
    }
    printf("Total Even count numbers: %d\n",
           ecount);
    printf("Total Odd count numbers: %d\n",
           ocount);
    getch();
}

```

(98) WAP to accept 20 numbers from the user & print individual sum of all even number & all odd numbers entered by user.

```
#include < stdio.h >
#include < conio.h >
void main()
{
    int i, esum, osum, num;
    clrscr();
    printf("Enter 20 numbers:\n");
    for (i=1; i<=20; i++)
    {
        scanf("%d", &num);
        if (num % 2 == 0)
        {
            esum = esum + num;
        }
        if (num % 2 == 1)
        {
            osum = osum + num;
        }
    }
    printf("Sum of all even numbers: %d\n", esum);
    printf("Sum of all odd numbers: %d\n", osum);
    getch();
}
```

(99) WAP to accept 20 numbers from the user & print largest of all entered number.

```
#include < stdio.h >
#include < conio.h >
void main()
{
    int i, num, lar;
    clrscr();
    printf("Enter 20 numbers : \n");
    scanf("%d", &lar);
    for (i = 1; i <= 19; i++)
    {
        scanf("%d", &num);
        if (num > lar)
        {
            lar = num;
        }
    }
    printf("Largest element is %d", lar);
    getch();
}
```

(100)

WAP. to accept the 20 numbers from the user & print second largest number among all entered numbers.

## 2) while loop:

General syntax:-

initialisation;  
while (condition)

{  
statements;  
—  
—

?      incr/decr;

Example:-

```
i=1;
while(i<=10)
{
    printf("Hello all\n");
    i++;
}
for(i=1;i<=5;i++)
{
    printf("%d\n",i);
}
```

- ① WAP to print all the numbers from 1 to 100.  
Use while loop.

```
#include <stdio.h>
#include <conio.h>
void main()
{
```

```
int i;
clrscr();
i=1;
while(i<=100)
{
```

```
printf("%d\n",i);
i++;
```

```
}
getch();
```

- ② WAP to print all the numbers from 1 to 100 in reverse number.

```
#include < stdio.h>
#include < conio.h>
void main()
{
    int i;
    clrscr();
    i = 100;
    while (i >= 1)
    {
        printf("%d\n", i);
        i--;
    }
    getch();
}
```

- ③ WAP to print all the even numbers from 1 to 100.

```
#include < stdio.h>
#include < conio.h>
void main()
{
    int i;
    i = 2;
    while (i <= 100)
    {
        printf("%d\n", i);
        i = i + 2;
    }
    getch();
}
```

- ④ WAP to accept the 20 numbers from the user & print sum of last digit of every entered number.

procedure name

```
#include <stdio.h> "qno 90 to 11A"
#include <conio.h> // b6o
void main()
{
    int count = 0, sum = 0, number, lastdigit;
    clrscr();
    printf("Enter 20 numbers:\n");
    while (count < 20)
    {
        scanf("%d\n", &number);
        lastdigit = number % 10;
        sum += lastdigit;
        count++;
    }
    printf("Sum of all last digit is : %d\n", sum);
    getch();
}
```

- ⑤ WAP to accept 20 numbers from the user,  
& print  
• "All are EVEN" - if all 20 numbers are even numbers  
• "All are ODD" - if all 20 numbers are odd numbers  
• "All are combination of EVEN & ODD numbers" - if all 20 numbers combination of even & odd.

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int numbers[20];
    int ecount=0, ocount=0, i;
    printf("Enter 20 numbers\n");
    for (i=0; i<=20; i++)
    {
        if (numbers[i] % 2 == 0)
            ecount++;
        else
            ocount++;
    }
}
```

```

if (eCount == 10) ; printing even
{
    printf("All are EVEN\n");
}
else if (oCount == 10)
{
    printf("All are ODD\n");
}
else
{
    printf("All are combination of EVEN &
        ODD\n");
}
getch();

```

- ⑥ WAP to print fibonacci series upto first 10 numbers. In these series, each new number is addition of previous two number.  
 For ex: 1 2 3 5 8 13 21 34

```

#include <stdio.h>
#include <conio.h>
void main()
{
    int n1=1, n2=2, n3, i=1;
    clrscr();
    printf("%d%d", n1, n2);
    while (i<8)
    {

```

```

n3 = n1 + n2;
printf ("%d", n3);
n1 = n2;
n2 = n3;
i++;
}
getch();
}

```

- ⑦ WAP to evaluate following expression by using while loop.

$$1^2 + 2^2 + 3^2 + 4^2 + 5^2 + 6^2 \dots + 8^2$$

```

#include < stdio.h>
#include < conio.h>
void main()
{
    int n=1;
    int sum=0;
    clrscr();
    while (n<=8)
    {
        sum+=n*n;
        n++;
    }
    printf("The sum of squares 1 to 8 is : %d\n",
           sum);
    getch();
}

```

```

#include <stdio.h>
#include <conio.h>
void main()
{
    int i=1, temp, sum=0;
    clrscr();
    while (i<=8)
    {
        temp = i*i;
        sum = sum + temp;
        i++;
    }
    printf("Result of Expression is %d", sum);
    getch();
}

```

- ⑧ WAP to evaluate following expression, using while - loop.

$$\frac{x-1}{1^2} + \frac{x-2}{2^2} + \frac{x-3}{3^2} + \dots + \frac{x-8}{8^2}$$

where, x is entered by the user.

```

#include <stdio.h>
#include <conio.h>
void main()
{
    int i=1, x;
    float temp, sum=0;
}

```

```

close();           // C:\Windows\system32\cmd.exe
printf("Enter value of x : "); // will
scanf("%d", &x); // read value

while (i<=8) // loop condition
{
    temp = (x - i)/(i*i); // division
    sum = sum + temp; // addition
    i++; // increment
}
printf("Result of Expression is %f", sum);
getch();
}

```

- ⑨ WAP to print all the numbers from n to 1. where n is entered by user.

```

#include <stdio.h>
#include <conio.h>
void main()
{
    int n;
    close(); // C:\Windows\system32\cmd.exe
    printf("Enter any number in "); // will
    scanf("%d", &n); // read value
    while (n>=1) // loop condition
    {
        printf(" %d\n", n); // output
        n--; // decrement
    }
    getch();
}

```

- ⑩ WAP to accept the number from user & print sum of all its digits.

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int num, r, s=0;
    clrscr();
    printf("Enter any number : ");
    scanf("%d", &num);
    while (num>0)
    {
        r = num % 10;
        s = s + r;
        num = num / 10;
    }
    printf("Result is %d ", s);
    getch();
}
```

- ⑪ WAP to accept the number from user & print reverse of them. Use while loop.

```

#include <stdio.h>
#include <conio.h>
void main ()
{
    int num, r, s=0;
    clrscr();
    printf("Enter any number : ");
    scanf("%d", &num);
    while (num>0)
    {
        r=num%10;
        s=s*10+r;
        num=num/10;
    }
    printf("Result is %d ", s);
    getch();
}

```

## \* Keyword "break"

- is used to terminate any loop or switch-case.
- use of keyword "break" is valid only within loop & switch-case.
- In loop, use of keyword "break" should be condition based.
- If we use "break" outside loop or switch case, then it will show compile time error. ("Misplaced break")

```
for(i=1; i<=5; i++)
{
```

```
    printf("Start - %d\n", i);
```

```
    if(i==3)
    {
```

```
        break;
```

```
    }
    printf("End - %d\n", i);
```

## \* There are two legal ways to terminate a loop.

- when condition becomes false.
- on "break".

\* Keyword "continue"

- is to take the controls back to condition
- use to "continue" is valid only inside loop.

```

for (i=1; i<=5; i++)
{
    printf("Start - %d\n", i);
    if (i==3)
        continue;
    printf("End. - %d\n", i);
}

```

\* Use of keyword "continue" should be condition based.

- (12) WAP to accept a number from the user & print whether it is a prime number or not.

```

#include <stdio.h>
#include <conio.h>
void main()
{
    int num, i=2, flag=1;
    clrscr();
    printf("Enter any number :\n");
    scanf("%d", &num);
}

```

```

while (i<num)
{
    if (num% i == 0)
    {
        flag = 2;
        break;
    }
    i++;
}

if (flag == 1)
{
    printf("The number is prime");
}
else if (flag == 2)
{
    printf("The number is not prime");
}
getch();
}

```

- (13) WAP to accept the number from user & print its divisors.

```

#include <stdio.h>
#include <conio.h>
void main()
{
    int num, i = 1, flag = 1;
    clrscr();
    printf("Enter any number: \n");
    scanf("%d", &num);

```

```

while (i <= num)           (common divisors)
{
    if (num % i == 0)      (if common)
    {
        printf ("%d\n", i);  (print)
    }
    i++;                   (i++)
}
getch();                   (end)
}

```

- (14) WAP to accept two numbers from the user & print their common divisors.

```

#include <stdio.h>
#include <conio.h>
void main()
{
    int num1, num2, sm, i;
    clrscr();
    printf("Enter two numbers\n");
    scanf("%d%d", &num1, &num2);

    sm = num1 < num2 ? num1 : num2;
    i = 1;
    while (i <= sm)
    {
        if (num1 % i == 0 && num2 % i == 0)
        {
            printf ("%d\n", i);
        }
        i++;
    }
    getch();
}

```

(15) Example of nested loops

```

int i, j;
for (i=1; i<=3; i++)
{
    for (j=1; j<=3; j++)
    {
        printf("%d - %d\n", i, j);
    }
}
getch();

```

(16)

```

#include <stdio.h>
#include <conio.h>
void main()
{
    int i, j, k;
    clrscr();
    for (i=1; i<=2; i++)
    {
        for (j=1; j<=2; j++)
        {
            for (k=1; k<=2; k++)
            {
                printf("%d - %d - %d\n", i, j, k);
            }
        }
        printf("Hello\n");
    }
    printf("Hi\n");
}
getch();

```

Output:-

(17) #include <stdio.h>  
 #include <conio.h>  
 void main()  
 {  
 int i, j;  
 clrscr();  
 for (i=1; i<=5; i++)  
 {  
 for (j=1; j<=i; j++)  
 {  
 printf(" \* \t");  
 }  
 printf("\n");  
 }  
 getch();  
}

Output:-

(18) #include <stdio.h>  
 #include <conio.h>  
 void main()  
 {  
 int i, j;  
 clrscr();  
 for (i=5; i>=1; i--) \*  
 {  
 for (j=1; j<=i; j++) \*  
 {  
 printf(" \* \t");  
 }  
 printf("\n");  
 }  
 getch();  
}

19

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int i, j;
    clrscr();
    i = 1;
    while (i <= 5)
    {
        j = 5;
        while (j >= i)
        {
            printf(" *%t");
            j--;
        }
        printf("\n");
        i++;
    }
    getch();
}
```

20 WAP to accept value of base & power, & calculate the result of base raised to power.

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int base, power;
    long result = 1;
    clrscr();
```

Date \_\_\_\_\_

```

printf("Enter Base:\n");
scanf("%d", &base);

printf("Enter Power:\n");
scanf("%d", &power);

for( i=0; i<power; i++)
{
    result = result * power;
}

printf("The base raised to power is %d", result);
getch();
}

```

(21) WAP to obtain following output:-

