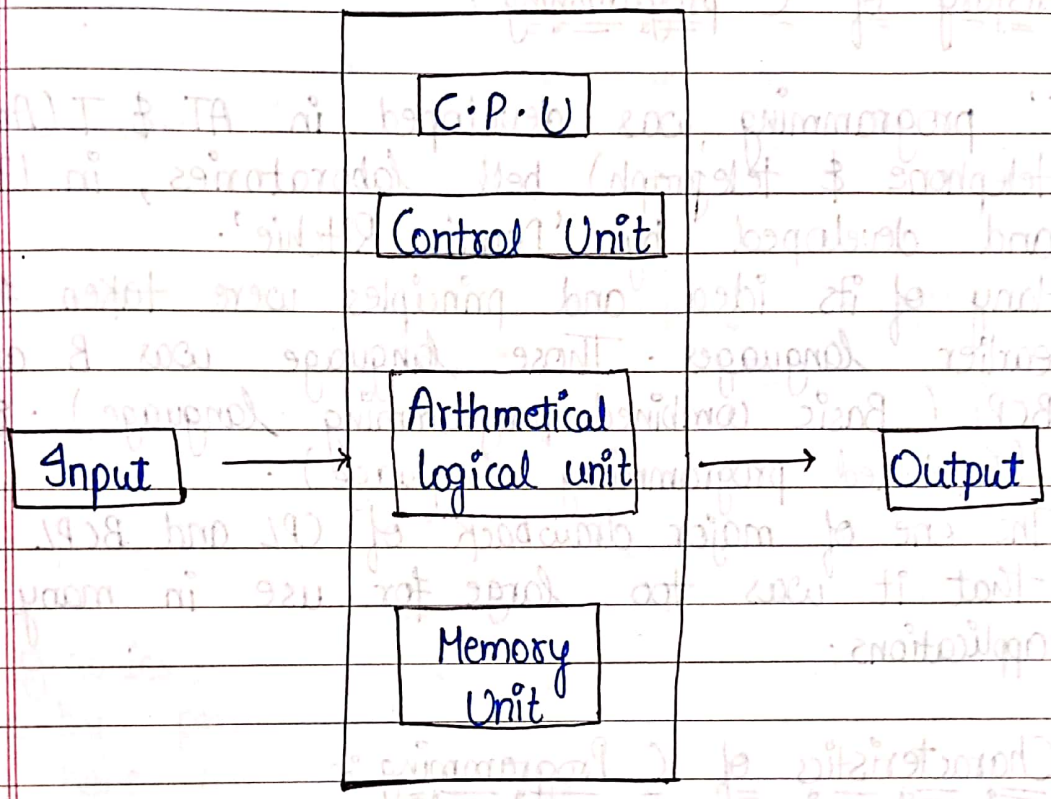
UNIT - I INTRODUCTION TO 'C' LANGUAGE

⇒ C language is developed in 1972 by 'Dennis Ritchie' at AT & T Bell laboratory.

Block diagram of computer :-



Programming language :-

A programming is a formal language, which comprises a set of instructions that produce various kinds of output. Programming language are used in computer programming to implement algorithms. Most programming language consist of instructions for computers. There are programmable machine i.e (turbo c) that use a set of specific instruction.

Advantages of programming languages :-

1) Its very productive to program.
2) Most programming language are fairly easy to learn and use.
3) You learn a lot from it.
4) You get better at it overtime.

History of C programming :-

'C' programming was developed in AT & T (American telephone & telegraph) bell laboratories, in 1972 and developed by 'Dennis Ritchie'.
Many of its idea and principles were taken from earlier languages. Those language was B and BCPL (Basic combined programming language). & CPL (Combined programming language).

⇒ The one of major drawback of CPL and BCPL was that it was too large for use in many applications.

Characteristics of C Programming :-

⇒ Modularity :- ability to breakdown a large module into sub module called as modularity.
This is an important feature of C programming language. This feature of C programming makes the de-bugging (i.e to find out the error) easy and fast.

(iii) Portability :- the ability to port, that is to install the software in different platform (i.e OS operating system) is called portability.
C language offers highest degree of portability.

(iii) Extendability :- 'C' is a modular. 'C' language code should be written in routine called function. These functions can be used for extend in other application.
In other words 'C' language is a basic programming language that's why we can say it is a building block of currently knowing language.

(iv) Speed :- 'C' programs are fast and efficient. This is because 'C' is used as a powerful set of data type and operators.

(v) Flexibility :- 'C' is a powerful and flexible language. 'C' used for projectation, word processor graphics, 'C' is a popular language that checked by professional programmers as a result and it have a large variety of companies are available.

⊡ COMPUTER LANGUAGE :-

A computer language means a set of rules and symbols uses to operate the ~~operate~~ computer, whatever command he give the computer, it first converted in its own language. A computer language is also known as a programming language. There are very style of programming language. Those are used to write program to tell the computer. What

Binary digits - 0-7

[4, 2, 1]

0 0 0 → 0

0 0 1 → 1

to do, to write those programms are called programmers.

These computer language has its own set of rules and grammar. Those are called as syntex and rules of the language. These syntex should be follow while writting the programs.

* THERE ARE 3 TYPES OF COMPUTER LANGUAGE :-

1) Machine level language OR low level language
2) Assembly level language.
3) High level language.

ⓐ MACHINE LEVEL LANGUAGE :-

As we know that a computer can only under-stand a special symbol which are represented by 0, 1. These two digits are called binary digits. Computer understand programms written in binary digit. The language which we used for the representation of binary digit is called machine level language.

→ Machine level language has its own advantage and disadvantage.

ⓑ Advantages of machine level language :-

1) Machine language makes fast and efficient in use of computer.
2) It do not require a translator to translate the code. It is directly

understood by the computer.

- Disadvantages :-

1) All operation of codes have to be remember.
2) It is hard to find out errors in a program written in machine level language.

(ii) ASSEMBLY LEVEL LANGUAGE :-

These language use letters and symbol instead of n binary digits. These symbols are called milimobics. Program written in assembly level language are called assembly code. Assembly codes are translated into machine level language instructions by assembler.
Assembly level language is easy to understand than machine level language.

- Advantages :-

1) Easy to understand as compare to machine level language.
2) Easy to remove error because the codes use english alphabets, it is easy to locate and correct them.
3) Changes can be possible in assembly level language.

(iii) HIGH LEVEL LANGUAGE :-

A high level language is a programming language that enables a programer to write program

that a more or less independent of a particular type of a computer. Such languages are considered as high level language because they are closer to human language.

* Advantages :-

⇒ The main advantages of high level language is that, they are easier to read, write and maintain

⇒ Program written in high level language must be translated into machine level language by a compiler or interpretors.

