Date : 18 / 1 / 18 Page No.: ____

Functions

A function is a self contained block of statement which is designed to fulfil a particular task.

Advantages of functions:-

* Code reuseability
* Easy to understand
* Code length is decreased
* Easy to debug
* Balanced load distribution

Types of function:-

1. Build-in functions
2. User defined function

User defined functions:-

(1) Function declaration (return type function name (argument type list);
(2) Function definition (return type function name (datatype value 1, data type 2 value 2)
(3) Function call ↳ Header

{
local declaration
body
return
}

Date : __/__/____ Page No.: ____

(5) Function call (syntax)

```
void main ( )
{
int n = function_name ( );
}
```

Q. W.A.P. to add two numbers & display their sum using function

1)
```
#include <stdio.h>   → optional
int add (int, int) ;
void main ( )
{
 int a, b, sum ;
printf ("enter two numbers");
scanf ("%d %d", &a, &b);
   sum = add (a, b);
printf ("sum = %d," sum );
}
int add (int n, int m)
{
  int c ;
c = n+m;
}
```

Date : __/__/____ Page No.: ____

Category of user-defined functions based on argument & return type

1) Function with argument & with return-type
2) Function with argument & without " "
3) Function without " " no " "
4) " " " but with " "

2)
```
#include <stdio.h>
void add (int a, int b)
{
  int c ;
  c = a+b;
Printf ("%d", c);
}
void main ( )
{
int a, b, c;
printf ("Enter two numbers");
scanf ("%d %d", &a, &b);
  add (a, b);
}
```

Date : / / Page No.:

2)
```
#include <stdio.h>
void add (int a, int b)
{
```

3)
```
#include <stdio.h>
void add ()
{
int a, b, c;
c = a+b;
Printf ("sum = %d", c);
}
void main ()
{
add ();
}
```

3)
```
#include (stdio.h>
void add ()
{
int a, b, c;
printf ("Enter two numbers");
scanf ("%d %d", &a, &b);

c = a+b;
printf ("sum = %d", c);
}
void main ()
{
add ();
```

Date : / / Page No.:

4)
```
#include <stdio.h>
int add ()
{ int a, b, c;
printf ("Enter two numbers");
scanf ("%d %d", &a, &b);

c = a+b;
return (c);
}
void main ()
{ int sum;
sum = add ();
printf ("sum = %d", sum);
}
```

Q. W.A.P. to calculate the factorial of a given number.

1)
```
#include <stdio.h>
void Fac (int a)
{
int s = 1, i;
for (i=1; i<=a; i++)
s = s * i;
printf ("Factorial = %d", s);
}
void main ()
{ int a;
```

Date : / / Page No.:

```
Printf ("Enter the number");
scanf ("%d", &a);

fac(a);
}
```

(2.)

```
#include <stdio.h>
int fac(a)
{
int s=1, i;
for( i=1; i<=a; i++)
 s=s*i;
return(s);
}
void main()
{
int a,b;
Printf ("Enter the number");
Scanf ("%d", &a);

 b= fac(a);
printf ("Factorial = %d", b);
}
```

Date : / / Page No.:

(3.)

```
#include <stdio.h>
void fac()
{
int a, i, s=1;
Printf ("Enter the number");
scanf ("%d", &a);
for (i=1; i<=a; i++)
s=s*i;
printf ("Factorial = %d", s);
}
void main()
{
 fac();
}
```

(4.)

```
#include <stdio.h>
Int fac(a)
{
int i, s=1, a;

Printf ("Enter the number");
scanf ("%d", &a);
for (i=1; i<=a; i++)
s=s*i;
return(s);
}
void main()
{
Int b;
```

Date : __/__/____ Page No.: ____

```
b = fac();
Printf("Factorial = %d", b);
}
```

Palindrome

```
#include <stdio.h>
void pal(int a)
{
int s=0, r, n;

n=a;
while(n!=0)
{
r= n%10;
s= s*10+r;
s= s+r*10;        n=n/10;
}

If(a==s)
printf("No. is palindrome");
else
printf("No. is not palindrome");
}
void main()
{
int a;
printf("Enter the number");
scanf("%d", &a);
```

Date : __/__/____ Page No.: ____

```
pal(a);
}
```

1 # Check Armstrong no.
2 # Calculate sum of a no.
3 # Reverse of a given no.
4. # Robinson armst

(2)

```
#include<stdio.h>
void sum()
{
int a, i, r, s=0;
Printf("Enter the number");
scanf("%d", &a);
for(i=1; a>0; i++)
{
r= a%10;
s= s+r;
a= a/10;
}
Printf("sum = %d", s);
}
Void main()
{
sum();
}
```

Date : __/__/____ Page No.: ____

3.
```
#include <stdio.h>
void reverse(int a)
{
int i, r, s=0;
for(i=1; a>0; i++)
{
r=a%10;
s = r+s*10;
a = a/10;
}
Printf("reverse=%d", s);
}
void main()
{
int a;
Printf("Enter the number");
scanf("%d", &a);
reverse(a);
}
```

Date : __/__/____ Page No.: ____

4. Robinson

```
#include <stdio.h>
int fac(int a)
{
int i, s=1;
for(i=1; i<=a; i++)
s=s*i;
return(s);
}
void main()
{
int a, s=0, n;
Printf("enter the number");
scanf("%d", &a);
n= a;
for( ; a>0; a=a/10)
{
r= a%10;
s = s + fac(r);
}
If (n==s)
Printf("No. is robinson");
else
printf("No. is not robinson");
}
```

Date : __/__/____ Page No.: ____

Armstrong no.

```
#include <stdio.h>
void armstrong (int a)
{
int b=0, s=0, n=a;
while (a>0)
{
a= a/10 ;
b++ ;
}
while (n>0)
{
r= n%10;
s = s + pow(r, b);
n= n/10 ;
}
If (p==s)
Printf ("No. is armstrong");
else
printf ("Not armstrong");
}
void main ()
{
int a;
Printf ("Enter the number");
Scanf ("%d", &a);
armstrong (a);
}
```

Date : __/__/____ Page No.: ____

Prime no.

```
#include <stdio.h>
void prime(int a)
{
int i, f=0 ;

for(i=2; i<a ; i++)
{

If(a%i==0)
f++;
}
If (f==0)
Printf ("No. is prime");
else
printf ("No. is not prime");
}

void main ()
{
int a;
Printf ("Enter the number");
Scanf ("%d", &a);

prime (a);
}
```

Date : __/__/____ Page No.: ____

Q. W.A.P. to calculate the binomial coefficient using function.

```
#include <stdio.h>
void fac (int a)
{
int s
```

```
#include <stdio.h>
int fac (int a)
{
int s=1, i ;
for(i=1 ; i<=a ; i++)
s=s*i ;
return(s);
}

float binomial (int a, int b)
{
float x ;

x= fac(a)/(fac(b) * fac(a-b));

return(x);
}

Void main()
{
int n, r; float s;
Printf ("Enter n & r");
Scanf ("%d %d", &n, &r);
```

Date : __/__/____ Page No.: ____

```
s = binomial (n, r) ;
Printf ("Binomial coefficient is %f", s);
}
```

Parameter passing techniques :-

1) Call by value

Q. W.A.P. to input a number & multiply it by 10 & display the result using call by value

```
#include <stdio.h>
void mul (int a)
{
int s ;
s = a * 10;
Printf ("No. is %d", s);
}
void main()
{
int a;
Printf ("Enter the number");
Scanf ("%d", &a);
mul(a);
}
```

~~5~~ 3) S, C

Date : __/__/__ C=3 Page No.: ____

Q. W.A.P. to input two numbers & swap their value using call by value.

```
#include <stdio.h>
void swap(int a, int b)
{
  int c;
  c = a;
  a = b;
  b = c;
  printf("No. are %d & %d", a, b);
}
void main()
{ int a, b;
printf("Enter two numbers");
scanf("%d %d", &a, &b);
 swap(a,b);
}
```

Call by value :-

* In call by value we send a duplicate copy of actual parameter to formal parameter.
* The changes made in formal parameters will not reflect back to the actual parameters.

Date : __/__/__ Page No.: ____

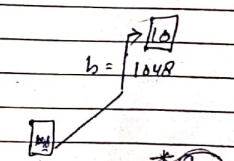
* It takes a large memory to copy the value of actual argument to formal argument if we are using many no. of arguments.
* To copy these values from actual to formal parameters will be a time taking process.
* In call by value we can only return a single value to the calling function.
* In call by value the values of actual arguments will remain same & hence it is secure parameter passing technique.