```

1
1 2
1 2 3
1 2 3 4
1 2 3 4 5

```

```

#include <stdio.h>
#include <conio.h>

```

```

void main()
{

```

```

    int i, j;

```

```

    clrscr();

```

```

    for( i=1; i<=5; i++)
    {

```

```

        for( j=1; j<=i; j++)
        {

```

```

    printf("%d\n", j);
}
printf("\n");
getch();
}

```

② WAP to obtain following output :-

1

2 2

3 3 3

4 4 4 4

5 5 5 5 5

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

```
#include <iomanip.h> // iomanip
```

```
void main()
```

```
{
```

```
    int i, j;
```

```
    clrscr();
```

```
    for (i=1; i<=5; i++)
```

```
{
```

```
        for (j=1; j<=i; j++)
```

```
{
```

```
            printf("%d\t", i);
```

```
}
```

```
            printf("\n");
```

```
}
```

```
    getch();
```

```
{
```

(23) WAP to obtain following output:-

A

A B

A B C

A B C D

A B C D E

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

```
#include <conio.h>
```

```
void main()
```

{

```
int i, j;
```

```
char c;
```

```
clrscr();
```

```
for (i = 65; i <= 69; i++)
```

{

```
    for (j = 65; j <= i; j++)
```

{

```
        printf("%c\t", j);
```

}

```
    printf("\n");
```

}

```
getch();
```

}

(4) WAP to obtain following output:-

A

B B

C C C

D D D D

E E E E E

```

#include <stdio.h>
#include <conio.h>
void main()
{
    int i, j;
    char c;
    clrscr();
    for (i = 65; i <= 69; i++)
    {
        for (j = 65; j <= i; j++)
        {
            printf("%c\t", i);
        }
        printf("\n");
    }
    getch();
}

```

- (25) WAP to obtain following output

```

5
4 4
3 3 3
2 2 2 2
1 1 1 1 1

```

```

#include <stdio.h>
#include <conio.h>
void main()
{
    int i, j;
    clrscr();
    for (i = 5; i >= 1; i--)
    {
        for (j = 5; j >= i; j--)
            printf("%d\t", i);
        printf("\n");
    }
    getch();
}

```

Q6) WAP to obtain following output:-

```

5 4 3 2 1
      5 4 3 2 1
          5 4 3 2 1
              5 4 3 2 1
                  5 4 3 2 1

```

```

#include <stdio.h>
#include <conio.h>
void main()
{
    int i, j;
    clrscr();
}

```

```

for (i=5; i>=1; i--) {
    for (j=5; j>=i; j--) {
        printf("%d\t", j);
    }
    printf("\n");
}
getch();

```

- (27) WAP to obtain following output:-

```

A
A B
A B C
A B C D
A B C D E

```

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

```
#include<conio.h>
```

```
void main()
```

```
{
```

```
int i, j;
```

```
char c;
```

```
clrscr();
```

```
for (i=65; i<=69; i++)
```

```
{
```

```
for (j=65; j<=i; j++)
```

```
{
```

```
printf("%c\t", j);
```

```
}
```

```
printf("\n");
```

```
getch();
```

- (28) WAP to obtain following output:-

1  
2 3 4 5 6

7 8 9 10  
11 12 13 14 15

```
#include < stdio.h>
#include < conio.h>
void main()
{
    int i, j, n = 5; num = 1;
    clrscr();
    for (i = 1; i <= n; i++)
    {
        for (j = 1; j <= i; j++)
        {
            printf("%d\t", num);
            num++;
        }
        printf("\n");
    }
    getch();
}
```

- (29) WAP to obtain following output:-

\*

#

\*

\*

\*

#

\*

\*

\*

#

\*

\*

\*

#

\*

\*

```

#include <stdlib.h>
#include <conio.h>
void main()
{
    int num = 1, i, j;
    clrscr();
    for (i = 1; i <= 5; i++)
    {
        for (j = 1; j <= i; j++)
            num % 2 == 1 ? printf("*\t") : printf("#\t");
        num++;
    }
    printf("\n");
    getch();
}

```

(30) WAP to obtain following output:

```

#include <stdio.h>
#include <conio.h>
void main()
{
    int num = 1, i, j, k;
    clrscr();
    for (i = 11; i <= 15; i++)
    {
        for (j = 15; j > i; j--)
            printf("=");
    }
}

```

```

for (k=11; k<=i; k++)
{
    printf("*");
}
printf("\n");
getch();

```

(31) WAP to obtain following output:-

```

-----1
      2 3
      4 5 6
      7 8 9 10
11 12 13 14 15

```

```

#include <stdio.h>
#include <conio.h>
void main()
{
    int num=1;
    int i, j, k;
    clrscr();
    for (i=11; i<=15; i++)
    {
        for (j=15; j>i; j--)
        {
            printf("-");
        }
        for (k=11; k<=i; k++)
        {
            printf("%d", num);
            num++;
        }
    }
}

```

```

    printf("\n");
}
getch();
    
```

- (32) WHP to obtain following output:-

```

-----1
---+ 1 2
--1 2 3
-1 2 3 4
1 2 3 4 5
    
```

```

#include <stdio.h>
#include <conio.h>
void main()
{
    int num = 1, i, j, k;
    clrscr();
    for (i = 1; i <= 5; i++)
    {
        for (j = 5; j >= i; j--)
        {
            printf("- ");
        }
        for (k = 1; k <= i; k++)
        {
            printf("%d ", k);
        }
        printf("\n");
    }
    getch();
}
    
```

(33) WAP to obtain following output :-

```

      - - - - 5
      - - - 5 4
      - - 5 4 3
      - 5 4 3 2
      5 4 3 2 1
  
```

```

#include <stdio.h>
#include <conio.h>
void main()
{
    int i, j, k;
    clrscr();
    for(i=5; i>=1; i--)
    {
        for(j=i; j>1; j--)
        {
            printf(" - ");
        }
        for(k=5; k>=i; k--)
        {
            printf(" %d ", k);
        }
        printf("\n");
    }
    getch();
}
  
```

(34) WAP to obtain following output :-

```

      * # 
      - - * * #
      - - * * * #
      - * * * * #
      * * * * * #
  
```

```

#include <stdio.h>
#include <conio.h>
void main()
{
    int i, j, k;
    clrscr();
    for(i = 11; i <= 15; i++)
    {
        for(j = 15; j > i; j--)
        {
            printf(" - ");
        }
        for(k = 11; k <= i; k++)
        {
            printf("* ");
        }
        printf("#");
        printf("\n");
    }
    getch();
}

```

- ③ WAP to obtain following output:-

1 2 3 4 5 6 7 8 9 10

:  
:  
;

91 92 93 94 95 96 97 98 99 100

```

#include <stdio.h>
#include <conio.h>
void main()
{
    int i, j;
    clrscr();
    for (i = 1; i <= 100; i++)
    {
        printf("%d\t", i);
        if (i % 10 == 0)
        {
            printf("\n");
        }
    }
    getch();
}

```

- 36) WAP to obtain following output:

```

-----*
---* * *
--* * * *
-* * * * *
* * * * * *

```

```

#include <stdio.h>
#include <conio.h>
void main()
{
    int i, j, k, temp = 1;
    clrscr();
}

```

```

for (i=1; i<=5; i++)
{
    for (j=5; j>i; j--)
        printf(" - ");
    printf(" * ");
    temp = temp + 2;
    printf("\n");
}
getch();
    
```

Q7) WAP to obtain following output:

```

-----*
-----* * #
-----* * * # # #
-----* * * * # # # #
    
```

```

#include <stdio.h>
#include <conio.h>
void main()
{
    int i,j,k,l;
    clrscr();
    
```

```

for(i=1; i<=8; i++)
{
    for(j=5; j>i; j--)
    {
        printf(" -");
    }
    for(k=1; k<=i; k++)
    {
        printf("*");
    }
    for(l=1; l<i; l++)
    {
        printf("#");
    }
    printf("\n");
}
getch();
    
```

- (38) WAP to obtain following output:-

```

1 1 1 1 1 1 1 1
2 2 2 2 2 2 2 2
3 3 3 3 3 3 3 3
4 4 4 4 4 4 4 4
5 5 5 5 5 5 5 5
    
```

Output block

1 2 3 4 5 6 7 8 9

```

#include <stdio.h>
#include <conio.h>
void main()
{
    int i, j, k, n = 5;
    clrscr();
    for (i = 1; i <= n; i++)
    {
        for (j = 1; j <= n - i; j++)
            printf("%d", j);
        printf("\n");
        for (k = 1; k <= i * 2 - 1; k++)
            printf("odd", k);
        printf("\n");
    }
    getch();
}

```

- ③ WAP to obtain following output:-

```

1
1 2 3
1 2 3 4 5
1 2 3 4 5 6 7
1 2 3 4 5 6 7 8 9

```

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int i, j, k, n = 8;
    clrscr();
    for (i = 1; i <= n; i++)
    {
        for (j = 1; j <= n - i; j++)
        {
            printf(" - ?");
        }
        for (k = 1; k <= i * 2 - 1; k++)
        {
            printf("%d.", k);
        }
        printf("\n");
    }
    getch();
}
```

3) do-while loop :-

- General syntax:-

initialization;

do

statements;

    }
     {
     incr/decr;
     }
     while (condition);

\* Notes:- An important characteristic about do-while loop is, it at least iterates once.

(40) WAP to print "Hello all" message 10 times.  
Use do-while loop.

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

```
#include <conio.h>
```

```
void main()
```

```
{ int i=1;
```

```
clrscr();
```

```
do
```

```
    {
```

```
        printf("Hello all\n");
```

```
        i++;
```

```
    } while (i<=10);
```

```
getch();
```

```
}
```

- (41) WAP to print all the numbers from 100 to 1. Use do-while loop.

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int i=100;
    clrscr();
    do
    {
        printf("%d\n", i);
        i--;
    } while (i>=1);
    getch();
}
```

\* Difference betn while loop & do-while loop.

### While loop

- ① While loop is also called as entry controlled loop.
- ② There is no semicolon (;) at the end of while condition.
- ③ In while loop, first condition is being checked, then statements execute

### do-while loop

- ① Do-while loop is also called as exit controlled loop.
- ② There is a semicolon at the end of while condition.
- ③ In do-while loop, first statements execute & then condition is being checked.

- ④ In while loop, condition is being checked to decide whether current iteration is possible or not.
- ⑤ If initial condition is false, then while loop performs no iteration.
- ⑥ While loop is preferred when numbers of iterations are not known. It depends on any internal situation or flag value.
- ④ In do-while loop condition is being checked to decide whether next iteration is possible or not.
- ⑤ If initial condition is false, then also do-while loop iterates at least one.
- ⑥ Do-while loop is preferred when at least one iteration is required, & in user driven programs.
- ⑦ General Syntax:-

```
initialization;
while (condition)
{
```

statements;

—

—

incr/decr;

}

```
initialization;
do
{
```

statements;

—

—

incr/decr;

} while (condition);

⑦ General syntax:-

(42) WAP to accept the numbers till user want & at the end & display sum of all entered numbers.

(43) WAP to accept the numbers till user wants & at the end print the count of all even numbers & odd numbers entered by the user.

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int num, ecount = 0, ocount = 0;
    char ch;
    clrscr();
    do
    {
        printf("Enter a number : \n");
        scanf("%d", &num);
        num % 2 == 0 ? ecount++ : ocount++;
        fflush(stdin);
        printf("Do you want to enter another
               number (y/n) : \n");
        scanf("%c", &ch);
    } while(ch != 'y' || ch == 'Y');

    printf("Total count of EVEN numbers : %d\n",
          ecount);
    printf("Total count of ODD numbers : %d\n",
          ocount);
    getch();
}
```

44

WAP to obtain following output :-

2016-09-20 10:17:49 [INFO] [main] Starting the main loop

It is feasible to implement a 100% solar power system

~~2010-2011~~

Both species have very old histories

*Final (d) - Summary*

100% 100% 100%

*(V. 14) The following student*

• Need more time to familiarize ourselves

Index 9

• As soluções são mais lentas

(45)

WAP to accept the numbers till user wants & for every number print its square & cube.  
Use do-while loop.