⇒ Understandability :- easy to understood

⇒ In high level language it is easy to find out error and solve them., (de-bugging).

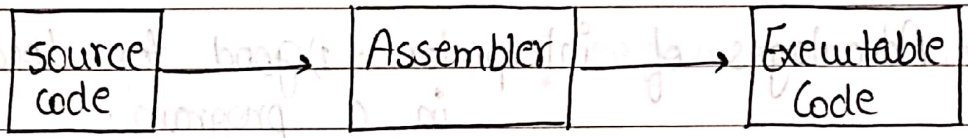
⇒ Program writter from one machine can run on different machine with very minor changes or no changes at all. (Portability)

* Program translators :-

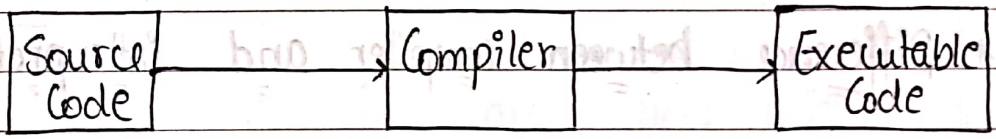
Sina a computer can only understand machine language, any program written in high level language cannot be execute directly. It must first translated into machine language. In order to convert it to machine language a translator is required.

There are three types of translator those are used for translating a program from high level language and assembly level language to machine language.

1) Assembler :- Since a computer cannot understand assembly level language directly, for this purpose a program is required to convert from assembly language to machine level language a program is required. It is known as assembler.



2) Compiler :- The high level language are english like languages and easy to learn and easy to understand. A compiler is a program that translate high level language into machine level language.



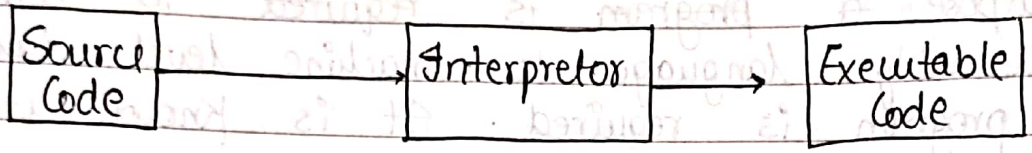
* Advantages of compiler :-

1) fast in execution.
2) The executable code produced by a compiler can be distributed.

* disadvantages of compiler :- 1) de-bugging of a program is must harder.

functionality of data type + the colour has been changed in white.

3) Interpretor :- An interpretor is also a program that translate high level source code into executable code. However, difference between compiler and interpretor is that an interpretor translate one line at a time and then execute it. while compiler translate a complete source code at a same time.



* Advantages of interpretor :- 1) good for locating errors in a program.
2) Useful for learning purpose.
3) de-bugging is easier since the interpretor stop when an error occured.

* Disadvantages of interpretor : 1) Rather slow than compiler.

* Difference between compiler and interpretor :-

Compiler	interpretor
1) translate entire source code into machine code in one attempt.	1) translate the source code line by line and indicates error at each step.
2) Every line is checked by the interpretor and then translate it.	2) Useful for commercial purpose.
3) Execution time is less.	3) Useful for learning purpose.
	4) Execution time is more

C ⇒ Case sensitive ⇒ Num, num
Both treated differently

32 = Keywords ⇒ predefined function

4) Slow for de-bugging. ≠ 5) good for fast de-bugging

Disadvantages of C programming :-

1) 'C' does not have concept oops (object oriented programming) that's why C++ is developed.
2) There is no routine checking in C language.
3) 'C' does not have the concept of constructer or deconstructer.

Variables and Keywords in C programming :-

A variable is a name that assign to a data storage location.

Syntax ⇒ Datatype variable name; / int a;

* Initialization of variable :- int a = 10;

* Rules for constructing variables :- A variable is always declared as character.

2) No comma or No space are allow within a variable name.
3) Special symbols are not allowed as an variable name.
4) Case matter's (that is case sensitive) then the variable name num and Num refers to a different variable.
5) Keywords cannot be used as a variable name.

Keywords :- are reserved words that are predefined in the language and cannot be used to denote other entity.

extension → .C , .CPP
↓ program can be run without header file
↳ program cannot be run without header file

String → sequence of character

All the keywords are in lowercase; and incorrect uses of keywords causes compilation error. There are 32 keywords in C programming. Some examples of keyword are :- int, char, float, break, continue, goto, return etc.

* Advantage of C programming :-

1) 'C' is highly portable language that means that 'C' program written from one computer can be easily run on another computer without any change or by doing a little change.
2) 'C' language is a building block of many other currently known language. C language has variety of datatype and powerful operators. Due to this program written in C language are efficient, fast and easy to understand.
3) C language is very easier to learn. The main advantages of C language is that there is no much vocabulary to learn.

* Constant in C programming :- constant refer to the fix value that the program may not alter/change during its execution. These fixed value are called literals. Constant can be any of the basic data type like an integer constant, a floating constant, a character constant or a string literals.

long int → format specifier → ~~%ld~~ %ld

'C' constant can be divided into two major categories:-

1) Primary constant :- (int, char, float).
2) Secondary constant :- (array, pointers, union).

* Rules for constructing integer constant :-

1) An integer constant must have at least one digit.
2) It must not have a decimal point.
3) It can be either positive or negative.
4) No (comma) or blank space are not allowed within integer constant.
5) The range of an integer constant depends upon the compiler; the range is (-32768 to 32767).

* Rules for constructing character constant :-

1) A character constant is a single alphabet, a single ~~digit~~ special symbol enclosed within a single inverted comma. Both inverted comma should point the left that is 'A'.
2) The maximum length of character constant should be one.
3) The range of a character constant is (-127 to 128)

* Rules for constructing float / Real constant :-

1) A float constant must have at least 1 digit.
2) It must have a decimal point, it could be either positive or negative.
3) No comma or blank space are not allowed with in a floating constant.

// changes with compiler.

4) The range of floating constant is $(-3.4e^{38}$ to $3.4e^{38})$.

* COMMENTS :- IN C PROGRAMMING :-

Comments are the words and statement in a program which are not compiled by compiler. Comments are used in C program to give some ~~hints~~ documentation about the program and to give some hints about the logic of an program. There are two types of comments in a program.

1) Single line comment :-

Single line comment

```
// write a program to add two numbers.
# include <stdio.h>
# include <conio.h>
void main ()
 {
  int a, b, c ;
// clrscr ();
//Printf (" Enter the value")
  Scanf ("%d%d", &a, &b);
  c= a+b;
  Printf ("%d", c);
  getch ();
 }
```

Single line comment

Single line comment on is always begin with two slash (//).