Call by refrence :-

* Note Pointer: A pointer is a variable which stores the address of another variable.

datatype *Var_name;

```
int *a;
int b=10;

a = &b;
printf("b = %d", b);
printf("b = %d", *a);
```



3.14
12.56

Date : / / Tut-2 Page No.: 1

(1) error (return two times)

(2) conflicting type for area circle
a is float & area circle is int

(4)
```
main()
{ int i=3, k, l;
  k = add(i++);
  l = add(i++);
  printf("i=%d k=%d l=%d", i, k, l);
}
add(int ii)
{
++ii;
return(ii);
}
```
i=5
k=5
l=5

Date : / / Page No.:

(5) k=38 k 35

(6) k=35 m=36
z=38

(7) int i=135, a=135, k; a=136 i=136

i=136, a=136

c=0

136, 136, 0.000000

```
#include<stdio.h>
int void pal(int a)
{
int r, s=0, n;
n = a;
for( ; a>0 ; a=a/10)
{
r = a%10;
s = r + s*10;
}
If (n==s)
return(1);
else
return(0);
}
```

Date : __/__/____ Page No.: ____

```
int
void prime (int a)
{
int i, f=0;
for(i=2; i<a; i++)
{
 if(a%i==0)
 {f++;
  break;}
}
If (f==0)
return (1);
else
return(0);
}
void main()
{
int a, x, y;
Printf(" Enter the number");
scanf("%d", &a);
a++;
while(1)
{
 x = pal(a);
 y = prime(a);
 If (x==1 && y==1)
 { Printf("No. is %d", a);
   break;
 }
 else
 a++;
}
```

Date : __/__/____ Page No.: ____

Storage classes

1. auto
2. register
3. static
4. extern

1) auto (automatic) :-

- storage ⟶ Memory (RAM)
- scope ⟶ local to the function or block where it is declared
- default value ⟶ garbage value
- Lifetime ⟶ upto the function call (until the control remains in that function / block)

Eg:

```
void fun1(void);
void main()
{
int a = 10;
Printf("a = %d", a);

fun1();
fun1();

Printf("a = %d", a);
}
void fun1()
{auto int a=5;
 a++;
 Printf("a= %d", a); }
```

Output

```
10
6
6
10
```

Date : __/__/____ Page No.: ____

2. Register

storage → CPU register
life
scope
default value } same as auto.

Q. W.A.P. for register to find the factorial of a no. using register storage class.

```
#include <stdio.h>
void fact(int n)
{
register int f=1, i;
for(i=1; i<=n; i++);
{
f = f * i;
}
Printf("Factorial = %d", f);
}
void main()
{
int a, f;
Printf("Enter the number");
Scanf("%d", &a);
```

register memory is used

It include only some no. of variable i.e. if Register memory=8 register a, b, c, d

Register auto

Date : __/__/____ Page No.: ____

```
fact(a);
}
```

3) static

default value → 0
scope → local to function/block where it is declared
life time → through out the program
storage → memory (RAM)

```
void fun();
static int count = 5;
void main();
{
while (count--)
{
fun();
}
}
void fun()
{
static int i=5;
i++;
Printf("i = %d & count = %d", i, count);
}
```

count	i
5	5
4	6
i=6	count = 4
i=7	count = 3
i=8	count = 2
i=9	count = 1
i=10	count = 0

Date : __/__/____ Page No.: ____

Extern :-

Default value ⟶ 0

scope ⟶ within program & file scope

life ⟶ Throughout the program

storage ⟶ RAM (memory)

```
int a=10;
void main()
{
```

Eg.

```
void fun()
int a=10;
void main()
{
printf("%d", a);
fun();
}
fun()
{
Printf("%d", a);
}
```

Date : __/__/____ Page No.: ____

Eg.

```
void fun();
void main()
{
extern int a;
printf("%d", a);
fun();
}
void fun()
{
extern int a;
printf("%d", a);
}
int a=10;
```

Eg.

```
#include <stdio.h>
int count;
extern void use_extern();
void main()
{
count = 5;
use_extern();
}
```

a.c

```
#include "a.c"
void use_extern()
{
extern int count;
printf("count=%d", count);
}
```

b.c

gcc a.c b.c

Date : __/__/____ Page No.: ____

Q.1 W.A.P. to input an array & search a particular element by passing array, it's size & element.

Q3. W.A.P. to input an array & find the position of 1st occurrence of a particular character.

Q2. Find the last occurrence.

Q4. W.A.P. to input an array, find the max, & min & swap their value & then display updated array.

Q5. W.A.P. to input an array & sort the element using bubble sort.

(1)
```
#include<stdio.h>
void search(int a[], int n, int a)
{ int i;
for (i=0; i<n; i++)
{
If (ar[i]==a)
{
Printf ("No. is found at %d", i+1);
break;
}
}
else
Printf ("No. is not found");
}
```

Date : __/__/____ Page No.: ____

```
void main()
{
int a, n, b[10];
Printf ("Enter the size of array");
scanf ("%d", &n);
Printf ("Enter the array");
for (i=0; i<n; i++)
scanf ("%d", &b[i]);
Printf ("Enter the number to be searched");
scanf ("%d", &a);

search (b, n, a);
}
```

(2.)
```
#include<stdio.h>
void occur(int ar[], int n, int a)
{ int i;
for (i=0; i<n; i++)
{
If (ar[i]==a)
{ Printf ("First occurence of no. is at %d", i+1);
break;
}}
Printf
}
```

```
void main()
{
int a, b[100], n;
Printf("Enter size of array");
scanf("%d", &n);
Printf("Enter Assay");
for(i=0; i<n; i++)
scanf("%d", b[i]);
printf("Enter the number");
scanf("%d", &a);

occur(b, n, a);
}
```

(3.)

```
#Include<stdio.h>
void last(int ar[], int n, int a)
{
int i, s;
for(i=0; i<n; i++)
{
if(ar[i]==a)
s=i;
}
Printf("last occurence of no. is %d", s);
}
```

```
void main()
{
int a[100], n, b;
Printf("Enter size of array")
scanf("%d", &n);
Printf("Enter array");
for(i=0; i<n; i++)
scanf("%d", &a[i]);
Printf("Enter the element");
scanf("%d", &b);

last(a, n, b);
}
```

(4.)

```
#Include<stdio.h>
void max(int ar[], int n)
{
int i, max, min; c,s;
max = ar[0];
min = ar[0];

for(i=0; i<n; i++)
{
if(ar[i] > max)
{ max = ar[i]; s=i;
}
if(ar[i] < min)
{ min = ar[i]; b=i;
}}
```

```
c = max;        c = ar[s];
max = min;      ar[s] = a[b];
min = c;        ar[b] = c;

Printf("array modified as");
for (i=0; (i<n; i++)
printf("%d", ar[i]);
}
void main()
{
int a[100], n,i;
Printf("Enter the size");
scanf("%d", &n);
Printf("Enter array");
for (i=0; i<n; i++)
scanf("%d", &a[i]);

max(a, n);
}
```

(5.)

```
#include<stdio.h>
void sort(int ar[], int n)
{
int i, j, t;
for(i=0; i<n-1; i++)
{
for(j=0; j<n-i-1; j++)
{
If (ar[j] > ar[j+1])
{
ar[j] = t = ar[j];
ar[j] = ar[j+1];
ar[j+1] = t;
}
}}
Printf("sorted array");
for(i=0; i<n; i++)
printf("%d", ar[i]);
}
void main()
{
int i, a[100], n;
Printf("Enter size of array");
scanf("%d", &n);
Printf("Enter array");
for(i=0; i<n; i++)
scanf("%d", &a[i]);
sort(a, n);
}
```

Date : __/__/__ Page No.: ____

Recursion

A process of function calling function
OR
It is the process of breaking a large problem into sub problem until the problem is small enough to be solved directly now & then we can combine the soln from base level to get back up to the final soln.

Requirement :-

① Base case
The smallest problem for which soln is known

② Recursive case
It is the sub problem that need to be solved

	Recursion	Iteration
①	Process of f^n calling itself	Process of repetition of certain set of statements until condition fails eg. loop control structure
②	It keeps your code simpler & shorter	May have lengthy & complex soln

Date : __/__/__ Page No.: ____

3.	Only have to specify base case & recursive case.	have to specify initialization, condition, execution & updation of loop variable.
4.	It is a memory taking process due to ~~no~~ overhead of maintaining stack data structure	It takes less memory in Iteration.
5.	As it is taking more memory hence, it will take more time to store activation records in stack	It takes less time.
6.	There could be problem of stack overflow	no problem of data maintainance

Q.1 W.A.P. to input two no. x & y & find the division of these no. using recursion without using division operator

Date : / / Page No.:

Ans.
```
#include <stdio.h>
int div(int x, int y)
{
if (x<0 || y<0 || x==0)
return 0;

if(y==0)
return(-1);

else
return(1+div(x-y,y));
}
void main()
{
int x,y,r;
Printf("Enter no to be divide");
scanf("%d",&x);
Printf("Enter divisor");
scanf("%d",&y);

r=div(x,y);

if(r==-1)
Printf("Divisor not possible");

else
printf("Division=%d",r);
}
```

Date : / / Page No.:

Q. W.A.P. to calculate x^y using recursion

```
#include <stdio.h>
int pow(int x, int y)
{
if(y==0)
return 1;

else
return(x*pow(x,y-1));
}
void main()
{
int x,y,z;
Printf("Enter the numbers");
scanf("%d %d",&x,&y);

z=pow(x,y);

if(z==-1)
printf("Indeterminant form");
else
printf("%d",z);
}
```

If (x==0 && y==0)
printf("Not determinant");
return(-1);

Q. W.A.P to calculate gcd of two number.

```
#include <stdio.h>
int gcd(int x, int y)
{ int d
if (x%y==0)
return y;
else
return(gcd(x-y, y));
```

```
#include <stdio.h>
int gcd(int x, int y)
{
If (y==0)
return(x);
else
return gcd(y, x%y);
}
```

```
void main()
{
int x, y, c, d;
Printf ("Enter the numbers");
scanf ("%d %d", &x, &y);
if (x<y)
{
c = x;
x = y;
y = c;
}
d = gcd(x, y);
```

Q: W.A. recursive function to compute the binary of a decimal.

```
#include <stdio.h>
int bin (int x)
{
If (x==0)
return(0);
else if (x==1)
return(1);
else
return (x%2 + 10*bin(x/2));
}
void main()
{
int x; double c;
Printf ("Enter the number");
scanf ("%d", &x);
c = bin(x);
Printf ("binary = %lf", c);
}
```

Date : / / 2026 Page No.:

Q. W.A.P. to check whether no. is prime or not using recursion.

```
#include<stdio.h>
int prime(int a)
{
If(a==0)
return(-1);
else if(a==1||a==0)
{
return(1);
}
else if(a>a)
return(prime(a));
else
return(0);
}
void main()
{
int a,c;
Printf("Enter the number");
Scanf("%d",&a);
c=prime(a);
If(c==0);
Printf("No. is prime");
else
printf("No. is not prime");
}
```

Q. W.A.P. to calculate binomial coefficient using recursion.

```
#include<stdio.h>
#include<math.h>
int binomial(int n, int r)
{
If(r==0||r==n)
return(1);
else
return(binomial(n-1,r-1)+binomial(n-1,r));
}
Void main()
{
int n,r,c
Printf("Enter the value of n & r");
Scanf("%d %d",&n,&r);
c=binomial(n,r);
printf("%d",c);
}
```

Q. W.A.P. for reverse of a number

```
#include <stdio.h>
int reverse(int a)
{
if (a%10 != 0)
return (reverse(a/10) + 10*(a%10));
}
```

a = 1234

rev(123) + 10 * 4

↓

rev(12) + 10 * 3

↓

rev(1) + 10 * 2

↓

1

321

```
#include <stdio.h>
reverse (int n)
{
static int rev;
int r;
if (n)
{
r = n%10;
rev = rev*10 + r;
reverse(n/10);
}
else
return 0;
return rev;
}
void main()
{
int a,b;
Printf("Enter the number");
Scanf("%d", &a);
b = reverse(a);
Printf("Reverse = %d", b);
}
```

Date : __/__/____ Page No.: ____

Q. Find a recursive function for

$$lambda(n) = \begin{cases} 0 & , \text{if } n=1 \\ lambda(n/2)+1 & , \text{else} \end{cases}$$

```
#include <stdio.h>
int lambda (int n)
{
if (n==1)
return 0;
else
return (lambda(n/2) +1);
}
```

Logarithmic function with base 2

ACKERMANN FUNCTION

$$f(m,n) = \begin{cases} n+1 & \text{if } m=0 \\ f(m-1,1) & \text{if } m!=0 \text{ but } n=0 \\ f(m-1, f(m,n-1)) & \text{if } m!=0 \&\& n!=0 \end{cases}$$

Date : __/__/____ Page No.: ____

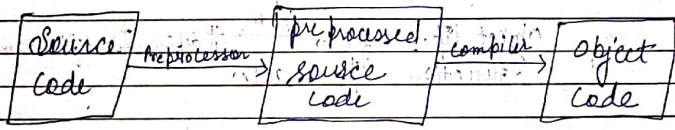
Binomial(m,n)

$$= \begin{cases} 1, & \text{if } n=0 \| m=n \\ \text{else} \\ \text{Binomial}(m-1, n-1) \\ + \text{Bino}(m-1, n) \end{cases}$$

1
1 1
1 2 1
1 3 3 1
1 4 6 4 1

PREPROCESSOR (#)

It is a program which process the source code before giving it to the compiler



Advantages

1. It makes your code readable & easy to understand
2. The code can be easily modified or updated
3. It makes your code efficient to use.
4. It make your code portable, i.e. code can be compiler in different execution

Date : __/__/____ Page No.: ____

environment.

Processor directives

unconditional directives

conditional directives

#if #elif #else #endif

#defined #include #undef

#ifndef #ifdef

(I) Unconditional directives:

(i) #define / macro definition

a) Object like macro — syntax → #define Identifier name string

b) function like macro

Q: W.A.P. using object like macro to calculate the area of circle

```
#include <stdio.h>
# define Pie 3.14
void main()
{
    float a, r;
    printf ("Enter radius");
    scanf ("%f", &r);
    a = 3.14 * r * r;
    printf("%f", a);
}
```

Date : __/__/____ Page No.: ____

```
#include <stdio.h>
# define Pie 3.1427
# define MAIN void main() {
# define F float
# define INPUT printf("Enter radius"); scanf ("%d", &r);
# define OUTPUT Printf ("%f", a);
# define END }

MAIN
F a, r;
INPUT;
  a = Pie * r * r;
OUTPUT;
END
```

Function like MACRO:

syntax #define macro_fun_name (argument1 --arg-n) string

```
#include <stdio.h>
# define imultiply (a,b) a*b
void main()
{
int a = 10, b = 20, c;
 c = multiply (a, b);
Printf ("%d", c);
}
```

Date : __/__/____ Page No.: ____

Q. W.A.P. using function like macro to find largest among two no.

```
#include <stdio.h>
# define largest(a,b) If (a>b)
                        Printf("largest=%d", a);
                      else
                        Printf("largest=%d",b);

void main ( )
{
int a, b;
Printf (" Enter the numbers");
scanf ("%d%d", &a, &b);
largest (a, b);
}
```

Diff. b/w function & Macros

Macro	Function
① It is pre processed	It is compiled
② No type checking	Type checking is done
③ Execution is faster	Execution is slower
④ Pre processed code length may increase for larger macro definitions Hence applicable for only smaller macro definitions	

Reverse of point 4

Date : __/__/____ Page No.: 8

Nesting of macros:

```
# define square(x)        ((x)*(x))
# define cube(x)          (square(x)*x)
# define fourth power(x)  (cube(x)*(x))
```

#include

→ System directory
→ #include<filename.h>

→ #include "filename.h"
or
c
① current directory
then
② obj directory

For path name

```
#include "C:\TC\BIN\A\A1\Fun.c"
```

#undef

```
# define a 3.14
void main ( )
{ int A1, A;
  A = a*2;
# undef a
  A1 = a*5;
}
```

error

Date : / / Page No.

CONDITIONAL COMPILATION

1) #if def

```
#define a 100
void main()
{
#if def a
printf("Hi");
#else
printf("Hello");
#end if
```

```
#define a 100
void main()
{
# if (a==200)
printf("Hi");
# elif (a==500)
printf("Hello");
```

Date : / / Page No.:

```
#else
printf("Bye");
#end if
}
```

```
void main()
{
int a=2;
# if(a==2)
printf("Hi");
#elif(a==500)
printf("Hello");
#else
printf("Bye");
#end if
}
```

any one can executed

MODULE-2

Date :

Pointer :

It is the variable which stores the address of variable.

Advantages :-

* You can access the memory directly using pointer variable.
* Using pointer we can return multiple values from a function.
* We can allocate memory dynamically
* We can change our memory requirements or resize the memory.
* It is efficient & faster in execution
* It saves our memory.
* Using pointers we can create complex data structure

Declaration

Syntax datatype * pointer variable name;

Ex:- int *p;
int a = 10;

Date Page No.:

```
Char ch='a';
Char *ch1=&ch;
```

Q. W.A.P to input two floating point number & add them using pointers

```
#include <stdio.h>
void main()
{
float *a, *b;
float c, d, e;
Printf(" Enter the number"); scanf("%f %f", &c, &d);
a = &c;
b = &d;
e = *a + *b;
Printf(" Sum = %f", e);
}
```

```
#include <stdio.h>
void main()
{
float a, b, c, *a1, *b1, *c1;
a1 = &a;
b1 = &b;
c1 = &c;
```

Date : / /

Page No.:

```
printf("Enter the number");
scanf("%f %f", a1, b1);
*c1 = *a1 + *b1;
Printf("Sum = %f", *c1);
}
```

Q. W.A.P to find largest among three numbers using pointer

```
#include <stdio.h>
void main()
{
int a, b, c, *a1, *b1, *c1;
a1 = &a;
b1 = &b;
c1 = &c;
printf("Enter the numbers");
scanf("%d %d %d", a1, b1, c1);
if (*a1 > *c1)
{
if (*a1 > *b1)
printf("Greatest = %d", *a1);
else
printf("Greatest = %d", *b1);
```

Date : __/__/____ Page No.: ____

```
if (*c1 > *b1)
printf("Greatest = %d", *c1);
```

Q: Factorial using pointer.