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int num;
    clrscr();
    do
    {
        printf("Enter any number: ");
        scanf("%d", &num);
        printf("Square: %d\n", num * num);
        printf("Cube: %d\n", num * num * num);
    } while (num != 1);
}
```

getch();

(46)

WAP to accept the numbers till user wants & each iteration print factorial of enter number.

```
#include <stdio.h>
#include <conio.h>
void main()
{
```

```

int num, factorial, i;
do
{
    printf("Enter a number: ");
    scanf("%d", &num);

    factorial = 1;
    for (i=1; i<=num; i++)
    {
        factorial = factorial * i;
    }

    printf("Factorial of %d is %d\n", num,
           factorial);
}
while (num != 1 || num != 0);
getch();
}

```

(47) WAP to accept the numbers from the user & print largest digit of that number.

```

#include<iostream.h>
#include<conio.h>
void main()
{
    int n1, n2, largest digit = 0;
    clrscr();
}

```

```
printf("Enter any number: ");
scanf("%d", &n1);
```

```
if (n1 < 0)
```

```
}
```

```
n1 = -n1;
```

```
do
```

```
{
```

```
n2 = n1 % 10;
```

```
if (n2 > largest digit)
```

```
{
```

```
largest digit = n2;
```

```
}
```

```
n2 /= 10;
```

```
} while (n1 > 0);
```

```
printf("The largest digit in the number is %d\n",
      largest digit);
```

```
getch();
```

- 48) WAP to accept a number from the user & print whether it is a strong number or not. A number is strong number if sum of number of each digit = original number.

Not  
sum

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int i, r;
    long num, sum = 0, temp;
    printf("Enter any number\n");
    scanf("%d", &num);

    temp = num;
    while (num > 0)
    {
        r = num % 10;
        f = 1;
        for (i = 1; i <= r; i++)
        {
            f = f * i;
        }
        sum = sum + f;
        num = num / 10;
    }
    if (temp == sum)
        printf("Number is a strong number");
    else
        printf("Number is not a strong number");
    getch();
}
```

## \* Switch-case Statement or case control Statement

Menu Driven Statement

OR

Page No.  
Date

To implement a switch-case we perform the following steps:-

Step 1:- Accept value on which you want to perform operation.

Step 2:- Show menu of operations & accept choice-number.

Step 3:- Write switch-case containing all operations (of menu).

\* To use switch-case, general syntax is :-

switch (switch-variable) {

    case 1: statements;

        break;

    case 2: statements;

        break;

    break;

    case n: statements;

    break;

    default: statements;

}

    break;

}

- \* Here in switch-case, we use total 4 keywords:  
 ① switch    ② case    ③ break    ④ default
- \* Keyword "continue" has no role in switch-case.

(49) WA menu driven program to perform following operations on an integer value.

- To print square & cube
- To print last digit
- To check EVEN or ODD

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int num, och;
    int s, c;
    clrscr();
}
```

// Step 1: Accepting number to perform operation

```
printf("Enter any number : ");
scanf("%d", &num);
```

// Step 2: Showing Menu of operations:

```
printf("Select an operation\n");
printf("1 - To obtain Square & cube\n");
printf("2 - To obtain Last Digit\n");
printf("3 - To check EVEN or ODD\n");
printf("Provide your choice : ");
scanf("%d", &och);
```

11 Step 3: Writing switch-case statement  
switch (choice)

{  
case 1:

```
s = num * num;
c = num * num * num;
printf("Square is %d\n", s);
printf("Cube is %d\n", c);
break;
```

case 2:

```
s = num % 10;
printf("Last digit = %d\n", s);
break;
```

case 3:

```
num % 2 == 0 ? printf("EVEN")
: printf("ODD");
break;
```

default : printf("Invalid choice");

} // switch (choice);

getch();

(Q) WAP to perform all arithmetic operation on two integers by switch-case:

1 - Addition

2 - Subtraction

3 - Multiplication

4 - Division

```
#include <stdio.h>
#include <conio.h>
void main()
{
    float n1, n2, result;
    int och;
    clrscr();
    printf("Enter two numbers : \n");
    scanf("%f %f", &n1, &n2);
    printf("Select an operation : \n");
    printf("1 - Addition \n 2 - Subtraction \n 3 -\n Multiplication \n 4 - Division \n");
    scanf("%d", &och);

    switch(och)
    {
        case 1: result = n1 + n2;
        printf("The addition of 2 numbers\n is %f \n", result);
        break;

        case 2: result = n1 - n2;
        printf("The subtraction of 2 numbers\n is %f \n", result);
        break;

        case 3: result = n1 * n2;
        printf("The multiplication of 2\n numbers is %f \n", result);
        break;

        case 4: if (n2 != 0)
        {
            result = n1 / n2;
        }
        else
            result = 0;
        printf("The division of 2 numbers\n is %f \n", result);
        break;
    }
}
```

## NOTES

```
printf("The Division of 2 numbers is %f\n",  
      result);  
    }  
    break;  
  default: printf ("Invalid choice number ..");  
  }  
  getch();  
}
```

(51) Write a menu driven program for following choices.

A - To check even or odd.

B - To check positive or negative

C - To check whether multiple of 5 or not.

D - To print multiplication table.

• 100% of the students have no disabilities  
• 100% of the students are English speakers

• Black or brown students at 100%

• Hispanic or Latino students at 90%

• Asian or Pacific Islander students at 90%

• Other multiracial students at 90%

## \* Keyword "goto" or Jump Statement:-

- The goto statement takes the controls at Specified 'label' where a label is a sticker / mile stone to indicate where control should jump.
- The goto statement use general syntax is:-

**goto label-name;**

- To create a label, general syntax is :-

**label-name:**

- A label name is also an identifier, therefore we have to follow same five identifier rules to give label name.

For example:-

```
① #include<stdio.h>
    #include<conio.h>
    void main()
    {
        int x=15;
        clrscr();
        printf("Hello 1\n");
        printf("Hello 2\n");
        if(x>10)
        {
            goto xyz;
        }
    }
```

Output:-

```

printf("Hello 3\n");
printf("Hello 4\n");
xyz:
printf("Hello 5\n");
printf("Hello 6\n");
getch();
}

```

Hello 1  
Hello 2  
Hello 5  
Hello 6

(2) #include &lt;stdio.h&gt;

#include &lt;conio.h&gt;

void main()

{

```

int x=5;
clrscr();
printf("Hello 1\n");
printf("Hello 2\n");
if(x>10)
{
    goto xyz;
}

```

printf("Hello 3\n");

printf("Hello 4\n");

xyz:

printf("Hello 5\n");

printf("Hello 6\n");

getch();

}

Output:-

Hello 1  
-11- 2  
-11- 3  
-11- 4  
-11- 5  
-11- 6

From above a program, note that the label xyz don't have body or logical block if it is only a sticker/milestone that specifies where controls should jump.

Above example demonstrates downward jump at label xyz. But we can also make upward jump using goto statement.

For example:-

(52)

```
#include <stdio.h>
#include <conio.h>
void main()
{
    clrscr();
    printf("main starts\n");
    int i=1;
    xyz:
    printf("Hello cell-%d\n",i);
    i++;
    if(i<=10)
    {
        goto xyz;
    }
    printf("main ends\n");
    getch();
}
```

From above example:, note that if logically used then we can use goto statement as an alternate of do-while loop. By using goto statement, we can only jump within that function.

By using goto statement we can jump to a label, present at anywhere in the function.

\* But according to structured programming algorithms, use of "goto" should be avoided.

\* Important points on switch-case:-

1) Case numbers can be any negative or positive or zero. It is not necessary to be 1, 2, 3, 4, ... only. We keep easy case numbers for the comfort of users.

For eg:-

```
(53) switch (ch)
{
    case 19230: statements;
    break;
    case -35: statements;
    =
    break;
    case -3774: statements;
    =
    break;
}
```

2) case numbers can be integers & characters only. character cases must be in single cots.

For e.g:-

(54) switch (ch)  
{

case 'A': statements;

=

break;

case 'B': statements;

=

break;

case 'C': statements;

=

break;

}

3) We can also use special symbols as case number, but has character case but in single cots.

For e.g:-

(55) switch (ch)  
{

case '+': statements;

= short

break;

case '-': statements;

= short

break;

case '\*': statements;

= short

break;

}

4) Floating point values & string values are not allowed as case number.

For eg:-

(56) switch (ch)

{  
is not  
allowed

case 3.5 : statements;

=

break;

case 17.263 : statements;

=

break;

case "three" : statements;

=

break;

}

5) case numbers must be unique (duplicate) cases are not allowed.

For eg:-

(57) switch (ch)

{  
is not  
allowed

case 3 : statements;

=

break;

case 7 : statements;

=

break;

case 3 : statements;

=

break;

}

6) case numbers can be in any random order. Even we can write a default case at any order. The case numbers should not be necessarily in ascending or descending order.

for eg:-

(S8)

```
switch (ch)
{
```

case 4 : statements;

=

break;

case 1 : statements;

=

break;

default : statements;

=

break;

case 2 : statements;

=

break;

?

7) The default case executes when number case numbers matches. Even writing default case is optional.

8) If "break" is not used, then next written case automatically executes. This will continue unless "break" appears or "switch-case end". This process is called as fall-down.

\* Writing "break" is optional for last case.

9) Variables are not allowed as case.

for eg:-

⑨ int a=1, b=2, c=3, d=4;  
switch (ch)  
{

case a : statements;

=

break;

case b : statements;

=

break;

case c : statements;

=

break;

case d : statements;

=

break;

}

10) Expression are allowed as case numbers  
but only console expression are allowed,  
variable expression are not allowed.

for eg:-

⑩ int a=1, b=2, c=3, d=4;  
switch (ch){

case 8/2 : statements;

=

break;

case 9+5 : statements;

=

break;

allowed

case a+b: statements;

are not  
allowed

=

break;

case → a+b: statements;

break;

}

11) Logical operators are not allowed in cases number.

for e.g:-

(61) switch(ch)  
{

case 'A' && 'a': statements;

are  
not  
allowed

break;

case 'm' || 'M': statements;

=

break;

case ! 'A' : statements;

=

break;

}

12) Characters case are case sensitive.

13) Blank cases are allowed

(means cases without statements)

for eg:-

(62)

switch (ch)

{

case '+':

case 'A':

case 'a':  $z = x * y;$ 

printf("Multiplication is %d", z);

break;

case '\*':

case 'm':

case 'M':  $z = x + y;$ 

printf("Addition is %d", z);

break;

}

## \* Functions

Every program that we write till date, contains only main () methods. and we write different logical statements in body of main () method.

But when numbers of operation increases, then it becomes difficult to write every logical block in main () method & maintain different variables. As a result, instead of writing every logic in main method, we should prefer to define individual function for every operation.

A function is set of statements which can be re-used.

There are two (2) main reasons to define a function.

- 1) for reusability
- 2) to reduce complexity

If you find, any operation is required to perform multiple times, then instead of writing it multiple times, define a function for it & call multiple times.

If you find, there are multiple operation being performed in main () method then instead of every operation in main () method prefer to define individual function for every operation & call it whenever required.

\* A Good programmer divides / distributes every operation in individual function because of this:-

- Program becomes distributed
- Error finding becomes easy
- Maintaining variables becomes easy.
- Improve the reusability.

\* There are 3 part of function:-

- Definition of function (Body of function)
- Calling of function (to use function)
- Declaration of function

\* One function, one operation.

\* We can define any numbers of functions in a program.

\* A program is set of functions.

\* A function is set of statements & logical blocks.

\* Every executable statement & logical block must be inside function's body.

\* To define function:-