2) Multiline comment :-

```
/* Write a program to add two number, the number
  should be entered by a user */
#include <stdio.h>
#include <conio.h>
 void main()
{
/* int a, b, c;
 clrscr ();
Print f (" enter the value");
scan f (" %d %d", &a, &b);
 c= a+b; */
Print f (" %d", c);
getch ();
}
```

* IDENTIFIERS :- A name defined by the user or a programer in a program is called identifier.

Eg:- num, Num are two different identifiers.

⇒ The naming rules for an identifier are same rules for naming the variable.

* DATATYPE :- C support different type of datatype, storage represcutation of these datatype have a different memory. There are 5 fundamental of datatype in 'C' programming which are integer, character, float, double, void.

Single → 6 decimical value
double → 16 " "

⇒ "We can deal with integers, ~~real~~ float numbers, characters etc. All of which are similar entities → (Real world object) At the bit level, the computer may not understand integers, float numbers, or characters. Therefore, it is upto the compiler to make sure the ~~compiler~~ computer handles bits or bytes.

Keypoint :- Datatype simply refers to the size and type of data associated with variable and function.

⇒ Purpose of type declaration :- there can be ~~only~~ one and only one type associated with an entity. The type of entity establish the following information about it.

- Its meaning
- function that can be used with it.
- operation that can be perform with an entity.

⇒ Integer is used for store any integer value.
⇒ Character is used for storing a single character.
⇒ float is used for store any single precision for floating point number.
⇒ double is used to store any double precision for floating point number.

Categories of datatype :- 1) Simple datatype :- int, char, float
2) Structure datatype :- array, pointer, union.

we have to write a program of addition of two number in exam

(Range is changed with respect to compiler)

3) The void datatype :- void
4) Pointer datatype

Datatypes	Bytes	Bit
integers		
- very small integer	1 byte	8
- Short integer	2	16
- Normal integer	2	16
- long integer	4	32
Floating point		
- Single precision (float)	4	32
- Double precision	8	64
- Very large double precision (long double)	10	80
Character		
- Any single character	1	8

Assignment

Q:- Explain the characterists or history of C programing.
Q:- What are computer languages explain all?
Q:- Write a short note on 1) Keywords, identifier, variable, datatype?

Structure of C program
function as a building block in C programing :-

The first line of a program is # include is a pre-processor command which tell a 'C' compiler to include stdio.h before going to actual compilation.

The next line is void main () is the main function where the program execution begin. The next line printf and scanf is the another predefined function in 'C' programing. Printf is used for printing the statement and scanf is used for memory allocation. Getch is used for holding the output screen.

Program for hello

```
#include <stdio.h>
#include <conio.h>
void main ()
{
Printf ("hello C programing");
getch ();
}
```

standard include statement

main function

⇒ Write a program to find whether an entered no is ~~positive~~ or ~~negative~~) odd or even

```
#include <stdio.h>
#include <conio.h>
void main ()
{
int num;
Printf ("enter the number");
Scanf ("%d", &num);
if (num %.2 == 0)
{
print ("number is even");
}
```

```
else
{
 Print f ("number is odd");
}
getch();
 }
```

⇒ Program for finding wheter a no. is negative or positive :-

```
# include <stdio.h>
# include <conio.h>
 void main ()
 {
  int num;
 Print f ("enter the number");
 Scan f ("%d", &num);
  if (num>0)
 {
  Print f ("no is positive");
}
  else if (num<0)
 {
  Print f ("no is Negative");
}
 else
{
Print f ("0 is neither positive nor negative");
}
 getch();
  }
```

If the value of i is 5

i++ = i = i+1 = 6
i-- = i = i-1 = -4

classmate
Date
Page

UNIT-II

* Operator:- are used in 'C' programming for performing a task. An operator is a symbol that interact with 'C' to perform some operation or action. On one operant or more operant. The operant are something that operate on acts and response. We can divide operators in two categories:

1) On the basis of operant use.
2) On the basis of operation performed.

⇒ On the basis operant use, we can divide the operator into three forms.

(i) Unary operator:- which take only single operants for its operation, are called unary operators. Such as ++, --, != etc

(ii) Binary operators:- which take two operants for its operation, are called binary operator. Such as +, -, /, * etc. Ex. a+b

(iii) Ternary operator:- takes three operants for its operation. In 'C' programming, only conditional operator is a ternary operator.

Ex: num1 > num2 > num3.

2) On the basis of operation performed:-

(i) Arithmetic operator:-

operator which perform arthmetic operation are known as artmetic operator. The arthmetic operator are

+ → add two numbers
- → substract two number
* → Product of two number
/ → division of two number
% → view remiender or (modulus)

- When an operation performed with an integer, then result always in integer.
- If any operant between two operaut is float. so the result is always in float.
- Modulus work for integer not for float value.
- **Relation or Comparision :-**

A operator which defines a relationship between two operant are known as relational operator.

Ex: >, <, =<, ==, !=

< → less than
> → greater than
=< → less than equals to
== → equals equals to / double equal to
!= → not equals to

- **logical operator :-** are the operators which defines a relation between two or more Statements.

&& → and operator
|| → or operator
! → not operator

Expression ← x+y = a → variable name or
Left side → Right side

Assignment operation :- The assignment operator is the equal sign (=) it used in programing in some what different from it. In Assignment statement the ~~right~~ left side can be any expression and right side can must be a variable name.

Ex:
x+y = a
x+y = 3

* Conditional operator :- The conditional operator is sometime called as ternary operator since it take three operants, this is a short end of if-else, if statement.

Syntax :- variable = <exp1> ? <exp2> : <exp3>

If the expression first is true (that is if the value is non zero), then value written will be expression two. Otherwise the value written will be the expression three.

Ex:
```
# include <stdio.h>
# include <conio.h>
void main ()
{
int x, y;
Printf ("enter the value of x");
Scanf ("%d", &x);
y= x>5 ? 3:4;
Printf ("%d", y);
getch();
}
```

c = a+b*d
Left to right

\n → escape sequence

The statement stored in 3 in y if x > 5, otherwise it will stored 4 in y.

* Associatidity of operator :-

Q:- Write a program to find sum of 5 numbers and also calculate the ~~avegare~~? average ?