Q: reverse of a no.

Q: sum of digits of a no.

(1)
```
#include <stdio.h>
void main()
{
int a, *a1, s=1, i=1, *s1;
   s1 = &s;
   a1 = &a;
printf("Enter the number");
scanf("%d", a1);
for(i=1 ; i<= *a1 ; i++)
*s1 = *s1 * i;
Printf("%d", *s);
}
```

(2)
```
#include <stdio.h>
void main()
{
int a, *a1, r, *r1, s=0 *s1,
   a1 = &a;
   r1 = &r;
   s1 = &s;
Printf("Enter the number");
scanf("%d", a1);
for(i=1 ; *a1>0 ; *a1 = *a1/10)
{
  *r1 = *a1 % 10;
  *s1 = *s1 *10 + *r1 ;
}
Printf("Reverse = %d", *s1);
}
```

(3)
```
#include <stdio.h>
void main()
{
int a, *a1, i, r, s=0 *s1
   a1 = &a;
   s1 = &s
printf("Enter the number");
```

```
scanf("%d", a1);
for(i=1; *a1>0; *a1 = *a1/10)
{
  r = *a1 % 10;
  *s1 = *s1 + r;
}
printf("sum = %d", *s1);
}
```

Robinson

```
#include <stdio.h>
void main()
{
int a, *a1, r, *r1, i, j, s=0, *s1, y, *y1;

a1 = &a;
r1 = &r;
s1 = &s;
y1 = &y;
k1 = &k;
```

14 345

```
printf("Enter the number");
scanf("%d", a1);
for(i=*a1; *a1>0; *a1 = *a1/10)
{
  *r1 = *a1 % 10;
  *s1 = 1;
for(j=1; j<=*r1; j++)
{
*s1 = *s1 * j;
}
*y1 = (*y1 + *s1);
}
If(i == *y)
printf("No. is robinson");
else
printf("No. is not robinson");
}
```

Date : 7/1 Tut VI Page No.:

(1.) Error (type mismatch)

(2.) a = 5

error (the operands are integer & integer (pointer))

a * int(b) = 500

(3.) Error

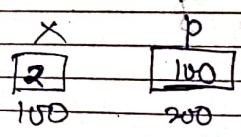
(4.) Error

(5.) d

(6.) 50

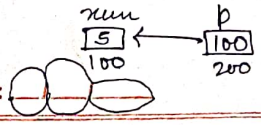
(7.) 51

(8.) Error



*p = 2

(9.) g, 11



Date : 96 Page No.:

10. num = 5 *p = 2 * num

x = 5;

5, 100 6, 7, 5

Disadvantage of pointer

(1.) Pointer increase the complexity of program

(2.) Uninitialised pointer may called segmentation fault.

(3.) In dynamic memory allocation user have to free the memory explicitely otherwise it will lead to memory leak.

(4.) Debugging of program is complex in case of error.

(5.) Hence, it the responsibility of the programmer to use the pointer responsibly.

(6.) There would be problems of memory corruption if we are updating or excessing the data not allocated.

Date : __/__/____ Page No.: ____

Arrays and pointers :-

Q. W.A.P. to search an element in an array using pointers.

```
#include<stdio.h>
void main()
{
int a[100], n, *a1, i, ele, f=0, c;
printf("Enter size of array");
scanf("%d", &n);
a1 = a;
Printf("Enter array");
for(i=0; i<n; i++)
scanf("%d", a1+i);
Printf("Enter the element");
scanf("%d", &ele);
for(i=0; i<n; i++)
{
If (*(a1+i) == ele)
{
f=1;
break;
c=i;
}
printf("Element is found at %d",
}
If (f==0)
printf("Element not found");
```

Date : __/__/____ Page No.: ____

```
else
printf("No. is found at %d", c+1);
```

Sorting

```
#include<stdio.h>
void main()
{
int a[100], n, i, j, t;
int *p=a;
printf("Enter size of array");
scanf("%d", &n);
printf("Enter array");
for(i=0; i<n; i++)
scanf("%d", (p+i));
for(i=0; i<n-1; i++)
{
for(j=0; j<n-i-1; j++)
{
If(*(p+j) > *(p+j+1))
{
t = *p+j;
*(p+j) = *(p+j+1);
*(p+j+1) = t;
}
```

```
printf("Array after sorting");
for (i=0; i<n; i++)
printf("%d", *(p+i));
}
```

Q1. To insert a no.
Q2. To delete a no using index & number
Q3. Find largest in an array with position

(3)
```
#include <stdio.h>
void main()
{
int a[100], n, i, max, c;
int *p = a;

Printf("Enter size of array");
scanf("%d", &n);

Printf("Enter array");
for(i=0; i<n; i++)
scanf("%d", p+i);

max = *p;
```

```
for (i=0; i<n; i++)
{
if (*(p+i) >= max)
{
max = *(p+i);
c = i;
}
}
Printf("max. no. = %d at %d", max, c);
}
```

(2.)
```
#include <stdio.h>
void main()
{
int a[100], n, i, pos;
int *p = a;

Printf("Enter size of array");
scanf("%d", &n);

Printf("Enter array");
for(i=0; i<n; i++)
scanf("%d", p+i);

Printf("Enter position");
scanf("%d", &pos);
```

Date : __/__/____ Page No.: ____

```
for (i=pos ; i<n; i++)
{
*(p+i-1) = *(p+i) ;
}
for (i=0 ; i<n-1 ; i++)
printf ("%d", *(p+i));
```

ii)

```
#include <stdio.h>
void main()
{
int a[100], n, i, ele, c, j;
int *p=a;
Printf ("Enter size");
scanf ("%d", &n);
Printf ("Enter array");
for (i=0; i<n; i++)
scanf ("%d", p+i);
Printf ("Enter Element to be searched");
scanf ("%d", &ele);
for (i=0 ; i<n; i++)
{
if (*(p+i) == ele)
```

Date : __/__/____ Page No.: ____

```
{
c = i ;
break;
}
}
for (j = c+1 ; j<n ; j++)
{
*(p+j-1) = *(p+j) ;
}
for (i=0 ; i<n-1 ; i++)
printf ("%d", *(p+i));
}
```

(i)

```
#include <stdio.h>
void main()
{
int a[100], n, i, ele, pos ;
int *p=a;
printf ("Enter size");
scanf ("%d", &n);
printf ("Enter array");
for (i=0; i<n; i++)
scanf ("%d", p+i);
```

printf ("Enter position and element");
scanf ("%d, %d", &pos, &ele);

```
for (i=n; i>=pos; i--)
{
*(p+i) = *(p+i-1);
}
*(p+pos-1) = ele;
Printf ("array after insertion");
for (i=0; i<n+1; i++)
printf ("%d", *(p+i));
}
```

int a=10, *a1, *b1;

$a_1 = \&a;$

$b_1 = a_1;$

P+4 > P

C = p - a

C = 4

100 in 104

Pointer arithmetic (operations on pointer)

1) Assignment of one address into another (=)

2) Addition of constant value into pointer

3) Subtraction of constant value from pointer

4) Subtraction of one pointer from another

5) Comparision of two pointers (<, >, <=, >=, ==, !=).