```
returntype function-name (parameter-list)
{
    statements;
}
```

*function's body*

\* To call a function:-

```
function-name (parameter-list);
```

\* To declare a function:-

```
returntype function-name (parameter-list);
```

\* A program is set of functions.

\* C/C++ & Java are structured languages.  
   ∴ only functions execute.  
   ∴ Every executable statement & logical-blocks  
     must be inside function's body.

\* In a program, we can define any numbers  
   of functions.

\* But nested functions are not allowed.  
   i.e., we cannot define body of one function  
     inside body of another function.  
   i.e., body of a function must start only  
     after completing body of previous function.

- \* In a program, there can be any numbers of functions.
- \* But execution begins from main() only.
- \* A program without main() is allowed.  
But such program will not begin execution.
- \* A function's name is also identifier.  
∴ We have to follow those 5 rules for functions name also.
- \* Functions name must be unique.  
(C does not support function overloading)
- \* Always prefer to define every user defined functions before/above main().

63) #include < stdio.h >

#include < conio.h >

- print-msg()

{

printf("I got the result");

printf("It is correct");

printf("Go to next step");

printf("Good day");

}

void main()

{

int a,b,c,d,e,f,g;

clrscr();

a=10;

b=20;

```
c = a+b;           : without curly braces
print-msg(); ← calling to print-msg()
d = b-a*c;
print-msg(); ← calling
e = b*a - d+c;
print-msg(); ← calling
f = d*c - h + a;
print-msg(); ← calling
g = f-a/b + c+d;
print-msg(); ← calling
getch();
}
```

- \* "void" is one of return type.
- \* To use a function, we must call it.

or

A function executes if & only if we call it.

or

A function will not execute automatically, if we do not call it.

- \* We can call to function from any logical-block.
- \* We can call a function multiple times.
- \* We can define functions in any order.
- \* The function which calls to another function is "calling function".
- \* The function which is being called by another function is "called function".

- # When a function is called, the calling function transfers its own controls to the body of called function & itself gets paused.
- # When execution of called function completes, the controls will be returned back to the statement from where it was called And calling function resumes.
- # this means, exact one function executes at a time.

64 #include <stdio.h>  
 #include <conio.h>

```

RT chair()
{
    printf("one");
}

RT car()
{
    printf("two");
    chair();
    printf("three");
}

RT table()
{
    printf("Four");
    chair();
    printf("Five");
}
  
```

```

    car();
    printf("Six");
}

void main()
{
    clear();
    printf("Seven");
    car();
    printf("Eight");
    table();
    printf("Nine");
    chair();
    printf("Ten");
    getch();
}

```

\* Any function can call to any other function  
any number of times.

\* Define a function related to addition of 2 numbers  
Call this function from main() method.

(65)

```

# include <stdio.h>
int addition()
{
    int a,b,c;
    printf("Enter 2 values\n");
    scanf("%d %d", &a, &b);
    c = a+b;
    printf("Addition is %d", c);
}

void main()
{
    clear();
    addition();
    getch();
}

```

\* Any function can call to any other function any number of times.

\* There are 3 types of variables (based on where we declare)

1. local variables:- Which are declared inside body of a function.

(Scope/availability of local variable is within the body of that function in which it is declared.)

2. global variables: Which are declared outside body of a function.

(Scope of global variables everywhere in that program (i.e. in every function))

3. block variables:- Which are declared inside any logical-block.

(Scope of block variable is within that logical-block.)

Notes:-

\* Prefer to avoid use of global variables.

\* Life time of any memory-block is upto end of execution.

\* Multiple functions can have same names for local variables.

\* But names of local variables & block variables cannot be same.

- \* Best names of global variables, local variables & block variables cannot be same.
- \* Even the variables declared in main() are also local-variables of main().

# Following program is not valid / invalid :-

① #

#

RT addition()

```
{ int c; ← local variables of addition()
    c=a+b;
}
void main()
{ }
```

```
int a,b; ← local variables of main()
printf("Enter 2 values\n");
scanf("%d %d", &a, &b);
addition();
printf("Addition is %d\n", add());
getch();
```

② #

void main()

{

```
int a,b; ← are local variables of main()
clrscr();
```

```
printf("Enter 2 values\n");
```

```
scanf("%d %d", &a, &b);
```

```
addition();
```

```
    printf("Addition is %d\n", c);
    getch();
}
RT addition()
{
    int c; ← is local variable of addition()
    c=a+b;
}
```

- \* Above program is not valid. Because of scope of local variable.
  - \* We know that scope of local variables is within that function in which they are declared.
  - \* Multiple functions can have same names for local variables.
  - \* But they will have different memory-blocks.
  - \* To cross-check memory-blocks we can print their address & check.
  - \* In above program, just declare a, b & c as global variables (instead of local variables), and it will execute without error.
  - \* Never prefer to declare global variables.
  - \* But global constants are fine to declare.

#

```

void main()
{
    int a,b;
    clrscr();
    printf("Enter 2 values\n");
    scanf("%d %d", &a, &b);
    addition(a,b);
}

```

calling to addition() & passing  
addition(a,b) ← values of a & b as parameter  
 ↑  
A.oP

```

} declared 2 parameter
RT addition(int x, int y) ← to receive passed
{ values of a & b
  int z; F.P
  z = x+y;
  printf("Addition is %d", z);
}

```

- \* This mechanism of passing values is called as "Parameter-passing".
- \* The parameters which are passed while calling is called as "Actual parameters/ arguments".
- \* The parameters in which Actual parameters are received are called as "Formal parameters".
- \* Every Formal parameters must be individually, declared with datatype.
- \* Names of Actual parameters & Formal parameters can be same.
- \* These formal parameters are also considered as local variables of that function.

- \* For any function, numbers of Actual parameters & formal parameters must be same.
- \* Prefer to use functions according to it's formal parameters.
- \* The formal parameters are optional part of any function. i.e. A function's body without parameters is valid.

67)

#include <iostream.h>

void main()

{ int a,b,p,q,i,j,h,r,x,t;

printf("Enter 2 values\n");

scanf("%d %d", &a, &b);

addition(a,b);  $\leftarrow$  A.o.P + x = s

printf("Again Enter 2 values\n");

scanf("%d %d", &p, &q);

addition(p,q);  $\leftarrow$  A.o.P

printf("Again enter 2 values\n");

scanf("%d %d", &i, &j);

addition(i,j);  $\leftarrow$  A.o.P

printf("Again enter 2 values\n");

scanf("%d %d", &l10, &l20);

addition(l10,l20);  $\leftarrow$  A.o.P

getch();

- \* we use/call a function according to its F.o.P. list.

- \* We use also pass console values to A.P.
- \* For any function:
  - return type \_\_\_\_\_ is mandatory
  - function name \_\_\_\_\_ is mandatory
  - body \_\_\_\_\_ is mandatory
  - F.o.P. \_\_\_\_\_ optional
- \* If we call a function multiple times, then each time it creates new memory-blocks.
- \* The A.P & F.o.P are different memory-blocks.
  - ∴ If we update in F.o.P. then it will not affect value of A.P.

①

Void main()

{

int a=5; A.P.

check(a);

printf("a=%d\n", a);

getch();

}

R1 check (int x) ← F.o.P.

{

x++;

Increment in x

②

#

Void main()

{

int a,b;

printf("Enter 2 values\n");

```
scanf("%d %d", &a, &b);
add(a,b); ← passed a & b as A.o.P.
=
subtraction(a,b); ← passed a & b as A.o.P.
=
multiplication(a,b); ← passed a & b as A.o.P.
=
getch();
}

RT add (int x, int y)
{
    int z;
    z = x+y;
    printf("Addition is %d", z);
}

RT subtraction (int x, int y)
{
    int z;
    z = x-y;
    printf("Subtraction is %d", z);
}

RT multiplication (int x, int y)
{
    int z;
    z = x*y;
    printf("Multiplication is %d", z);
}
```

- \* We can pass same A.o.P to multiple function.
- \* C & Java don't allow default argument.

68

~~return type~~~~values~~~~is to receive  
returned  
value of~~~~2~~~~?~~~~returntype~~~~int addition (int x, int y)~~~~?~~~~F.o.P.~~~~int z;~~ ~~$z = x + y;$~~ ~~return z; ← returning value of z~~~~?~~

- \* A good programmer should never print resulting values in function's body.
- \* But instead, prefer to return the resulting value by using keyword "return".
- \* If a function returns, then receive the returned value of left-side of calling statement.
- \* The datatype of returned value / variable will be the returntype of that function.
- \* If a function is not returning then returntype will be "void".
- \* "void" means "no return".

- \* A function can return only one value.
  - \* The keyword "return" not only returns value but it also returns control.
    - ∴ Statements after "return" will not execute.
    - ∴ "return" should be last statement of any function.
  - \* It is optional to receive returned value.
  - \* The parameters and returning are not related.
  - \* The datatypes of parameters and return-type has no concern.
- \* There are 4 types of functions :-

- 1) with parameter & with return
- 2) with parameter & without return
- 3) without parameter & with return
- 4) without parameter & without return.

- \* To declare a function, use the signature/header of function as it is.
- \* If body of called function is appearing after body of calling function is mandatory.
- \* If body of called function is appearing above / before body of calling function then declaration of called function is optional.

## \* Pointer (%u range (1 to 65535)).

- \* We know that, every memory-block has address.
- \* This address is automatically assigned.
- \* This address is always positive, non-zero, integer.
- \* We can logically use the address, but we cannot change or generate the address.
- \* Whenever we use variable's name in a statement, then by default it works on value of variable.
- \* We have an additional operator & to work with address of variable.

"&" - "address of operator"  
 OR "address specifier"  
 OR "referencing operator"

- \* To print address, use %u as format specifier.

%u - unsigned int.

(69)

```
#include <stdio.h>
#include <conio.h>
void main()
{
    clrscr();
    int a=5;
    float b=9.2;
    printf("Value of a is %d\n",a);
    printf("Address of a is %u\n", &a);
    printf("Value of b is %f\n", b);
    printf("Address of b is %u\n", &b);
    getch();
}
```

\* If we run a program multiltimes, then in each execution, the memory-location (address) of variables may differ.

- A pointer is variable (memory-block) which stores address of another variable (memory-block).
- Similar to normal variables, we also have to declare pointer.

datatype \*pointer-name;

- The datatype of pointer & datatype of target variable must be same.
- A pointer can store only address, it cannot store normal values.
- A normal variable can store only normal value it cannot store address.
- We cannot perform any arithmetic & relational operations on pointer.
- A pointer is also a memory-block.  
∴ A pointer also has address.
- A pointer of any datatype occupies 2 bytes (in Turbo C) & 4 bytes (in GCC / MinGW).
- Declaration of \* indicates "pointer".
- Statement of \* indicates "value of address" operator

\* - "value at address" operator  
"dereferencing" operator

(70) #include <stdio.h>  
#include <conio.h>  
void main()  
{

int a = 5;

int \*p;

p = &a;

clrscr();

printf("%d", a);  $\leftarrow$  ⑤

printf("%u", &a);  $\leftarrow$  14295

printf("%u", p);  $\leftarrow$  14295

printf("%d", \*p);

printf("%u", &p);  $\leftarrow$  1559

getch();

}

(71) #include <stdio.h>  
#include <conio.h>  
void main()  
{

int a, b, c;

int \*p1, \*p2, \*p3;

p1 = &a;

p2 = &b;

p3 = &c;

printf("Enter 2 values\n");

scanf("%d %d", &a, &b);

$c = a + b;$

or

$c = (*p_1) + (*p_2);$

or

$*p_3 = a + b;$

or

$*p_3 = (*p_1) + (*p_2);$

`printf("Addition is %d", c);`

or

`printf("Addition is %d", *p_3);`

`getch();`

}

- \* Ultimately a pointer is variable.  
i.e. we can over-write address in it.

(72) `void main()`

{

`int a=5, b=10, int c=15;`

`int *p;`

—

`p=&a;`

—

`p=&b;`

—

`p=&c;`

—

}

- \* A pointer can store the address of variable, with which it's datatype matches.

(73)

{

```
int a=5;
float b=9.2;
int *p;
```

$p = \&a$ ;  $\leftarrow$  valid

$p = \&b$ ;  $\leftarrow$  Invalid

=

}

- \* If required, then we can also declare constant pointers by using keyword "const".