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a,b,c,d,e,sum,avg;
Printf("Enter the five numbers");
Scanf("%d%d%d%d%d",&a,&b,&c,&d,&e);
sum = a+b+c+d+e;
avg = (a+b+c+d+e)/5;
Printf("\n Sum is:",sum);
Printf("\n Average is:",avg);
getch();
}
```

* Precidence and associativity of operator :-

() → high c = (a+b) * d;
Right to left

+ - / * % → low c = a+b * d;
Left to Right

=

← L R

a, b, c d = 30
30 30 50 30
a = b = c = d

⇒
```
#include <stdio.h>
#include <conio.h>
void main ().
{
 int a, b=10, c=10, d=20;
a=b+c*d;
printf("%d", a);
 getch();
}
```

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

{
 int a, b=10, c=10, d=20;
a=(b++c)*d; // a=b=c=d;
Printf ("%d", a);
 getch();
}
```

⇒ Operator precidence determine which operator is performed first in an expression with more than one operators with different precidence.

Ex:- `c=a+b*d;`

This expression and `c=(a+b)*d;` will a different result.

* Associavity of operator :-

When an expression contain two operators of equal priority. The type tie between them is resolved by using associativity.

Associativity of can be of two type :-

(i) Left to Right ←
(ii) Right to left →

It means left operator which come first is evaluate firstly. Similarly, in case of Right to left associavity right operator is evaluated first which come first from the right side.

⇒ Write a program to calculate largest number between two number :-

```
#include <stdio.h>
#conio <conio.h>
void main()
{
int a,b;
Printf("enter the values");
Scanf("%d%d",&a,&b);
If (a>b)
{ Printf("a is largest");
}
else
{
Printf("b is largest");
}
getch();
}
```

```
#include <stdio.h>
#include <conio.h>
void main()
{
 int a,b,c;
Printf("enter the value of a,b,c");
Scanf("%d%d%d",&a,&b,&c);
If (a>b && a>=c)
{
 Printf ("a is largest no");
}
If (b>=a && b>=c)
{
Printf ("b is largest no");
}
If (c>=a && c>=b)
{
 Printf (" c is largest");
}
getch();
}
```

Q:- Write a program to find sum of all digit of an entered number :-

```
#include <stdio.h>
#include <conio.h>
 void main()
{
  int num, sum = 0; r;
Printf("enter the no");
Scanf("%d", &num);
```

```
while (num != 0)
{
r = num % 10;
Sum = sum + r;
num = num / 10;
}
Printf ("%d", Sum);
getch();
}
```

⇒ Step 1;

r = num % 10;
= 4231 % 10
= 1

Sum = sum + r
= 0 + 1 = 1

Num = num / 10;
= 4231 % 10;
= 423

⇒ Step 2: r = num % 10;
= 423 % 10;
= 3

Sum = sum + r
= 1 + 3
= 4

num = num / 10;
= 423 / 10
= 42

⇒ Step 3:- r = 42 % 10
= 2

Sum = sum + r
= 4 + 2
= 6

num = num / 10
= 42 / 10
= 4

⇒ Step 4:- r = 4 % 10
= 4

Sum = sum + r
= 6 + 4
= 10

Q:- Write a program for reverse digit of a entered number ?

```
# include < stdio.h >
# include < conio.h >
void main ()
{
 int num, rev = 0, r;
Print f (" enter the no ");
Scan f ("%d", & num);
while ( num != 0 )
{
r num = num % 10
rem = rev * 10 + r;
 num = num / 10 ;
}
Print f ("%d", rev);
 getch ();
}
```

reverse-

Step 1 :- r = num % 10
= 12345 / 10
= 5

rev = rev * 10 + r
= 0 + 5
= 5

num = num / 10
= 12345 / 10
= 1234

Step 2 :- r = num % 10
= 1234 % 10
= 4

rev = rev * 10 + r
= 5 * 10 + 4
= 54

num = num % 10
= 1234 / 10
= 123

Step3: r = num % 10
= 123 / 10 = 3

rev = 54 * 10 + r
= 543

num = num / 10
= 123 / 10 = 12

Step 4 :- r = 12 / 10 = 2

rev = 543 * 10 + 2
= 5432

num = num / 10
= 12 / 10 = 1

Step5:-
r = num % 10
= 1 / 10 = 1

rev = 5432 * 10 + 1
= 54321

Q:- Write a program to count all the digit of entered number ?

Preprocessor directive in 'C' :-

'C' language has the special feature of preprocessor which makes it different from other high level language, which do not have this type of facility.

* Some advantages using pre-processor directive are :-

1) Readability of a program increase.
2) Program modification become easy.
3) Make the program portable
4) We know that source code that we write it first translated into object code by compiler. but before the compiler compile each code it first passed through the preprocessor.
5) The pre-processor can get source code and modify it and then given to the compiler.

Source code → Pre-processor → compiler → Executable code.

to be continue ....

Q ⇒

```
# include <stdio.h>
# include <conio.h>
 void main ( )
  {
int num, r, count = 0;
Printf (" enter the number");
Scanf (" %d" & num) ;
while (num != 0);
  {
r = num % 10;
Count = count + 1;
num = num / 10;
 }
Printf (" %d", count);
 getch ( );
 }
```

Step 1 ⇒ $r = num \% 10 = \frac{4567}{10} \Rightarrow 7$

$Count = Count + 1 \Rightarrow 0 + 1 \Rightarrow 1$

$num = num / 10 \Rightarrow \frac{4567}{10} \Rightarrow 456$

Step 2 ⇒ $r = num \% 10 \Rightarrow \frac{456}{10} \Rightarrow 6$

$count = Count + 1 \Rightarrow 1 + 1 \Rightarrow 2$

$num = num / 10 \Rightarrow \frac{456}{10} \Rightarrow 45$

Step 3 ⇒ $r = num \% 10 \Rightarrow \frac{45}{10} \Rightarrow 5$

$Count = Count + 1 \Rightarrow 2 + 1 \Rightarrow 3$

num = num//10 ⇒ 45/10 ⇒ 4

Step 4 ⇒ r = num %10 ⇒ 4 %10 ⇒ 4
count = count + 1 ⇒ 3 + 1 ⇒ 4

Started

⇒ The line started with the symbol (#) are as a pre-processor.
⇒ Each pre-processor directive start with symbol (#). Their is no semi colon at the end of pre-processor directive.
⇒ To continue a directive on a next line, we should leave a blank space at the line end.
⇒ There can be only one directive on a line.
⇒ The pre-processor directive that perform these function given below :-

1) # include
2) # if
3) # else
4) # define
5) # line

* # define :- we also use this directive to define symbolic constant. The general syntax # define pre-processor is :- 3.14 / # define true.

⊡ Including files :- the pre-processor directive # include is used to include a file into a source code. We have already used this directive to include header file in a program.

To be continued....