6) Increment & decrement operators on pointer

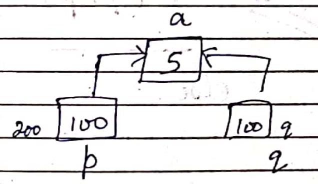
Eg ① int a=5;
int *p, *q;

p = &a;

q = p;

*p = 6

*q = 7



Eg ② p = p+1, → 104

Eg ③ p = p-1 → 96

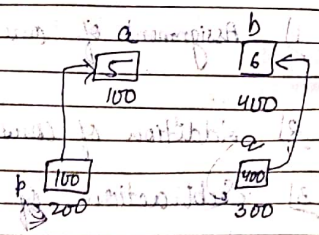
Date : __/__/__ Page No.: ____

eg 4 int a=5, b=6, c;

int *p, *q;

p = &a;

q = &b;



c = p-q

c = q-p

400-100

⇒ 300/4 = 75

Can give no. of integers stored

eg 5 if (q>p)

else

eg 6 p++ ⇒ p = p+4 for integer

We can use pointer upto 32 level

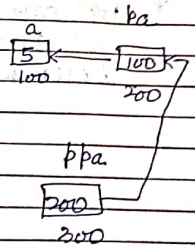
Date : __/__/__ Page No.: ____

Pointer to pointer :

int a=5;

int *pa = &a;

int **ppa = &pa;



Difference b/w pointer & array :-

Array	Pointer
* Collection of similar type of elements	a variable which stores address of another variable
* array name is constant pointer we cannot change this value (we can't use array name as l value)	it is a variable so it can be changed
* size of array gives size of complete array size of data type × No. of elements int a[5] size = 20	int *p = a; size of (p) = 4;

*p= p-2 p-8

Date : / / Page No.:

We cannot apply increament & decreament operator	We can apply on pointer

Date : / / Page No.:

Tutorial

(2) Yes m = (float *) & p but m should be float pointer

(4.) Output - 5

(5.) Output - 2

(6) T-VIII

(4.) Error

5) Error (dynamic memory allocation)

100 200 300 112 116 120

(6.) p = x + 4

1, 2, 3, 4, 5, 6

p = 116

y = 116 - 100 = 16 / 4 = 4

y = 4 Output

Date : / / Page No.:

(7)

10	100	200
100	104	108

ptr = 100

while ptr <= 108)

{

10
100
200

Passing pointer to function

int armstrong (int *);

main()

{

int n;

scanf("%d", &n);

int m = armstrong (&n);

}

int armstrong (int *n)

{

}

Shubhangi

Date : Page No.

By reference

Q1 Check robinson
Q2. Check prime
Q3. Check palindrome
Q4. Check armstrong
Q5. Check perfect no.
Q6. Check frequency
Q7. To count upper & lower case

// with range

(1)

#include <stdio.h>

int rob(int *p, int *q);

{

int , i, c, r, s=0, t, u=1;

for (i=*p ; i<=*q ; i++)

{

c=i;

for (; c>0 ; c=c/10)

{

r = c%10;

u=1;

for(t=1 ; t<=r ; t++)

u = u*t;

s = s+u;

}

If (i==s)

printf ("Robinson number = %d\n", i);

}

}

1. swapping of string
2. swapping of copy of string

string

upper case | lower case | digit | special

Date : __/__/____ Page No.: ____

```
void main()
{
int x, y;
Printf("Enter range");
scanf("%d %d", &x, &y);
prime(x, y);
}
```

(2)

```
#include<stdio.h>
void prime(int *p, int *q)
{
int i, j, f;
for(i=*p; i<=*q; i++)
{
  f=0;
for(j=2; j<=i-1; j++)
{
  if(i%j==0)
    f=1;
}
If(f==0)
Printf("Prime = %d\n", *p);
}
```

Date : __/__/____ Page No.: ____

```
void main()
{
int x, y;
Printf("Enter range");
scanf("%d %d", &x, &y);
prime(x, y);
}

void main()
{
int p, q;
Printf("\n Enter starting & ending point");
scanf("%d %d", &p, &q);
prime
```

Date : __/__/__ Page No.: ____

Pointers & 2-D array:

int array [row] [column];

Eg. int a [2] [3];

int (*p) [3];

p=a;

for (i=0; i<2; i++)

{

for (j=0; j<3; j++)

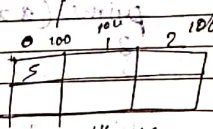
{

scanf ("%d", ((*p+i)+j));

}

}

Printf



p = 100

*p = 100

**p = 5

((*p+0)+0) = 100

*((*p+0)+0) = 5

Q. W.A.P. to input a 2-d array & display it using pointer

```
#include <stdio.h>
void main()
{
int a[100][100], (*p)[100], i, j, m, n;
p=a;
Printf ("Enter row & column");
Scanf ("%d %d", &m, &n);
Printf ("Enter array");
for(i=0; i<m; i++)
{
for (j=0; j<n; j++)
{
scanf ("%d", (*(a+i)+j));
}}
Printf ("array is");
for (i=0; i<m; i++)
for (j=0; j<n; j++)
{
Printf ("%d", *(*(a+i)+j));
}
```

&p[i][j]

*(p+i)+j

i[p]+j

*(i+p)+j

&i[p][j]

Q. W.A.P. to input 2d array & display boundary element using pointer

Q. W.A.P. to input 2d array & find the sum & average of array elements

Q. Addition of two matrix

Q. multiplication of two matrix

Q. Print Z of 2d array
Print N of 2d array

Q. Print both diagonal

Date : __/__/__ Page No.

Tut-7

(3) c
segmentation fault

(6) * str = "hello world";

Output ← hello world

(7) Output – 12

(8) Output – hi

00 01 02
10 20 30
40 50 60
10 11 12

error due to brace

after correcting error

Output → 10 20 30
40 50 60
g g g

Date : __/__/__ Page No.

10 Answer – 3

I – VIU

(1)

100	101	102	103	104	105
P	e	a	c	e	\0

str [100] 200

output – Ce

(2)

100	101	102	103	104	105
I	m	a	i	a	\0

s1 [100] 300

200	201	202	203
B	I	x	\0

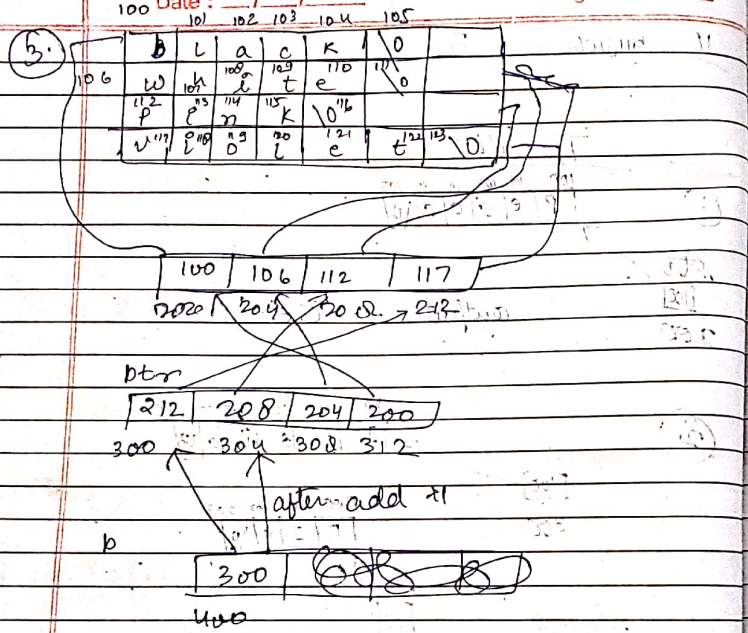
s2 [200] 400

Bodia Bidia BD

~~while (*s1 == *s2)~~

while (I = B)

Date : __/__/____ Page No.: ____



ink

Date : __/__/____ Page No.

Array of pointers :-

Char *s[ ] = {"Hello", "Hi", "Hey"};

Q. W.A.P to input a 2d array of size mxn using array of pointers & display it.

```
#include <stdio.h>
void main()
{
int a[100][50], m,n, i,j;
int *p[100];

for(i=0; i<100; i++)
  p[i] = a[i];

Printf("Enter size");
scanf("%d %d", &m, &n);

printf("Enter 2D array element");
for(i=0; i<m; i++)
{
   for(j=0; j<n; j++)
   {
   scanf("%d", p[i]+j);
   }}
```

Date : __/__/____ Page No. ~~00000~~

```
for (i=0; i<m; i++)
{
  printf("\n");
for (j=0; j<n; j++)
{
   printf("%d \t", *(p[i]+j));
}}
}
```

Q. (1) To enter a array & print sum of all the element

(2) Addition of two ~~matrix~~ matrix

(3) Transpose of matrix

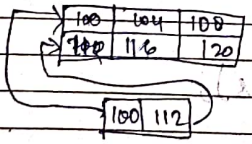
(4) Identity matrix

(5) Print zig-zag

Date : __/__/____ Page No.: _____

Difference b/w pointer to array & array of pointer

Array of pointers	Pointer to an array
(1) Syntax:- datatype *Array name[row size][column size]	Syntax:- datatype (*pointername[column size][row size])
(2) Here size refers to the row size of 2d array & column size may be dynamically allocated	Here size refers to the column of 2D array and row size may be dynamically allocated
(3) Eg: int a[2][3]; int *p[2]; for(i=0; i<2; i++) p[i] = a[i];	int a[2][3]; int (*p)[3]; p=a;



Date : __/__/____ Page No.: ____

(1)

```
#include <stdio.h>
void main()
{
int a[100][500], i, j, m, n, s=0;
int *p[100];

Printf("Enter size");
scanf("%d%d", &m, &n);

for(i=0; i<100; i++)
  p[i] = a[i];
Printf("Enter 2d array");
for(i=0; i<m; i++)
{
for(j=0; j<n; j++)
{
scanf("%d", p[i]+j);
}}

for(i=0; i<m; i++)
for(j=0; j<n; j++)
{
s = s+*(p[i]+j);
}
Printf("sum = %d", s);
}
```