```
int a=5;
const int *p = &a;
```

- \* ज्यातीं value दिलायला ज्यातीं call by value होती वाचे.

- \* ज्यातीं value दिलायला ज्यातीं call by reference होती वाचे.

74 void main()

5

```
int a,b,c;
```

Chorus:

```
printf("Enter 2 values\n");
```

```
scanf("%d %d", &a, &b);
```

add (a,b,c);

T - K

Passed values of  
a and b as A.o.P.

passed

address of

as A. P.

```
printf("Addition is %d", c);
```

geteh ();

2

void add (int x, int y, int \*p)

1

$$* p = x + y;$$

1

45

```
void main()
```

5

int a,b; ad, sub mult;

float div;

classer ( ) ;

```
printf("Enter 2 values\n");
```

```
scanf("%d %d", &a, &b);
```

operations(a,b,&ad,&sub,&mult,&div);

```

printf("Addition is %.d", add);
printf("Subtraction is %.d", sub);
printf("Multiplication is %.d", mult);
printf("Division is %.f", div);
getch();
}

void operation (int x, int y, int *P1, int *P2, int *P3,
float *P4)
{

```

$$\begin{aligned}
*P1 &= x + y; \\
*P2 &= x - y; \\
*P3 &= x * y; \\
*P4 &= x / (float)y;
\end{aligned}$$

\* Difference between call by value & call by reference.

<u>Call by value</u>	<u>call by reference</u>
1. We pass value of variables as Actual parameter.	1. We pass address of variable as Actual parameter.
2. We declared normal variables as formal parameter.	2. We declare pointers as formal parameter.
3. In this mechanism, a function can return only one value.	3. This mechanism, a function can return multiple values, indirectly.

76 W.A.P to swap values of two integers by using call by reference.

```
#include < stdio.h >
#include < conio.h >
void swap (int *P1, int *P2)

void main()
{
    int a, b;

    printf("Enter 2 values\n");
    scanf("%d %d", &a, &b);

    printf("Before swapping a = %d and
           b = %d", a, b);

    swap (&a, &b);

    printf("After swapping a = %d and
           b = %d", a, b);

    getch();
}

void swap (int *P1, int *P2)
{
    int temp;
    temp = *P1;
    *P1 = *P2;
    *P2 = temp;
}
```

- \* To user-defined header file making
  - 1) Firstly avoid to include void main
  - 2) Make user-defined header file in turbo c  
(include → save it)
  - 3) Call it header file whenever required.

### \* Structure

- \* When we want to work on any real world entity (that has attributes) then prefer structure.
  - i. A structure is a set of attributes of different datatypes.

\* The declare structure, general syntax is

Struct is

a      → struct      structure-name

Keyword      {

attribute - 1      declaration ;

attribute - 2      declaration ;

:

attribute - n      declaration ;

?;

- \* A structure contains only declaration of attributes.
- \* Executable statements & logical blocks are not allowed in structure.
- \* We cannot initialize attributes.
- \* We can declare any number of attributes in a structure.
- \* A string is set of characters.
- \* We can prefer string to store any alpha numeric OR large numeric OR only alphabets.

for ex:- Name Pan No. Account No.  
address Mobile No. Aadhar No.

- \* A string can store almost everything of any range / length.
- \* But we cannot perform any arithmetic & relational operations on string.

- \* To declare string:-

```
char str_name [length];
```

for ex:-

```
char f-company [20];
```

```
char stud-mob [15];
```

```
char stud-adhar [75];
```

```
char stud-name [30];
```

- \* To prefer use fflush() before string input.

- \* To perform input - output operations on string, we have format specifier %s.

```
void main()
{
    char st-name [25];
    printf("Enter student name");
    scanf ("%s", &st-name);
    printf("Student is : %s", st-name);
    getch();
}
```

- \* Declare a structure / related to attributes of fan

```
struct fan
```

```
{
```

```
    char company[15];
```

```

char model [10];
int price [100];
char type [5];
int no. of blades [3];
};

```

- \* Declare a structure related to attributes of a library book.

```

struct library book
{
    int price [100];
    int book no [20];
    char book name [5];
    char book publication [13];
    long book date
};

```

- \* Declare a structure related to attributes of a vehicle.

```

struct vehicle
{
    int price [10];
    int model no [20];
    float average [50.55];
    char v-name [8];
};

```

- \* Declare a structure related to attributes of a bank account

```
struct bank account
```

{

```
    int balance [5000];
```

```
    string A no [200500300501];
```

```
    int IFSC code [20105];
```

```
    char branch no [18];
```

};

- \* A structure is like blue print of attributes.
- \* A structure does not occupy any runtime memory.  
like cannot store any data / value in structure
- \* we have to think about allocation memory to structure  
∴ we declare objects of structure.
- \* An object is copy of structure with memory.
- \* To declare an object, general syntax is :

```
struct structure-name - object-name;
```

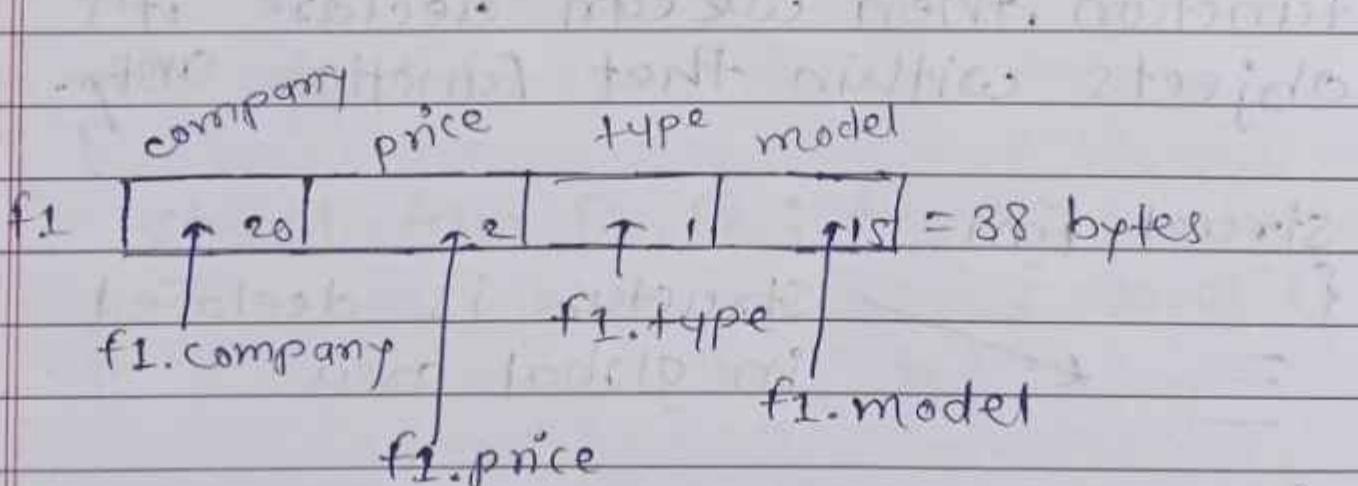
- \* Structure name and object name is identifier.  
∴ we have to follow same 5 rules of identifier for them.

- \* we can declare any numbers of objects of a structure.
- \* and every object will have individual memory and address.
- \* we store values / data in object, not in structure.
- \* An object occupies runtime memory. An object also has address.
- \* When an object is declared, the attributes of structure will create individual blocks.
- \* No. of blocks = No. of attributes.
- \* These blocks in object is called as "Instance".
- \* An object is also called as "set of instances".
- \* we store data in instances.

- \* To identify an instance : obj-name . attr-name
- \* we can declare any numbers of objects of a structure.
- \* and each object will occupy same size of a memory but they will have individual memory locations.

`printf("%u", &f1);` ← prints address of object f1

`printf("%u", &f2);` ← prints address of object f2



- \* we cannot perform any arithmetic & relational operation on entised object. But we can perform arithmetic & relational operation on instances.
- \* we can declare structure
  - as global variable
  - as local variable
- \* If we declare structure in global area then it's scope/availability is everywhere in the program.  
∴ we can declare it's objects in every function.
- \* If we declare structure in local area of function, then we can declare it's objects within that function only.

① struct fun ...  
{  
     $\leftarrow$  Structure is declared  
    in global area  
};  
void main()  
{  
     $\leftarrow$

object declaration is valid

```
struct fan f1; // object declaration is valid
```

void check()

```
{
```

=

```
struct fan f2; // object declaration is valid
```

=

```
}
```

② void main()

```
{
```

struct fan

```
{
```

= ← structure is declared in local area of main()

```
};
```

struct fan f1; ← object declaration is valid

```
=
```

```
}
```

void check()

```
{
```

struct fan f1, f2; ← object declaration is invalid

```
=
```

```
}
```

77

Declare a structure related to attributes of a fan. Declare two objects of this structure to store details of two fan. Perform accept & display operation on instances of both object.