⇒ Write a program to swap two variables :-

```
#include <stdio.h>
#include <conio.h>
void main ()
{
int a,b,c;
Printf(" enter the value of a,b,c before swapping");
Scanf(" %d", &a);
Scan Printf(" %d, &b);
  c = a;
  a = b;
  b = c;
Printf(" after swapping");
Printf(" %d", a);
Printf(" after swapping b = %d", b);
getch ();
}
```

⇒ Write a program to swap two variable without using third variable :-

```
#include <stdio.h>
#include <conio.h>
void main ()
{ int a,b;
Printf(" enter the value of a before swapping");
scanf (" %d", &a);
Printf(" enter the value of b before swapping");
Scanf(" %d", &b);
  a = a+b;
  b = a-b;
```

```
a= a-b;
Printf("/n after swapping a= %d", a);
Printf("/n after swapping b= %d", b);
getch();
}
```

a= 10; b= 20

Step 1
a= a+b
= 10+20
= 30

Step 2
b= a-b
= 30-20
= 10

Step 3:
a= a-b
= 30-10
= 20

started... :

The file name should be with in a angular brackets (<) or (>) (" ") double codes. The syntax is of including a file is

include < file name >
include " file name"

OR

include < stdio.h >
include " stdio.h"

The preprocessor replace the # include directive by the content of the specific file after including the file. The entire content of the file can be use in the program.

If the file name is in " " codes, Ist it is search is current directly where the source file is present.

If not found that it is search in standard include directly. If the file name is in < > brackets than the file is search in standard include directly only.

LOOPS IN 'C' PROGRAMMING :-

Loops are used when we execute a part of the program or a block of statement several times.

Ex:- Suppose we want to print "I am best" ten times, one way to get desire output is we write ten Printf statements, which is not suitable. Other way is. use loops. Using loop we can write statement and only 1 printf statement, and this approach is definately better than 1st one.

" Loops are used for repeating task

There are three types of loops in 'C' programming

(i) while loop
(ii) for loop
(iii) do-while loop

(i) While loop :- the while statement also called the while loop, execute block of statement as long as the specified condition is true. a while statement has following term.

Syntax :-

```
while (condition)
{
Statement 1;
Statement 2;
}
```

⊡ Write a program to print number from 1 to 10 using by loop :-

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

⊡ Write a program in reverse order with a difference of two :-

```
# include<stdio.h>
# include<conio.h>
void main ()
{
int i= 10;
while (i>=2)
{
Printf("\n %d", i);
i= i-2;
}
getch ();
}
```

(ii) For loop :- It is a 'c' program that execute the block of one or more statement a certain no. of time. It is sometime called loop.

\t → tabular form

Syntax :- for (intialization ; condition ; Increment/decrement)
{
statement1;
statement 2;
}

1. Write a program to print 1 to 100 using for loop :-

```
#include <stdio.h>
#include <conio.h>
void main ()
{
 int i=1;
for ( i=1; i<=100 ; i++)
{
printf ("\t %d", i);
}
getch ();
}
```

2. Write a program to find sum of series upto n numbers:

```
#include <stdio.h>
#include <conio.h>
void main ()
{
int num, i, sum = 0, [redacted];
Print f(" enter the numbers");
Scan f(" %d", & num );
for ( i=1; i<=num ; i++) //
{
sum = sum + i[redacted]; // r = i%.10
                          rev = rev*10 + r
                          i = i/10
                        // r = i%.10
                          count = count+1;
                          i = i/10;
```

for loop { reverse digit
count of digit

```
}
Printf("\n sum of term: %d", sum);
getch();
}
```

(ii) Write a program to print a table :-

```
void main()
{
int num, i;
Printf("enter the num");
Scanf("%d", &num);
for (i=1; i<=10; i++)
{
Printf("\n %d * %d = %d", n, i, n*i);
}
getch();
}
```

(iii) Do-while loop :- unlike the for and while loop, the do-while is an exit control loop. This means that the do-while loop always execute atleast one's. The do-while statement is suited for problems where the no. of times a statement block will execute is not known in advance.

```
syntax :- do
          {
          statement;
          }
          while (expression);
          { } → next line
```

* Write a program to print number 10 to 20 :- using do-while :-

```
# include <stdio.h>
# include <conio.h>
void main ()
{
  int a=10;
/* do loop execution */
do
{
printf("value of a is = %d \n", a);
a++;
}
while (a<=20);
getch();
}
```

* Write a program to calculate ~~sum of all digits~~ add numbers until the user enter a zero :-

```
# include <stdio.h>
# include <conio.h>
void main ()
{
int num, sum = 0;
do
{
printf ("enter the num");
scanf ("%d", &num);
{sum = sum + num;
}
while (num != 0);
```

Program for prime no :- 25/2 = 12

25 = 2 . . . 5 . . 12

divisible by 5 so it in not a prime no.

```
Printf("sum is = %d", sum);
getch();
}
```

- Write a program to print sum of all digit of a user given number using do while loop :-

```
#include <stdio.h>
#include <conio.h>
void main()
{
int num, sum=0, r;
do
{
printf("enter the num");
scanf("%d", &num);
r = num%10;
sum = sum+r;
num = num/10;
}
while (num!=0);
Printf("sum is = %d", sum);
getch();
}
```

- Write a program to find a no whether it is prime or not :-

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

```
{
int no, i, ans;
Printf("enter the no");
Scanf("%d", &no);
for(i=2; i<=no/2; i++)
{
ans = no%i;
if(ans==0)
{
Printf("no is not prime");
}
else
{
Printf("no is prime no");
}
getch(); → break!
}
```

- Program to calculate reverse of a digit using for loop:-

```
#include <stdio.h>
#include <conio.h>
void main()
{
int num, rev;
Printf("enter the no");
Scanf("%d", &num);
for(rev=0; num>0)
{
r= num%10;
rev = rev*10+r;
num = num/10;
}
```

```
// rev = (rev*10) + (num%10);
   num = num/10;
```

```
Printf("reverse = %d", rev);
getch();
}
```

Important for viva :-

Difference between :-

While	Do while
1) It is an entry control loop	1) It is an exit control loop.
2) Only one keyword 'while' is use	2) Two keywords 'Do' and 'while' are used.
3) Braces are not mendatory if there is only one statement in the body of the loop.	3) Braces are mandatory whether there are one or more statement in the body of the loop.
4) There is no semi-colon after the while condition.	4) Semi-colon is mandatory after the while condition.

* Write a short note on goto statement :-

```
# include <stdio.h>
# include <conio.h>
void main ()
{
int i=1;
clrscr ();
lab;
Printf("%d", i)
i++;
while (i<=10)
```

```
goto lab;
getch();
}
```

⇒ In 'C' language, the goto statement is used to transfer the control of a program from one statement to another statement unconditionally. Often goto is used with if statement to act only the certain criteria.