Date : __/__/____ Page No.: ____

(2)

```
#include <stdio.h>
void main()
{
int a[10][10], b[10][10], i, j, c[10][10], m,n;
int *p[10], *q[10], *r[10];

for(i=0; i<10; i++)
{
  p[i] = a[i];
  q[i] = b[i];
  r[i] = c[i];
}

Printf("Enter array size");
scanf("%d%d", &m, &n);
Printf("Enter 1st matrix");
for(i=0; i<m; i++)
for(j=0; j<n; j++)
{
scanf("%d", p[i]+j);
}
printf("Enter 2nd matrix");
for(i=0; i<m; i++)
for(j=0; j<n; j++)
scanf("%d", q[i]+j);

for(i=0; i<m; i++)
for(j=0; j<n; j++)
*(r[i]+j) = *(p[i]+j) + *(q[i]+j);
```

```
for (i=0; i<m; i++)
{
printf ("\n");
for (j=0; j<n; j++)
Printf ("%d \t", *(r[i]+j));
}
}
```

③
```
#include<stdio.h>
void main()
{
int a[10][10], m, n, i, j;
int *p[10];

for (i=0; i<10; i++)
p[i] = a[i];

Printf ("Enter size");
scanf ("%d %d", &m, &n);

Printf ("Enter matrix");
for (i=0; i<m; i++)
for (j=0; j<n; j++)
scanf ("%d", *p[i]+j);
```

```
Printf ("Transpose is");
for (i=0; i<n; i++)
for (j=0; j<m; j++)
printf ("%d", *(p[j]+i));
}
```

④
```
#include<stdio.h>
void main()
{
int a[10][10], m, n, i, j, f=0;
int *p[10];
for (i=0; i<10; i++)
p[i] = a[i];
Printf ("Enter size");
scanf ("%d %d", &m, &n);
Printf ("Enter matrix");
for (i=0; i<m; i++)
for (j=0; j<n; j++)
scanf ("%d", p[i]+j);

for (i=0; i<m; i++)
for (j=0; j<n; j++)
{
if (i==j && *(p[i]+j)==1) || i!=j && *(p[i]+j)==0
f=m;
else if (i!=j && *(p[i]+j)==0)
f=0;
```

```
else
{ f++;
  break;
}
If (f==0)
  Printf ("Matrix is an Identity matrix");
else
Printf("Matrix is not identity matrix);
}
```

⑤

```
#include <stdio.h>
void main()
{
int a[10][10], i, j, m, n;
int *p[10];

for(i=0 ; i<10; i++)
  p[i] = a[i];

Printf ("Enter size");
scanf ("%d%d", &m, &n);

Printf ("Enter array");
scanf

for (i=0; i<m ; i++)
for (j=0; j<n ; j++)
{
scanf ("%d", *p[i]+j);
}
```

```
for (i=0 ; i<m ; i++)
for (j=0 ; j<n ; j++)

for (i=0 ; i<m ; i++)
{ Printf ("\n");
If (i%2==0)
{ for (j=0 ; j<n; j++)
Printf ("%d", *(p[i]+j));
}
else
{ for (j=n-1 ; j>=0 ; j--)
  Printf ("%d", *(p[i]+j));
}
}
}
```

Date : / / Page No.:

Function pointer / pointer to function:

A function pointer is a pointer which holds (points to function) address of a function of same signature or prototype.

deceleration

return type (*function pointer name) (argument list);

E.g.
```
int (*fp) (int, int);
int add (int a, int b)
{
return (a+b);
}

Int main ()
{
fp = add;
int b = fp(5,7);
}
```

Date : Page No.:

E.g.
```
void print(int n);
void (*fp)(int );

void main ()
{
 fp = print; // fp = &print;
 fp(10);
}
void print(int n)
{ printf ("value of n = %d", n);
}
```

E.g.
```
int add (int a; int b)
{
 return (a+b);
}
int subtract (int a, int b)
{
 return (a-b);
}
int multiply (int a, int b)
{
 return (a*b);
}
```

```
int (*fp)(int,int)
void main()
{ int result;
 fp = add;
result = fp(7,9);
Printf("%d", result);
fp = multiply;
result = fp(7,9);
Print("%d", result);
fp=subtract;
result= fp(9,7);
Printf("%d", result);
}
```

Passing function into another function :

```
int add (int a, int b);
int sub (int a, int b);
int operate (int (*fp)(int,int), int a, int b);
void main()
{
result;
result= operate (add, 5, 7);
printf("Result = %d", result);
}
int operate (int (*fp)(int,int), int a, int b)
{
int result;
result = fp(a,b);
return result;
}
int add (int a, int b)
{
 return (a+b);
}
int sub (int a, int b)
{return (a-b);
}
```

Advantage

* Reduce code redendency repeating something
* easy to understand

Date : / / Page No.:

Function returning function pointer:

```
int add (int ,int);
int sub (int ,int);
int (*(getptr(char)))(int,int);
   or
typedef   int (*ptr)(int,int)
ptr(getptr(char));

void main()
{
int (*fp) (int, int);
fp = getptr('+');
result = fp(7,9)
Printf("sum = %d", result);
}
ptr getptr(char ch)
{
if (ch = = '+')
return add;

if (ch = = '-')
return sub;
}
```

Date : / / Page No.:

```
int add (int a,int b)
{
return (a+b);
{
int sub (int a,int b)
{
return (a-b);
}
}
```

Q. W.A.P. to check whether a no. is perfect no. or not using function pointer.

Q. W.A.P. for prime no.

Q. To input array & check how many prime numbers are there.

Q. Reverse of a string.

(1.)

```
#include <stdio.h>
void perfect (int n)
{
int i, j=n, s=0;
for (i = 2 ; i<n ; i++)
{
if (n%i ==0)
{ s = s+i;
```

```
        n = n/i;
        i = i-1;
    }
}
if(j == s+1)
    Printf("No. is perfect");
}

void (*fp)(int n)
void main()
{ int n;
  fp = perfect;
  Printf("Enter no.");
  Scanf("%d", &n);
fp(n);
}
```

Tut IX

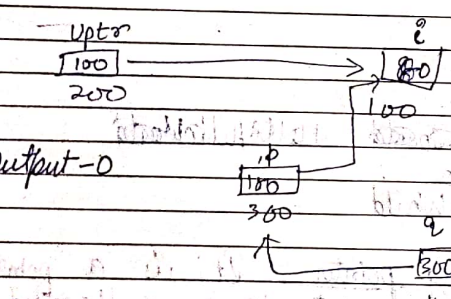
1) k = 101

k = k+1
k = 102
*[k+2] = m

output :- 4

2) Output :- 10

3) Output - 0



4) Output :- 30, -10

5) 24

6) 5
5

Date : Page No.:

(7) 50

(8) 15, 5, 50, 2

Tut – x

(1) 100

(2) Hello

(3) 0

(4) Hello
HelloWorld

(5) Dangling pointer :- It is a pointer which points to a deallocated memory block.

To resolve this problem assign p as NULL after deallocating memory.

(6) Error

(7) 10

Date : / / Page No.:

(8) Hello
World

Q. W.A.P. to input a 2d integer array of size m×n & display it using DMA.

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

int main()
{
int *p, m, n, i, j;
printf("Enter size");
scanf("%d %d", &m, &n);

p = (int*) calloc (m*n, sizeof(int));
                          If(p==NULL)
printf("Enter elements");  { Printf("Not alloc ated");
for(i=0; i<m; i++)          exit(0);
{                          }
for(j=0; j<n; j++)
{
scanf("%d", *(p+i*n+j));
scanf("%d", (p+i*n+j));
```

Date : __/__/____ Page No.:

```
for(i=0; i<m; i++)
for(j=0; j<n; j++)
printf("%d", *(p+i*n+j));
}
```

Date : __/__/____ Page No.: ____

Module - III

structures:-

A structure is a collection of elements under the same name. The elements could be hetrogeneous or homogeneous and the memory will be allocated in contigious manner for these elements.

Declaration:- (User defined or derived data type)

```
struct structure_name
{
  data type 1  mem_name1;
  data type 2  mem_name2;
  .
  .
};
```

Struct structure name var_name1, var_name2 ---;

Date : __/__/____ Page No.: ____

Ex:-
```
struct student
{
    int roll;
    char name[50];
    float fees;
    char course[50];
    char dob[50];
};

struct student stud1;
```

Q. Define a structure for a point in coordinate system for a plane