```
#include <stdio.h>
#include <conio.h>
struct fan
{
    char company[20];
    int price;
    char model[15];
    int no_of_blades;
};

void main()
{
    clrscr();
    struct fan f1,f2;

    printf("Accepting details of first fan\n");
    printf("Enter company name : ");
    scanf("%s", &f1.company);
    printf("Enter price : ");
    scanf("%d", &f1.price);
    printf("Enter Model name : ");
    scanf("%s", &f1.model);
    printf("Enter count of blades : ");
    scanf("%d", &f1.no_of_blades);
```

```
printf("Accepting details of Second fan\n");
printf("Enter Company name : ");
scanf("%s", &f2.company);
printf("Enter price : ");
scanf("%d", &f2.price);
printf("Enter Model name : ");
scanf("%s", &f2.model);
printf("Enter count of blades : ");
scanf("%d", &f2.no-of-blades);
```

```
printf("Details of First fan\n");
printf("Company : %s\n", f1.company);
printf("Model : %s\n", f1.model);
printf("Price : %.d\n", f1.price);
printf("Number of blades : %.d\n", f1.no-
of-blades);
```

```
printf("Details of Second fan\n");
printf("Company : %s\n", f2.company);
printf("Model : %s\n", f2.model);
printf("Price : %.d\n", f2.price);
printf("Number of blades : %.d\n", f2.
no-of-blades);
```

```
getch();
```

- (78) Declare a structure related to attributes of an employee - such as name, salary, post. Declare two objects of these structures to accept details of two employees and print the name of employee who has higher salary:

```
#include <stdio.h>
#include <conio.h>
struct employee
{
    char name_e[20];
    char post[15];
    long salary;
};

void main()
{
    clrscr();
    struct employee e1, e2;
    printf("Details of First employee\n");
    printf("Name of the employee : ");
    scanf("%s", &e1.name_e);
    printf("Post of the employee : ");
    scanf("%s", &e1.post);
    printf("Salary of the employee : ");
    scanf("%d", &e1.salary);
```

```
printf("Details of second employee\n");
fflush;
printf("Name of the employee : \n");
scanf("%s", &e2.name);
fflush;
printf("Post of the employee : \n");
scanf("%s", &e2.post);
fflush;
printf("Salary of the employee : ");
scanf("%d", &e2.salary);

if(e1.salary > e2.salary)
{
    printf("%s have higher salary... ", e1.name);
}
else
{
    printf("%s have higher salary... ", e2.name);
}

getch();
```

79

Declare a structure selected to attributes of a student such as name, average marks, gender. Declare three objects of these structure to store details of three students. Accept details of 3 students & print name of topper.

```
#include <stdio.h>
#include <conio.h>
struct student {
    char name[50];
    float avgmarks;
    char gender;
};

void main()
{
    clrscr();
    struct student s1, s2, s3;

    printf("Enter data for the first student :\n");
    printf("Name : ");
    scanf("%s", s1.name);
    printf("Average marks : ");
    scanf("%f", &s1.avgmarks);
    printf("Gender (M/F) : ");
    scanf("%c", &s1.gender);

    printf("\nEnter data for the second student\n");
    printf("Name : ");
}
```

```
scanf("%s", s2.name);
printf("Average marks : ");
scanf("%f", &s2.avgmarks);
printf("Gender (M/F) : ");
scanf("%c", &s2.gender);
```

```
printf("Enter data for the third student :\n");
printf("Name : ");
scanf("%s", s3.name);
printf("Average marks : ");
scanf("%f", &s3.avgmarks);
printf("Gender (M/F) : ");
scanf("%c", &s3.gender);
```

```
struct student topper;
if (s1.avgmarks > s2.avgmarks && s1.avgmarks
    > s3.avgmarks)
{
    topper = s1;
}
else if (s2.avgmarks > s3.avgmarks)
{
    topper = s2;
}
else
{
    topper = s3;
```

```
printf("%s is the Topper ... \n", topper.name);
getch();
```

- \* Address is per byte
- \* Pointer stores/holds base address of any memory block.

### \* Pointer to object

working on : address of object

From above examples, we know that an object is ultimately a memory block & therefore structure object also has address, which is called as base address.

struct fan f1;

printf("Address is %u", f1);

printf("size of object f1 is %d", sizeof(f1));

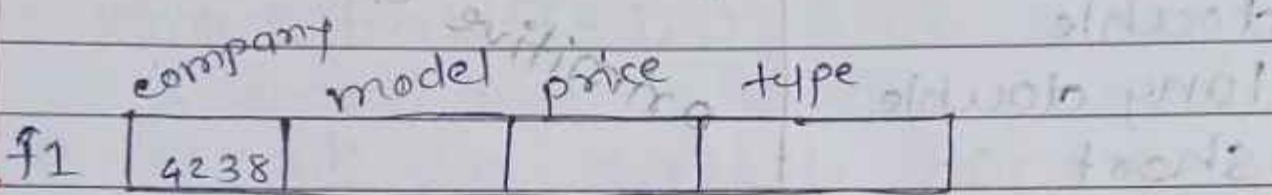
	company	model	price	type	
f1	20	10	2	57	= 33 bytes
	4238	4258	4267	4269	
	to	to	to		
	4257	4267	4268		

```
struct fcm f1;
struct fcm *P;
```

$$P = \&f1;$$

↓

- \* The pointer P will store Base address of object f1.
- \* Datatype of structure object is structure's name.



~~\* P  
4238~~

- \* If an object is referred by pointer, then instances of object are also identified by pointer & Arrow operator (→).

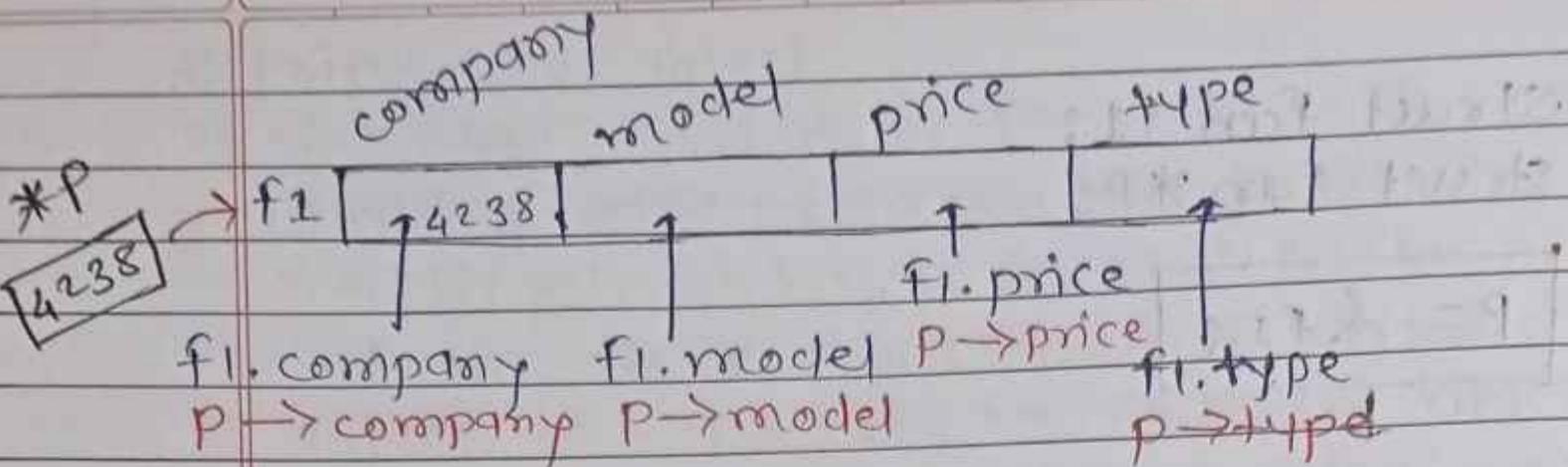
Only in case  
of structure's  
object

object के साथ Dot operator (.)  
pointer के साथ Arrow operator  
(→)

\* Objects datatype is structure's name  
Therefore a structure is user defined

Date  
Page

datatype.



\* Built-in datatypes

int  
float  
char  
double  
long double  
short  
long

primary or primitive datatypes

\* Structure's name } secondary datatypes  
union's name } user-defined datatypes

non-primitive datatypes

## Array:-

- \* When we want to perform same operation on multiple values, then prefer array.
- \* An array is set of similar/homogeneous datatype.
- \* To declare an array:  
**(General syntax of an array):-**

**datatype name[length];**

**exp:-**

int a[5];

float b[10];

struct sfm f[5];

char a[10];

2	4	company	price	model	type
2	4				
2	4				
2	4				
2	4				
2	4				
2	4				
2	4				
2	4				
2	4				
10					
4					
4					
4					
4					
4					
4					
4					
4					
4					
4					
4					
40					

- \* If we declare individual variables, then their memory is not serial.  
i.e. They might get generated at different locations.
  - \* If we declare an array, then array-block will get serial memory.  
∴ they have serial address.
  - \* An array obtains runtime memory.  
∴ we store data in array.  
∴ No mechanism like declaring objects of array.
- ~~Types~~ {
- \* While declaring, array length must be non-zero, positive and integer.
  - \* We cannot declare variable length array.
  - \* The [ ] in array is called as "Array indexing" operator.
  - \* Each block in array is provided with a serial-number.
  - \* This serial-number is technically called as "index-number";  
or  
"subscript".
  - \* This indexing starts from 0 (zero)

$a[0] = 25;$

$a[1] = -30;$

$a[2] = a[0] * 3;$

`scanf("%d", &a[3]);`

`int a[5];`

0	$\rightarrow a$	$\leftarrow a[0]$
1		$\leftarrow a[1]$
2		$\leftarrow a[2]$
3		$\leftarrow a[3]$
4		$\leftarrow a[4]$

- (80) WAP to accept 10 elements of an array and display them.

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

```
void main()
```

```
{
```

```
    int a[10], i;
```

```
    clrscr();
```

```
    printf("Enter 10 elements\n");
```

```
    for (i=0; i<=9; i++)
```

```
{
```

```
        scanf("%d", &a[i]);
```

```
}
```

(80)

```
    printf("Array elements are :\n");
```

```
    for (i=0; i<=9; i++)
```

```
{
```

```
        printf("%d\n", a[i]);
```

```
}
```

```
getch();
```

```
}
```

- (81) WAP to accept 10 elements of an array and display them in reverse order.

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int a[10], i;
    clrscr();
    printf("Enter 10 elements\n");
    for (i=0; i<=9; i++)
    {
        scanf("%d", &a[i]);
    }
    printf("Array in reverse order are :\n");
    for (i=9; i>=0; i--)
    {
        printf("%d\n", a[i]);
    }
    getch();
}
```

- (82) WAP to accept 10 elements of an array and print the elements present at odd index only.

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int a[10], i;
    clrscr();
    printf("Enter 10 elements\n");
    for (i=0; i<=9; i++)
    {
        scanf("%d", &a[i]);
    }
    printf("Array elements in odd index are:\n");
    for (i=1; i<=9; i=i+2)
    {
        printf("%d\n", a[i]);
    }
    getch();
}
```

- (83) WAP to accept 10 elements of an array and print even elements of it.

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int a[10], i;
    clrscr();
    printf("Enter 10 elements\n");
}
```

```
for(i=0; i<=9 ; i++)
{
    scanf("%d", &a[i]);
}
printf("Array elements in even are : \n");
for(i=0; i<=9 ; i++)
{
    if(a[i] % 2 == 0)
    {
        printf("%d\n", a[i]);
    }
}
getch();
}
```

- 84) WAP to accept 10 elements of an array and find sum of all elements.

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int i, a[10], sum = 0;
    clrscr();
    printf("Enter 10 elements \n");
    for(i=0; i<=9; i++)
    {
        scanf("%d", &a[i]);
    }
}
```

```
for(i=0; i<=9; i++)  
{  
    sum = sum + a[i];  
}  
printf("Sum of all array elements are %d",  
      sum);  
 getch();  
}
```

- (85) WAP to accept 20 elements of an array & print total count of EVEN elements & ODD elements in each.