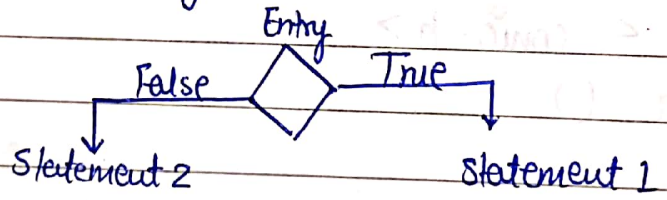
Syntax:- The lab is declared as;

```
label name:
goto labelname;
```

Note:- In the above program, the lab is the name of the label and the statement goto lab takes the control.

If else

- ~~Switch~~ statement :- If the statement evaluate true, then the statement is in block one are execute other wise the statement 2 is in block 2. This process is also ~~known~~ describe in diagram.



syntax:-

```
if(condition)
{
 statement 1;
}
else
{ statement 2;
}
```

Switch statement :- (Alternative)

The control statement that allow us to make a decision from the member of choice, is called switch. 'Default' Keyword is used to make when control we use switch statement.

'C' provide one or more multiway branching statement 'switch'. The switch statement provide a better alternative than a other statements. To select or choose multiple decision we use if else statement. If it is neccessary to select too many decision, it is recommended not to use if else statement. For this, there is a switch statement, with the help of switch case, we may choose any number of decisions.

Syntax :-

```
switch ( condition / expression)
{
case 1:
statement 1;
break;
Case 2:
statement 2;
break;
:  :
:  :
Case n:
statement n;
break;
default:
statement;
}
```

printf ~ puts
↓
similar

* Program for print 7 days of a week using switch statement :-

```
# include <stdio.h>
# include <conio.h>
void main ()
{
 int day_no;
Printf(" enter the day no");
Scanf(" %d", & day_no );
Switch ( day-no)
{
Case 1:
printf(" Sunday");
 break;
Case2:
printf(" Monday");
 break;
Case 3:
printf(" Tuesday");
 break;
Case 4:
printf(" Wednesday");
 break;
Case 5:
printf(" Thursday");
 break;
Case 6:
printf (" Friday");
 break;
Case 7:
print f (" Saturday");
```

c is small

```
break;
default:
printf("invalid day_no");
}
getch();
}
```

* Program for calculator:-

```
#include <stdio.h>
#include <conio.h>
void main()
{
char choice;
float no1, no2;
printf(" enter the choice");
scanf("%c", &choice);
printf(" enter two number");
scanf("%f%f", &no1, &no2);
switch (choice)
{
 case '+':
printf(" %f + %f = %f", no1, no2, no1+no2);
 break;
Case '-':
printf("%f - %f = %f", no1, no2, no1 - no2);
 break;
case '*':
printf(" %f * %f = %f", no1, no2, no1 * no2);
 break;
Case '/':
printf(" %f / %f = %f", no1, no2, no1 / no2);
```

```
break;
default;
printf(" enter operator is not correct");
}
getch();
}
```

OR

```
#include <stdio.h>
#include <conio.h>
void main()
{
char choice;
int a,b,c;
Printf("enter the choice");
scanf("%c",&choice);
switch(choice)
{
case 1:
printf(" enter the no");
scanf(" %d %d",&a,&b);
c=a+b;
printf("%d",c);
break;
case 2:
printf(" enter the no");
scanf(" %d %d",&a,&b);
c=a-b;
printf("%d",c);
break;
Case 3:
printf("~~%d",c~~) enter the no");
scanf(" %d %d",&a,&b);
```

```
c=a*b;
printf("%d",c);
break;
case 4:
printf("enter the no");
scanf("%d%d",&a,&b);
c=a/b
printf("%d",c);
break;
default:
printf("invalid number");
}
getch();
}
```

Q:- Write a program to check whether a character is vowel or consonant?

```
#include<stdio.h>
#include<conio.h>
void main()
{
char ch;
printf("enter the alphabet");
scanf("%c",&ch);
switch(ch)
{
case 'a':
printf("vowel");
break;
case 'e':
printf("vowel");
```

```
break;
case 'i':
printf("vowel");
 break;
case 'o':
printf("vowel");
 break;
case 'u':
printf("vowel");
 break;
default:
printf("consonant");
}
getch();
}
```

* Break statement :- / (Jumping statement)

The break statement transfering the control out of the block in which it is used. The break statement must be used in conjuction with a for, with a while and switch statement. When break statement execute in the switch statement, it terminates the statement in a sequence. When break statement execute in a loop, control exist the loop immediately.

syntax :- switch (condition / expression)

```
{ case 1:
statement 1;
 break;
```

```
case 2:
statement 2;
break;
  :   :
  :   :

case n:
statement n;
break;
default:
statement;
}
```

Cout << character;

Ex:- ⇒ Write a program of switch :-

* Continue statement :- //(Jumping statement)

```
# include <stdio.h>
# include <conio.h>
void main ()
{
int a;
for (a=1; a<=20; a++)
{
if (a==9)
{
continue;
}
printf ("%d", a);
}
getch ();
}
```

In looping statement, sometime a situation may arise where we want that from a given statement onward upto the last statement of the loop are to be bypassed. This task is accompolished by using the continue statement, this continue statement must be used in conjuction with a for, while and do while statement.

UNIT - 3

PROBLEM SOLVING :-

problem solving is a mental process and is a part of the larger problem process that include problem finding and problem ~~slipping~~ shaping. Problem solving skills include the ability to recognise and define problem, invent an implement solution and track and evaluate result. Each of this is capable of solving problem in our own way and thus, a selection of problem solving may seen a big difficulty to some people. However, each of this can ~~improve~~ work to improve our problem solving skills.

Problem solving in a computer is a task of expresing the solution of the problem in term of simple concept operation and code.

⇒ Understanding the problem.
⇒ Producing an algorithm to solve the problem.
⇒ Translating the algorithm into a specific programming language.
⇒ Testing the resultant program.

Characterstic of difficult problem :-

(i) Complexity :- large no. of item and decisions.

(ii) Intransparency :- lack of clearity of the ~~situation~~ solution.

(iii) problem solving technique :-

⇒ Try an error.
⇒ Testing possible solution until the right one is found.
⇒ Divide & conquer.

Breaking down a large complex problem into smaller, and after the solution merge all the problems.

⇒ Research :- gather existing ideas or adopting existing solution to similar problem.

⇒ Brain-storming :- Suggesting a large no. of sol" or idea and combining and developing them until an optimum sol" is found.

⊡ ALGORITHMS :- An algorithm is a sequence of instruction design in such a way that if the instructions are executed in a specified sequence.

⊡ Characteristics of algorithms :-

1) It should accept input.
2) Process method should be clear and understandable.
3) It should have finite steps.
4) It provide atleast one output.