```
struct point
{
    int x, y;
};
```

~~struct point p;~~

Date : __/__/____ Page No.: ____

Initialisation

syntax:-
```
struct structure_name
{
    data type 1  mem1;
    data type 2  mem2;
    .
    .
};
struct structure_name Var_name = {constant, constant};
```

Ex:-
```
struct student
{
    int roll;
    char name[50];
    float fees;
    char course[50];
    char dob[50];
};

struct student stud1 = {101, "Shyam", 50000.5, "b.tech"};
```

Date : __/__/____ Page No.: ____

typedef :-

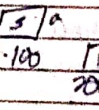
It is used to give a new name to an existing datatype. It is also a user defined datatype.

Syntax :-

typedef existing datatype name New datatype name;

Ex:- typedef int INT;

int a=5;
INT b=10;



Ex:- typedef struct student {old structure name} (optional)
{
int roll;
char name[50];
float fees;
char name[50];
char dob[50];

Date : __/__/____ Page No.: ____

new
{ structure name;

new structure name var name;

Ex. typedef struct
{
=
=
=
}
student;
student std1;

typedef

Accessing members of structures

syntax :- structure var name . member name;

struct student
{
int roll;
char name[50];
float fees;
} std1 = {100, "RAM", 50000};

Date : __/__/____ Page No.: ____

Printf ("%d", stud1.roll); ⟶ 100

Printf ("%s", stud1.name);
float ("%f", stud1.fee);

Input scanf :- • has more priority that roll

scanf ("%d", &stud2.roll());

scanf ("%s", stud2.name);
scanf ("%f", &stud2.fee);

Q. W.A.P to input the details of a student of your class & then display them using structures

```
#include <stdio.h>
void main()
{
struct student
  {
    int roll;
    char name[50];
Char Course [50];
} stud1;
```

Date : __/__/____ Page No.: ____

```
printf ("Enter roll no.");
scanf ("%d", &stud1.roll);
printf ("Enter name");
gets (stud1.name);
printf ("Enter course");
gets (stud1.course");

Printf ("Roll no. = %d", stud1.roll);
Printf ("name is");
puts (stud1.name);
Printf ("Course is");
puts (stud1.course);
}
```

Anonymous structure :-

Syntax :

```
struct
{
   data type 1  mem1;
   data type 2  mem2;
   .
   .
} a;
```

Date : __/__/____ Page No.: ____

Copying & comparing structures :-

To copy one structure variable information into another we simply have to use assignment operator to copy on variable name.

But to compare two structure variable we have to compare every member individually.

Ex:-

```
struct POINT
{
  int x, y;
} P1, P2 = {5, 7};

P1 = P2;

if ( P1.x == P2.x && P1.y == P2.y)
Printf ("Equal");
else
Printf ("Not equal");
```

P1 | 5 | 7 P2 | 5 | 7

Date : __/__/____ Page No.: ____

Q.1 W.A.P. to check in which quadrant a point lies in a coordinate system for a plane using structure

Q2. W.A.P. to find the distance b/w two points in a coordinate system for a plane

Q3. W.A.P. to find the largest among three numbers using structure

Q4. W.A.P. to find the difference between two times.

Q5. W.A.P. to add two times.

Q6. W.A.P. to input two complex numbers & find their addition, subtraction, multiplication acc. to the user using switch case.

Q7. W.A.P. to add two distance in km & m

Date : __/__/____ Page No.: ____

(1)
```
#include <stdio.h>
void main()
{
 struct coordinate
 {
   int x, y;
 }a;

 Printf("Enter coordinates");
 scanf("%d %d", &a.x, &a.y);
 If(a.x>0)
 {
 If(a.y>0)
 Printf("Point is in first quadrant");
 else
 Printf("Point is in fourth quadrant");
 }
 If(a.x<0)
 {
 If(a.y>0)
 Printf("Point is in second quadrant");
 else
 Printf("Point is in ~~fourth~~ third quadrant");
 }}
```

dist

Date : __/__/____ Page No.: ____

```
If(a.x==0 && a.y==0)
Printf("Point is on origin");
}
```

(2)
```
#include <stdio.h>
#include <math.h>
void main()
{
struct distance
{
 int x, y;
}a,b;

Printf("Enter first coordinates");
scanf("%d %d",&a.x, &a.y);
Printf("Enter second coordinates");
scanf("%d %d", &b.x, &b.y);

~~Printf("distance = %f", sqrt((a.x-b.x)^2+(a.y-b.y)^2))~~
}
Printf("distance = %f", sqrt(pow(a.x-b.x,2)+pow(a.y-b.y,2));
                       (float)
}
```

Date : __/__/____ Page No.: ____

(3.)
```
#include <stdio.h>
void main()
{
struct largest
{
int a, b, c;
} d;

Printf("Enter 3 numbers");
scanf("%d %d %d", &d.a, &d.b, &d.c);

If (d.a > d.b)
{
If (d.a > d.c)
Printf("Maximum number = %d", d.a);
else
Printf("Maximum number = %d", d.c);
}

If (d.b > d.c)
Printf("Maximum number = %d", d.b);

else
Printf("Maximum number = %d", d.c);
}
```

Date : __/__/____ Page No.: ____

(4.)
```
#include <stdio.h>
void main;
{
int H, M, S;
struct time
{
int h, m, s;
} a,b;

Printf("Enter ending time");
scanf("%d %d %d", &a.h, &a.m, &a.s);

Printf("Enter starting time");
scanf("%d %d %d", &b.h, &b.m, &b.s);

if (b.s > a.s)
{
a.s = a.s+60;
a.m = a.m-1;
}
S = a.s - b.s;

if (b.m > a.m)
{
a.m = a.m+60;
a.h = a.h-1;
}
M = a.m - b.m;

H = a.h - b.h;
```

Date : / / Page No.

```
Printf ("Time = %d h, %d m %d s", H, M, S);
}
```

(5.)

```
#include <stdio.h>
void main()
{
int H, M, S;

struct Time
{
int h, m, s;
} a, b;

Printf ("Enter starting time");
scanf ("%d %d %d", &a.h, &a.m, &a.s);

Printf ("Enter ending time");
scanf ("%d %d %d", &b.h, &b.m, &b.s);

~~if(a+b >= 60)~~

S = a.s + b.s;
If (S >= 60)
{
S = S - 60;
b.m++;
}

M = b.m + a.m;
```

Date : / / Page No.

```
If (M >= 60)
{
M = M - 60;
b.h++;
}

H = a.h + b.h;

Printf ("Total time = %dh %dm %ds", H, M, S);
}
```

(6.)

```
#include <stdio.h>
void main()
{
int K, M;

struct distance {
int Km, m;
} a, b;
Printf ("Enter first distance");
Scanf ("%d %d", &Km, &m);

Printf ("Enter second distance");
Scanf ("%d %d", &b.km, &b.m);

M = a.m + b.m;
If (M >= 10000)
{
M = M - 10000;
a.km++; }
```

Date : __/__/____ Page No.: ____

```
K = a.km + b.km;
Printf("Total distance = %.d km %.d m", K, M);
}
```

(6)

```
#include <stdio.h>
void main()
{ int e;
struct complex
{
int a, b;
} c, d;
Printf("Enter real & complex part of first no.");
scanf("%.d %.d", &c.a, &c.b);
Printf("Enter real & complex part of second no.");
scanf("%.d %.d", &d.a, &d.b);
Printf("1. Addition \n 2. Subtraction \n
        3. Multiplication \n 4. Division");
Printf
Scanf("Enter choice");
scanf("%.d", &e);
```

Date : __/__/____ Page No.: ____

```
switch(e)
{
Case 1 : Printf("Real part = %.d \n complex part
          = %.d", (c.a+d.a), (c.b+d.b));
         break;
Case 2 : Printf("Real part = %.d \n complex part = %.d",
          (c.a-d.a), (c.b-d.b));
         break;
Case 3 : Printf("Real part = %.d \n complex part = %.d",
          (c.a*d.a - c.b*d.b), (c.a*d.b
          + c.b*d.a));
         break;
Case 4 : Printf("%.d i %.d", ((c.a*d.a)+(c.b*d.b))
          /(d.a*d.a + d.b*d.b),
          (c.b*d.a - c.a*d.b)
          /(d.a*d.a + d.b*
          d.b));
default :
}
}
```

Date : __/__/____ Page No.: ____

D.O.B

```
1. #include <stdio.h>
   void main()
   {
      struct time
         int d,m,y;
   } t1, t2, t3;
   printf("Enter your D.O.B");
   scanf("%d %d %d", &t1.d, &t1.m, &t1.y);
   printf("Enter current date");
   scanf("%d %d %d", &t2.d, &t2.m, &t2.y);
   if (t2.d > t1.d)
   {
      t3.d = t2.d - t1.d
   }
   else{
         t2.d = t2.d + 30;
         t2.m = t2.m - 1
         t3.d = t2.d - t1.d;
      }
      if (t2.m > t1.m)
      {
         t2.m = t2.m - t1.m;
      }
else{ t2.m = t2.m + 12;
      t2.y = t2.y - 1;
      t3.m = (t2.m - t1.m); }
      t3.y = t2.y - t1.y
printf("%d %d %d", &t3.d, &t3.m, &t3.y);
}
```

Date : __/__/____ Page No.: ____

Nested structure

Syntax

```
struct inner_structure name
   {
   data type 1  mem1;
   data type 2  mem2;
   :
   };

struct outer_structure name
   {
      data type 1  mem1
      data type 2  mem2
   struct  inner_structure name  member name;
      :
      :
};
struct  outer_structure name  var. name;
```

Accessing :-

Outer_structure var. name. Outer structure mem-name. inner structure mem-name.

Date : __/__/____ Page No.: ____

1. W.A.P. to input the details of student taking D.O.B & name of student as nested structure for student structure.