```
#include <stdio.h>  
#include <conio.h>  
void main()  
{  
    int a[20]; ocount=0, ecount=0, i;  
    clrscr();  
    printf("Enter 20 elements\n");  
    for(i=0; i<=19; i++)  
    {  
        scanf("%d", &a[i]);  
    }  
    for(i=0; i<=19; i++)  
    {  
        if(a[i] % 2 == 0)  
        {  
            ecount++;  
        }  
        else
```

```
{  
    ocount++;  
}  
}  
printf("Odd number count = %d", ocount);  
printf("Even number count = %d", ecount);  
getch();  
}
```

- (86) WAP to accept 20 numbers of an array and print Yes if all numbers are even elements, otherwise print No.

```
#include <stdio.h>  
#include <conio.h>  
void main()  
{  
    int count=0, i, a[20];  
    clrscr();  
    printf("Enter 20 elements : \n");  
    for (i=0; i<19; i++)  
    {  
        scanf("%d", &a[i]);  
        if (a[i] % 2 == 0)  
        {  
            count++;  
        }  
    }  
    if (count == 19)
```

```
    printf("YES");  
}  
else  
{  
    printf("NO");  
}  
getch();  
}
```

- (87) WAP to accept 20 elements of an array and print individual sum of even elements & odd elements in each.

```
#include<stdio.h>  
#include<conio.h>  
void main()  
{  
    int a[20], sumeven=0, sumodd=0, i;  
    clrscr();  
    printf("Enter 20 elements:\n");  
    for(i=0; i<=19; i++)  
    {  
        scanf("%d", &a[i]);  
    }  
    for(i=0; i<=19; i++)  
    {  
        if(a[i] % 2 == 0)  
        {  
            sumeven = sumeven + a[i];  
        }  
        else
```

```

    {
        sumodd = sumodd + a[i];
    }
}

printf("sum of all even elements is %d\n",
       sumeven);
printf("sum of all odd elements is %d\n",
       sumodd);

getch();
}

```

- (88) WAP to accept 10 elements of an array & copy them into another array.

```

#include < stdio.h >
#include < conio.h >
void main()
{
    int a[10], b[10], i;
    clrscr();
    printf("Enter 10 elements : \n");
    for(i=0; i<=9; i++)
    {
        scanf("%d", &a[i]);
    }
    for(i=0; i<=9; i++)
    {
        b[i] = a[i];
    }
    printf("Elements of copied array are : \n");
}

```

```

    for(i=0; i<=9; i++)
    {
        printf("%d\n", b[i]);
    }
    getch();
}

```

- (89) WAP to copy elements of one array into another array in reverse order.

```

#include <stdio.h>
#include <conio.h>
void main()
{
    int a[10], b[10], i, j;
    clrscr();
    printf("Enter 10 elements:\n");
    for(i=0; i<=9; i++)
    {
        scanf("%d", &a[i]);
        b[j] = a[i];
        j--;
    }
    printf("The elements in reverse order
           are :\n");
    for(i=9; i>=0; i--)
    {
        printf("%d\n", a[i]);
    }
    getch();
}

```

(90) WAP to add two arrays into third array.

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int a[5], b[5], i, result[10];
    clrscr();
    printf("Enter 5 elements for First array: \n");
    for(i=0; i<=4; i++)
    {
        scanf("%d", &a[i]);
    }
    printf("Enter 5 elements for Second array: \n");
    for(i=0; i<=4; i++)
    {
        scanf("%d", &b[i]);
    }
    for (i=0; i<=4; i++)
    {
        result[i] = a[i] + b[i];
    }
    printf("Result after addition: \n");
    for(i=0; i<=4; i++)
    {
        printf("%d\t", result[i]);
    }
    getch();
}
```

(91)

WAP to merge elements of two array into third array.

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int a[5], b[5], i, j, merge[10];
    clrscr();
    printf("Enter 5 elements : \n");
    for(i=0; i<5; i++)
    {
        scanf("%d", &a[i]);
        merge[i] = a[i];
    }
    j = i;
    printf("Enter 5 elements : \n");
    for(i=0; i<5; i++)
    {
        scanf("%d", &b[i]);
        merge[j] = b[i];
        j++;
    }
    printf("The New array after Merge is : \n");
    for(i=0; i<10; i++)
    {
        printf("%d\n", merge[i]);
    }
    getch();
}
```

Q2 WAP to accept 10 elements of an 'array' and swap the elements present at 'even indexes' with the elements present at exact next odd indexes.

- 93) Write a C program to check whether desired element is present in array or not.

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int a[10], num, i, flag = 1;
    clrscr();
    printf("Enter 10 elements : \n");
    for(i=0; i<=9; i++)
    {
        scanf("%d", &a[i]);
    }
    printf("Enter element to search : ");
    scanf("%d", &num);
    for(i=0; i<=9; i++)
    {
        if(a[i] == num)
        {
            flag = 2;
            break;
        }
    }
    if(flag == 1)
    {
        printf("Element not present");
    }
    else if(flag == 2)
    {
        printf("Element present");
    }
}
```

```
    printf("Element is present");  
}
```

- 94) WAP to accept array element from the user and find largest element of that array.

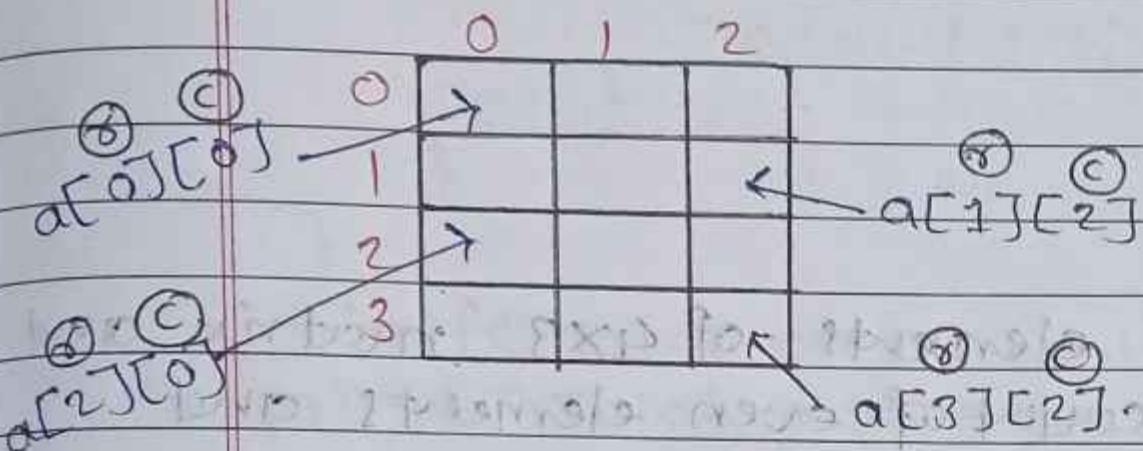
```
#include<stdio.h>  
#include<conio.h>  
void main()  
{  
    int a[10], i, lar;  
    clrscr();  
    printf("Enter 10 elements:\n");  
    for(i=0; i<=9; i++)  
    {  
        scanf("%d", &a[i]);  
    }  
    lar = a[0];  
    for(i=1; i<=9; i++)  
    {  
        if(a[i] > lar)  
        {  
            lar = a[i];  
        }  
    }  
    printf("Largest element is %d", lar);  
    getch();  
}
```

## \* Two-dimensional array:

- To declare two-dimensional array:-  
(general syntax is)

	(row)	(column)
datatype	name	[length][length];

exp:- int a[4][3]



- Q5) WAP to accept elements of  $4 \times 3$  array and display them in matrix form.

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

```
#include <conio.h>
```

```
void main()
```

```
{
```

```
int a[4][3], i, j;
```

```
clrscr();
```

```
printf("Enter elements for  $4 \times 3$  matrix:\n");
```

```
for(i=0; i<3; i++)
```

```
{
```

```
    for(j=0; j<2; j++)
```

```
{
```

```
    scanf("%d", &a[i][j]);  
}  
}  
printf("Elements are\n");  
for(i=0; i<3; i++)  
{  
    for(j=0; j<2; j++)  
    {  
        printf("%d\t", a[i][j]);  
    }  
    printf("\n");  
}  
getch();  
}
```

- Q6 WAP to accept elements of  $4 \times 3$  matrix and print total count of even elements and odd elements.

```
#include <stdio.h>  
#include <conio.h>  
void main()  
{  
    int a[4][3], i, j, even=0, odd=0;  
    clrscr();  
    printf("Enter elements for  $4 \times 3$  matrix:\n");  
    for(i=0; i<3; i++)  
    {  
        for(j=0; j<2; j++)  
        {  
            scanf("%d", &a[i][j]);  
        }  
    }  
}
```

```

for(i=0; i<=3; i++)
{
    for (j=0; j<=2; j++)
    {
        if (a[i][j] % 2 == 0)
        {
            ecount++;
        }
        else
        {
            ocount++;
        }
    }
}

printf("Total even count are %d\n", ecount);
printf("Total odd count are %d\n", ocount);
getch();
}

```

- (97) WAP to accept elements of  $5 \times 5$  matrix and print sum of all array elements.

```

#include <stdio.h>
#include <conio.h>
void main()
{
    int a[5][5], i, j, sum=0;
    clrscr();
    printf("Enter elements for 5x5 matrix:\n");

```

```

for (i=0; i<=4; i++)
{
    for (j=0; j<=4; j++)
    {
        scanf("%d", &a[i][j]);
    }
}
printf("sum of all array elements is : ");
for (i=0; i<=4; i++)
{
    for (j=0; j<=4; j++)
    {
        sum = sum + a[i][j];
    }
}
printf("%d", sum);
getch();

```

~~Q8~~ WAP to accept of  $5 \times 5$  matrix & print it's diagonal elements only.

```

#include <stdio.h>
#include <conio.h>
void main()
{
    int a[5][5], i, j;
    clrscr();
    printf("Enter elements for  $5 \times 5$  matrix : \n");
    for (i=0; i<=4; i++)
    {

```

```
for(j=0; j<=4; j++)
{
    scanf("%d", &a[i][j]);
}
for(i=0; i<=4; i++)
{
    for(j=0; j<=4; j++)
    {
        if (i == j)
        {
            printf("%d\n", a[i][j]);
        }
    }
}
getch();
```

- 99) WAP to accept elements of  $5 \times 5$  matrix & swap the elements row(0) & row(4).

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int a[5][5], i, j, temp = 0;
    clrscr();
    printf("Enter elements for  $5 \times 5$  matrix:\n");
    for(i=0; i<=4; i++)
    {
        for(j=0; j<=4; j++)
        {
```

```

        scanf("%d", &a[i][j]);
    }

}

for(j=0; i<4; i++)
{
    temp = a[0][i];
    a[0][i] = a[4][i];
    a[4][i] = temp;

}

printf("%d", temp);
getch();
}

```

100) WAP to add two matrix.

```

#include < stdio.h>
#include < conio.h>
void main()
{
    int a[3][3], b[3][3], sum[6][6], i, j;
    clrscr();
    printf("Enter elements of 1st matrix : \n");
    for(i=0; i<2; i++)
    {
        for(j=0; j<2; j++)
        {
            scanf("%d", &a[i][j]);
        }
    }

```

```
printf("Enter elements for 2 matrix : \n");
for(i=0; i<=2; i++)
{
    for(j=0; j<=2; j++)
        scanf("%d", &b[i][j]);
}
for(i=0; i<=2; i++)
{
    for(j=0; j<=2; j++)
        sum[i][j] = a[i][j] + b[i][j];
}
printf("Sum of matrix : \n");
for(i=0; i<=2; i++)
{
    for(j=0; j<=2; j++)
        printf("%d\t", sum[i][j]);
    printf("\n");
}
getch();
```

①

WAP to accept 9 elements for one dimensional array & feed it into a  $3 \times 3$  matrix row wise.