⊡ FLOWCHART :- A flowchart is step-by-step diagramatic or graphical representation of an algorithm. That are used for the sol" of problem and flowcharting is a technique of creating the flowchart.

* Nested if-else :-

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

```
void main ()
{
int var1, var2;
printf(" enter the value of var1, var2");
scanf(" %d %d", &var1, &var2);
if (var1 != var2)
{
printf(" var1 is not equal to var2");
Nested if - else //
if (var1 > var2)
{
printf (" var1 is greater than var2");
}
else
{
printf(" var1 is less than var2");
}
}
else
{
printf (" var1 is equals to var2");
}
getch();
}
```

* Steps in problem solving :-

Problem solving on computer is a task of expressing the sol^n of the problem in terms of simple concept, operation and computer code. ~~The newly develo~~ There are three following steps for solving any type of problem.

(i) Define the problem:- The most important of the problem solving step is to define the problem correctly. The way you define the problem will determine how you attempt to solve it.
For example :- If you recieve a complain about one of your project team members from a client, the sol" you come up with will be different based on the way you define the problem.

(ii) Determine the causes :- Ones you have define the problem you are ready to start to determine what is causing it.

(iii) Generate idea :- Once the hardwork of defining the problem and determining its cases has been complete. Its time to get creative and develop possible solution to the problem.

(iv) Select the best sol" :- After you come with several ideas that can solve the problem, one problem solving technique you can use to decide which one is the best sol" for the problem.

(v) Take action :- Once you have determine which sol" you will implement its time to take action.

* Armstrong no :- cubic sum of all digit as same as a no. is known as armstrong no.

Ex:- 1, 0, 153, 1634 etc.

```
#include <stdio.h>
#include <conio.h>
void main ()
{
int no, rem, result=0, orgin no;
printf("enter the no");
scanf("%d", &no);
orig no= no;
while (orig no != 0)
{
rem = orig no % 10;
result= result + rem * rem * rem;
orig no = orig no /10;
}
if (result == no)
{
printf("%d is a Armstrong no");
}
else
{
printf("%d is not armstrong no");
}
getch ();
}
```

if (no == 0) printf("No is Armstrong");

no > 0

* Write a program to find whether an entered no palandrome ? or not ?

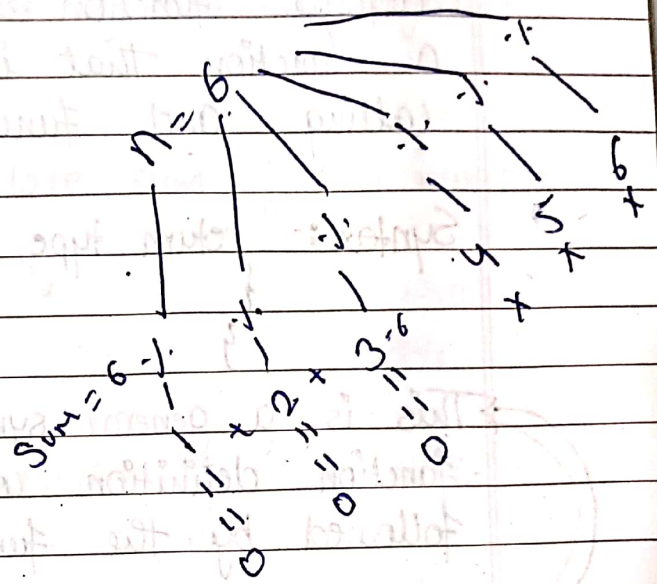
Q:-

```
#include<stdio.h>
#include<conio.h>
void main()
{
int num, rem, rev=0; orig no;
printf("enter the no");
scanf("%d", &no);
orig no= no;
while(orig no!=0)
{
rem = orig no %.10;
rev = rev*10+r;
orig no = orig no/10;
}
if(rev==no)
{
printf("Palindrom no");
}
else
{
printf("Not palindrom");
}
getch();
}
```

UNIT-5

Q:- Write a program to check whether a no. is perfect no. or not ?

```
#include <stdio.h>
#include <conio.h>
void main ()
{
 int no, i, sum = 0;
printf("enter the no");
scanf("%d", &no);
for( i=1; i<no; i++)
 if ( no%i==0)
sum = sum+i;
if ( sum==no)
{
printf("Number is perfect");
}
else
{
printf("Number is not perfect");
}
getch ();
}
```



☐ FUNCTION :- A function is a self contain programe structure that perform a specific task. Any 'C' program we contain atleast one function. If a program contain one function, it must be main.

If their are more than one function work of

Function → declaration
→ calling
→ definition

these function are increased:
In 'c' programming there are some function already define. They are called library function. These library function can be used in 'C' program. Beside them user can create new function with library function. These function are called user defined function. There are three method of a function that is function definition, function calling and function declaration

Syntax:- return type function name(parameter list)
{
}

This is a general syntax to define a function. The function definition consist of a ~~for~~ called the declarator followed by the function body.

Definition of function:-

The function is composed of the statement that make up the function determine by braces. The declarator of the function has 3 basic parts.

1) The type of the value that the method ~~written~~ return (~~written~~ return type) ~~put~~ could be a simple datatype. (Such as int, float etc) as well as any user define datatype. If the function does not ~~written~~ return any value it could be even be void type.
2) The rules of ~~for~~ naming a function are same as for identifier. parentheses () are necessary after the function name.
3) An optional list of parameters, on which the

); → with semicolon (declaration)
) → without semicolon fund definition

function will operate, must be enclosed in parentheses. This list contain the variable name with their type. These are used to hold all the values that the function will operate all.

syntax:-
datatype function name (list of parameter/Argument)
int abc (int a, int b)

Q:- Write a program to calculate sum of two numbers with function declaration?

```
#include <stdio.h>
#include <conio.h>
void main () int sum (int x, int y);
int main ()
{
int a, b, s;
printf("enter the value");
scanf("%d%d", &a, &b);
S= Sum (a,b);
printf("%d", s);
getch ();
return 0;
}
{
int sum (int n, int y)
{
int s;
S=n+y;
return (S);
}
```

int max (int i, int num)
{
int result

Q: Write a program to calculate sum of an entered no.?

```
#include <stdio.h>
#include <conio.h>
int Digit sum (int num);
void main ()
{
int num, S;
printf(" enter the no");
scanf("%d", &num);
S= Digit sum (num);
printf("%d", S);
getch ();
}
int Digit sum (int num)
{
 int s=0, r;
while (num!=0)
{
 r= num %10;
 S= S+r;
num= num/10;
}
return (S);
}
```

Definition

```
int max(int num1, int num2)
 {
int result;
 if (num>num2)
  result= num1;
else
 result=num2;
 return result;}
 }
```

☐ Calling of a function :- A function is called by specifying its name with an expression. The call is terminated by a semicolon. Executing the call statement causes the function to execute. i.e control is transfered to the function; the statement in the function body are executed

and then control returns to the statement following the function call statement.
If we take the example of a program when main execute, there is calling statement for volume function.

Ex:- V= volume (width, height, breadth);

The function declaration :- [Prototype]
A function prototype is a function declaration that specify the datatypes of its argument in the parameter. The compiler use the information in a function prototype to ensure that the corresponding function definition and all corresponding function declaration and calls with in the scope of the prototype contain the correct no. of argument or parameter.
We can eliminate the function declaration if the function definition appears in the listing before the first call to the function.

Syntax:- return type function name (Parameter list)
int ABC (int n)

Example:-
```
int add (int x, int y)
{
return (a+b);
}
```

Advantage of function defination :-

Function definition has following advantages :-

1) It helps the compiler in determining whether a function is called correctly or not.
2) A function must be define before calling it but function definition allow a function to be called before using it.

Explain local and global variable :-
A quantity which may vary during program execution is called variable. Variable names are those names given to the location in the memory of computer where different values are stored. Two major type variables are :-

1) **Local variable** :- those variable which are declared inside a block or in the body of a function are known as local variable. local variable are excessable to the block in which they are declared. They cannot be access by other function.

2) Global variable :- those variable, which are declared outside all function, are known as global variable. Global variable are excessable to all the function of the program. That means, the scope of global variable is global. In most cases they are declared in the begining of the program.

Q:- Ex:- Program to calculate the average of 3 number using local and global variable.

```
#include <stdio.h>
#include <conio.h>
int n1, n2, n3;
float Average ();
void main ()
{
float a;
n1 = 423;
n2 = 30;
n3 = 50;
a = Average ();
printf ("%f", a);
getch ();
}
float Average ()
{
float avg;
avg = (n1+n2+n3)/3;
return (avg);
}
```

* Categories of user define function :- According to the argument, return type. user define function can be four type :-

1) function with argument and with return type :- those function come in this category which takes one or more argument as well as return some values using the return statement. function digitrev () in a program 1 in this category.

Q. Write a program to calculate rev of digit of an entered no ?

```
#include <stdio.h>
#include <conio.h>
void int digitrev(int num);
void main()
{
int num, rev;
printf("enter the no");
scanf("%d", &num);
rev = digitrev(num);
printf("%d", rev);
getch();
}
int digitrev(int num)
{
int rev=0, r;
while (num!=0)
{
r= num%10;
rev = rev*10+r;
num = num/10;
}
return (rev);
}
```

2) Function without argument and without return type:- These function come in this category which does not take any argument as well as return no values.

Q: Program to display a simple message using a function having no argument and no return type.

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

3) function with argument and without return type :-

Those function come in this category, which takes one or more argument but return no value.

Ex- Program to print the greater no among two int using a function having two arguments and no return type?

```
#include <stdio.h>
#include <conio.h>
void greater_no(int a, int b);
void main()
{
greater_no(45,60);
getch();
}
void greater_no(int a, int b)
{
if(a>b)
```

```
{
printf("a is greater than b");
}
else
{
print("b is greater than a");
}
```

* Approaches of passing argument to a function :-

There are two approaches to passing argument to a function. These are :- call by value, call by refrenc

1) Call by value :- this approach is also popularly known as passing argument by value. In this approach, the value of the actuall arguments are passed to the function during function call. when control is transfered to the called function.

Ex:-

```
# include <stdio.h>
# include <conio.h>
void swap ( int a, int b);
void main ()
{
 int a = 10;
 int b = 20.
Printf(" before swapping");
printf(" %d %d", a, b);
swap (a, b);
printf(" after swapping ");
printf(" %d %d", a, b);
```

* Use of ⇒ pointer → which contain the address of another variable

```
getch();
}
void swap (int a, int b)
{
int c;
c=a;
a=b;
b=c;
}
printf("after swapping");
printf("%d %d", a, b);
}
```

2) Call by refrence :- this approach is also popularly known as passing argument by refrence. In this approach, the address of the actuall argument are passed to the function during function call. When control is transfered to the called function.

```
#include <stdio.h>
#include <conio.h>
void swap (int *a, int *b);
void main()
{
int a=10;
int b=20;
```

```
printf(" Before swapping");
printf("%d%d", a, b);
swap (&a, &b);
printf(" After swapping");
printf("%d%d", a, b);
getch();
}
void swap (int *a, int *b)
{
int c;
c = *a;
*a = *b;
*b = c;
}
```

* Difference between call by value and call by refrence :-

Call by value	Call by refrence
1) The actuall parameter may be constant, variable, function, function call or a combination of these.	1) The actual parameter must be a variable,
2) The actuall value of a variable ~~function~~ is call.	2) The actuall ~~of value of~~ address of the variable is passed.
3) Only a single value can be return from a function using return statement.	3) ~~One val~~ More than one value can be return from a function, without using the return statement.

⇒ function call itself again & again

Date
Page

* Recursion :- in mathematics and computer sicnce, recursion means self refrence, recursion is a repeative process in which a function call itself again and again till a termination condition is true.

Q:- Program to calculate factorial of a number ?

```
#include <stdio.h>
#include <conio.h>
void main()
{
long num, i, fact=1;
printf("enter the number");
scanf("%ld", &num);
for(i=1; i<=num; i++)
fact= fact * i;
printf("%ld", fact);
getch();
}
```

⇒ Using this technique we can solve those problems, which are defined in terms of themselves consider the following examples 5!= 5*4! according to the first examples factorial of 5 can be calculated after calculating the factorial of 4. That means these operation are defined in terms of themselves.

⇒ For using recursion technique in programing, the programing must support this technique. 'C' support recursion technique.

Q:- Program to calculate factorial of a no. using function ?

```
#include <stdio.h>
#include <conio.h>
void fact long (num);
void main ()
{
long num;
printf(" enter the value");
scanf ("%ld", &num);
fact (num);
getch ();
}
void fact ( long num)
{
long fact = 1, i;
for (i=1; i<=num; i++)
fact = fact * i;
printf("%ld", fact);
}
```

- STORAGE CLASSES :- every variable in 'C' holds a characteristic called its storage class. The storage class define mainly two characterstic of the variable.

1) lifetime :- The lifetime of a variable is the length of time it retains a particular value.
2) visibility or scope :- the visibility or a scope of a variable refers to those part of a program, which will be able to access it directly.

Types of storage classes :-

1) Automatic storage class.
2) static storage class.
3) Register storage class.
4) External storage class.