```
#include < stdio.h >
#include < conio.h >
void main()
{
    int a[3][3], i, j;
    clrscr();
    printf("Enter 9 elements for  $3 \times 3$  matrix:\n");
    for(i=0; i<=2; i++)
    {
        for(j=0; j<=2; j++)
        {
            scanf("%d", &a[i][j]);
        }
    }
    printf("The  $3 \times 3$  matrix is :\n");
    for(i=0; i<=2; i++)
    {
        for(j=0; j<=2; j++)
        {
            printf("%d", a[i][j]);
        }
        printf("\n");
    }
    getch();
}
```

- ② WAP to fill entised  $5 \times 5$  matrix with digit with 1.

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int a[5][5], i, j;
    clrscr();
    for(i=0; i<=4; i++)
    {
        for(j=0; j<=4; j++)
        {
            printf("1\t");
        }
        printf("\n");
    }
    getch();
}
```

- ③ WAP to accept elements for two  $5 \times 5$  matrix and print YES if both matrix are exact same (identical), otherwise print NO.

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int a[5][5], b[5][5], i, j, flag = 1;
    clrscr();
```

```
printf("Enter elements for 1 array:\n");
for(i=0; i<=4; i++)
{
    for(j=0; j<=4; j++)
    {
        scanf("%d", &a[i][j]);
    }
}
for(i=0; i<=4; i++)
{
    for(j=0; j<=4; j++)
    {
        printf("%d", a[i][j]);
    }
    printf("\n");
}

printf("Enter elements for 2 array:\n");
for(i=0; i<=4; i++)
{
    for(j=0; j<=4; j++)
    {
        scanf("%d", &b[i][j]);
    }
}
for(i=0; i<=4; i++)
{
    for(j=0; j<=4; j++)
    {
        printf("%d", b[i][j]);
    }
    printf("\n");
}
```

```
printf("Both arrays are equal : ");
for (i=0; i<=4; i++)
{
    for (j=0; j<=4; j++)
    {
        if (a[i][j] == b[i][j])
        {
            flag = 2;
            break;
        }
    }
}
if (flag == 2)
{
    printf("YES");
}
else if (flag == 1)
{
    printf("NO");
}
getch();
```

## \* Initializing Array:

Similar to initializing a variable, we can also initialize an array.

### A. Initializing one dimensional array:-

We initialize a one dimensional array with set of elements within pair of {}.

General syntax is:

datatype name[length] = {list of elements}; for

exp:-

① int a[5] = {41, 28, 38, 67, 82};

a	0	41
1	28	
2	38	
3	67	
4	82	

② char nm[5] = {'p', 'a', 't', 'w', 'n'};

nm	0	'p'
1	'a'	
2	't'	
3	'w'	
4	'n'	

③ float b[4] = {21.67, 38.51, 78.82, 23.29, 18.45};

b

21.67	0
38.51	1
78.82	2
23.29	3
18.45	4

for

\* Remember that, while initializing one dimensional array, it is optional to specify length of array. The interpreter will automatically decide length based on initialized elements.

\* Remember that, while initializing an array, if we initialize few/less numbers of elements as compared to declared length, then remaining elements will be 0 (zero).

for ex:- ① int a[8] = {51, 45, 28, 71, 35} .

51	0
45	1
28	2
71	3
35	4
.....	

### B. Initializing Two dimensional array:-

A Two dimensional array can be initialize within a nested linner braces of {} and we initialize it row wise.

for ex:- int a[4][3] = { {10, 20, 30}, {40, 50, 60}, {70, 80, 90}, {100, 110, 120} } ;

(row-0)	(row-1)	(row-2)
{10, 20, 30}	{40, 50, 60}	{70, 80, 90}
{100, 110, 120}		
(row-3)		

- If we initialize a two dimensional array, then specifying length is optional.
- While initializing two dimensional array, if initialized elements are fewer/lesser than declared length, then remaining elements will be 0. (zero).

#### ④ Matrix multiplication

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

```
#include <conio.h>
```

```
void main()
```

```
{
```

```
int a[3][3], b[3][3], c[3][3];
```

```
int i, j, k, sum;
```

```
clrscr();
```

```
printf("Enter elements for a : \n");
```

```
scanf("%d", &a[i][j]);
```

```
printf("Enter elements for b : \n");
```

```
scanf("%d", &b[i][j]);
```

```
for(i=0; i<=2; i++)
```

```
{
```

```
    for(j=0; j<=2; j++)
```

```
{
```

```
        sum=0;
```

```
        for(k=0; k<=2; k++)
```

```
{
```

```
            sum = sum + (a[i][k] * b[k][j]);
```

```
}
```

```
c[i][j]=sum;
```

```
}
```

printf("Elements of resulting matrix is\n");

for(i=0; i<=2; i++)

{

for(j=0; j<=2; j++)

{

printf("%d\t", c[i][j]);

{

printf("\n");

getch();

{

## \*String\*

- A string is set of characters.

~~or~~  
A string is character array.

exp:- char sname[25];  
char company[20];

- To input-output string, we use %s.
- We cannot perform any arithmetic and relational operations on string.
- The %s can read/input one-word string only.
- this gets () is in <stdio.h>.
- To accept multi-word string, we have gets () .
- To use gets () (General syntax is):

```
gets (string-name);
```

exp:- gets (sname);  
gets (scoll);

- Here, string-name is passed as Actual parameter.
- An array by-default performs call-by reference.
- The %s can print multi-word string.

(5) WAP to accept roll no, name & college name of a student & display it with proper message.

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

```
#include<conio.h>
```

```
void main()
```

```
{
```

```
char rollno[20];
```

```
char sname[25];
```

```
char scoll[30];
```

```
clrscr();
```

```
printf("Enter student roll number:");  
gets(rollno);
```

```
printf("Enter student name:");  
gets(sname);
```

```
printf("Enter student college name:");  
gets(scoll);
```

```
printf("In Roll number : %s\n", rollno);
```

```
printf("In Student name : %s\n", sname);
```

```
printf("In College name : %s\n", scoll);
```

```
getch();
```

```
}
```

- \* When we input a string, each character of input string will be placed at index-number in string.
  - \* At the end of string, interpreter will automatically add '\0' to indicate termination.
  - \* This '\0' is called as 'null' character.
  - \* In other words, every string ends with '\0'.
  - \* Every string operations will be performed upto '\0' only.

P	r	a	n	a	v	o					
0	1	2	3	4.	5	6	7	-	-	-	24

⑥ #include <stdio.h>

```
#include <conio.h>
```

#include <string.h>

void main()

$$\{ \text{GOT10D} - 5 \}$$

```
char sname[30];
```

int i;

classical;

```
printf("Enter student name:");
```

gets(sname);

i = 0 ;

```
while(sname[i] != '\0')
```

7

```
printf("%s\n", sname[i]);
```

i++;

7

getch();

- \* It is also possible to pass an array as Actual parameter.
- \* But, while passing an array, as Actual parameter, never use [ ].
  - ∴ Pass only name of array.
- \* But, while receiving an array as formal parameter, we must use [ ]. But don't provide size / length.
- \* An array, by-default performs call-by-reference.
- \* whereas, a normal variable and structure objects by-default performs call-by-value.
- \* C/C++ functions cannot return array.

### \* String Library Functions:

To perform general purpose string operation, we have some library functions in <string.h> they are

- `strlen()`
- `strrev()`
- `strupr()`
- `strlwr()`
- `strcmp()`
- `strcat()`
- `strcpy()`

\* In C/C++ strings are mutable.

\* In Java/Python strings are immutable.

Page

### ① strlen () :-

This function returns length of specified string (). To use this function, general syntax is:-

[ int variable = strlen (string-name); ]

for eg:-

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
void main()
{
    char str[30];
    int x;
    clrscr();
    printf("Enter any string : ");
    gets(str);
    x = strlen(str);
    printf("Length of string is : %d", x);
    getch();
}
```

## (2) Strrev() :-

This function permanently reverses parameters string to use this function.

\* General syntax is:-

Strrev(string-name);

This function reverses parameter string, but will not return anything.

for eg:-

```
#include<stdio.h>
#include <conio.h>
#include <string.h>
void main()
{
    char str[50];
    clrscr();
    printf("Enter any string : ");
    gets(str);
    printf("Original String : %s", str);
    strrev(str);
    printf("Updated String : %s", str);
    getch();
}
```

### ③ `strupr()`:

This function permanently converts parameter string to uppercase.

\* General syntax is :

`strupr(string-name);`

In this function, the uppercase letter, digit & special symbols remains unchanged. Only lowercase letters get changed into uppercase.

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

void main()
{
    char str[50];
    clrscr();
    printf("Enter any string : ");
    gets(str);
    printf("Original String : %s", str);
   strupr(str);
    printf("Updated String : %s ", str);
    getch();
}
```

#### ④ Strlwr():-

This function permanently converts parameter string to lowercase.

\* General syntax is:

`Strlwr (String-name);`

In this function, the lowercase letters, digit & special symbols remains unchanged. Only uppercase letters gets changed into lowercase.

for eg:-

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
void main()
{
    char str[50];
    clrscr();
    printf("Enter any string :");
    gets(str);
    printf("Original String : %s", str);
    Strlwr(str);
    printf("Updated String : %s", str);
    getch();
}
```

## (5) strcmp () :-

This function compares two strings based on their ASCII values! To use this function (Lexicology).

\* General syntax is :-

int variable = strcmp (string1-name, string2-name);

This function compares string based on their ASCII values, & returns zero (0) if both strings are totally equal, otherwise returns any non-zero value.

for eg:-

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
    int x;
    char str1[50], str2[50];
    clrscr();
    printf("Enter first string : ");
    gets(str1);
    printf("Enter second string : ");
    gets(str2);
```

```
x = strcmp (str1, str2);  
if (x == 0)  
{ printf("Both string are Equal\n");  
}  
else  
{ printf("Both string are Unequal ");  
}  
getch();  
}
```



(6)

strcpy () :-

This function is to copy one string into another string. To use this function.

It always copies string from source to target.

\* General syntax is :-

strcpy = (target\_string\_name, source\_string\_name);

for eg:-

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
void main()
{
    char str1[50], str2[50];
    clrscr();
    printf("Enter first string : ");
    gets(str1);
    strcpy(str2, str1);
    printf("Copied string : %s ", str2);
    getch();
}
```

(7) **Strcat () :-**

This function concatenate / join two strings . To use this function -

\* General syntax is:-

```
strcat ( string 1-name , string 2-name );
```

This function concatenates string 2 at the end of string 1, it updates string 1 permanently & string 2 remains unchanged.

for eg:- Str1 = "Jerry"  
Str2 = "Mouse"

```
strcat (str1, str2);
```



Str1 = "JerryMouse"  
Str2 = "Mouse"

```
Str cat (str2, str1);
```



Str1 = "Jerry"  
Str2 = "Mouse Jerry"

17.2.1992 ①

most 1 strawberry plant 1991

most 2 strawberry plants 1991

most 3 strawberry plants 1991 木

most 4 strawberry plants 1991 1992

most 5 strawberry plants 1991 1992

most 6 strawberry plants 1991 1992

most 7 strawberry plants 1991 1992

"most" = 1992

"most" = 1992

1992, 1992 1992

"most" = 1992

"most" = 1992

(1992, 1992) 1992

4

"most" = 1992

"most" = 1992