```
#include<stdio.h>
 void main()
{
 struct DATE
 int dd, mm, yy;
 };
struct NAME
{
Char f-name[20], m-name[20], l-name[20];
};

struct student
 {
   int roll;
struct NAME name;
   float fees;
struct DATE dob;
  int marks[6];
};

struct student stud1;
Printf("Enter student details\n");
 Printf("Enter roll no.");
scanf("%d", &stud1.roll);
Printf("Enter name");
Scanf("%s %s %s", stud1.name.f_name, stud1.name.m_name, stud1.name.l_name);
```

Date : __/__/____ Page No.: ____

```
Printf("Enter DOB");
Scanf("%d %d %d", &stud1.dob.dd, &stud1.dob.mm,
                  &stud1.dob.yy);
Printf("Enter fees");
Scanf("%f", &stud1.fees);
for(i=0; i<6; i++)
scanf("%d", &stud1.marks[i]);
}
```

2. W.A.P. to input details of book ISBN no., Title of book, no. of pages, price, Publication year, author name, author address.

```
#include<stdio.h>
 Void main()
{
Struct year
{
int yy;
};
 struct NAME
{
char f_name[20], m_name[20], l_name[20];
};
 Struct BOOK
 {
  int ISBN;
 Struct NAME name;
 Struct Year Publication year;
```

```
    int Pages
    };
struct Book B;

Printf ("Enter book details");
printf ("Enter ISBN");
Scanf (" %d", & ISBN);
printf ("Enter author name");
scanf (" %s %s %s", b.name.f_name, b.name.m_name,
                    b.name.l_name);
}
```

Date : __/__/__ Page No.:

Array of structure

Syntax:

```
struct structure_name
{
 data type 1  name 1;
 data type 2  name 2;
};
struct structure_name array_name [SIZE]
```

Ex: WAP to input the details of all the student of your class.

```
struct student
{
int roll;
char name [50];
float fee;
int marks [6];
} stud [100];
int i, n;
printf ("Enter no. of students");
scanf ("%d", &n);
Printf ("Enter details");
for (i=0; i<n; i++)
{
Printf ("Enter roll no. of student = %d", i+1);
scanf ("%d", & stud [i].roll);
printf ("Enter name");
```

Date : __/__/__ Page No.:

```
gets ( &stud[i].name );
Printf("Enter fees");
scanf ("%f", &stud[i].fees);
Printf ("Enter marks");
for (j=0; j<6; j++)
scanf ("%d", &stud[i].marks[j]);
}
```

Date : __/__/__

Scanned by CamScanner

Date : Page No.:

Structure & pointer :-

```
struct structure name
{int data type1 mem1;
 data type 2 mem2;
};
struct structure name var_name, *pointer name
                                =&var_name;
printf("%d", var_name.mem1);
printf("%d", (*pointer name).mem1);
```

or

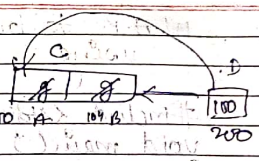
```
printf("%d", pointer name -> mem1);
```

(->) Pointing to operator

Date : / / Page No.:

Eg

```
struct ABC
{
 int A;
 int *B;
} C, *D = &C;
(*D).A = 10; // D->A = 10;
D->B = &D->A;
printf("%d", C) ------> 10
```



C.A ------> 10	D->A=10
&C ------> 100	D->100
&C.A ------> 100	*D->10
C+1 ------> 11	(*D).A->10
C.A+1 ------> 11	(*D).B->100
&C+1 ------> 108	D->B=100
	&D->B->1

Date : __/__/____ Page No.: ____

Q. W.A.P. to input the details of a student using pointer to structure.

```
#include <stdio.h>
void main()
{
struct student
{
int rollno.;
char name[50];
float fees;
char courses[50];
} a, *c=&a;
Printf("Enter roll no.");
Scanf("%d", &c->rollno.);
Printf("Enter name");
gets(c->name);
Printf("Enter fees");
scanf("%f", &(c->fees));
Printf("Enter courses");
gets(c->courses);
}
```

Date : __/__/____ Page No.: ____

```
Printf("Roll no. of student");
printf("%d", c->rollno.);
Printf("name is");
puts(c->name);
Printf("fees is");
printf("%f", c->fees);
printf("Course is");
puts(c->courses);
}

#include <stdio.h>
void main()
{
struct student
{
int roll;
char name[50];
float fees;
char course[50];
} a[40], *c=&a;

for(i=0; i<40; i++)
{
printf("Enter roll no. of %d student", i+1);
scanf("%d", &c[i]->roll);
Printf("Enter name of %d student", i+1);
gets(c[i]->name);
```

(c+i) → name or c[i].name

Date : __/__/____ Page No.: ____

```
Printf("Enter fees of %d student", i+1);
scanf("%f", &c[i] -> fees);
                &(c+i) -> fees
Printf("Enter course of %d student", i+1);
gets( c[i] -> course);
      (c+i) -> course
}
for(i=0; i<40; i++
```

Date : __/__/____ Page No.: ____

Questions :-

a)

```
#include <stdio.h>
void main()
{
struct student
{
 int roll;
char name[20];
char department[20];
char course[20];
int yoj;
} b[40];
 int n,i, m,j;
Printf("Enter no. of students");
scanf("%d", &n);
for(i=0; i<n; i++)
{ Printf("Enter details of %d students", a(i+1));
Printf("Enter roll no, department, course, year of joining");
scanf("%d", &b[i].roll);
scanf gets(b[i].department);
      gets(b[i].course);
scanf("%d", &b[i].yoj);
}
Printf("Enter the particular year");
scanf("%d", &m);

for(i=0; i<n; i++)
{
```

```
if (b[i].yoj == m)
{
puts (b[i].name);
}

for Printf("Enter roll no.");
  scanf("%d", &j);

for (i=0 ; i<n; i++)
{
if( b[i].roll == j)

Printf ("roll no. = %d", b[i].roll);
Printf ("name is");
puts ( b[i].name);
Printf ("department is");
puts (b[i].department);
Printf ("Course is");
puts ( b[i].course);
}
```

Date : / / Page No.:

Structure and functions :-

Method for passing structure into function:
- Passing individual member by call by value
- Passing whole structure by " "
- Passing individual member by call by reference
- Passing whole structure " "

① passing individual member by value

Ex:

```
void fun(int a, int b)
{
printf ("x-coordinate = %d \n Y coordinate %d", a, b);
}

void main()
{
struct POINT
{
 int x, y;
}P, B = {5, 6};
fun( P.x, P.y);
}
```

Date : __/__/____ Page No.: ____

(2.) Passing complete structure by value :-

```
struct POINT
{
int x,y;
};
Void fun(struct POINT);
void main()
{
struct POINT P1={5,6};
   fun(P1);
}
Void fun(struct POINT P)
{
Printf("X-coordinate=%d\n Ycoordinate=%d",P.x,P.y);
```

(3.) Passing individual structure members by reference:

```
struct Point POINT
{
int x,y;
};
Void fun(int *, int *);
```

Date : __/__/____ Page No.: ____

```
main()
{struct POINT P1={5,6};
 fun(&P1.x, &P1.y);
}
void fun(int *a, int *b)
{
Printf("X-coordinate=%d\nYcoordinate=%d",*a,*b);
}
```

(4.) Passing complete structure by reference :-

```
struct POINT
{
int x,y;
};
Void fun(struct POINT *)
main()
{
struct POINT P1={5,6};
fun(&P1);
}
Void fun(struct Point *P)
{
printf("X-coordinate=%d\n Ycoordinate=%d"
                    (*P).x,(*P).y);
}
```

Date : __/__/____ Page No.: ____

Q. W.A.P to Input details of students & display them by structure & functions.

```
#include<stdio.h>

struct student
{
int roll;
char name[50];
float fees;
};

Void display (struct student);

void main()
{
struct student stud;
printf ("Enter roll no, name & fees of student");
scanf ("%d%c%[^\n] %f", &stud.roll, stud.name,
                        &stud.fees);

display (stud);
}
void display (struct student stud1);
{
printf ("roll no. = %d \n name = %s \n Fees = %f",
        stud1.roll, stud1.name, stud1.fees
```

Date : __/__/____ Page No.: ____

Ex:

```
struct student
{
int roll;
char name[50];
float fees;
};
void display(struct student*, int);

void main()
{
struct student stud[100];
int n, i;
printf ("Enter no. of students");
scanf ("%d", &n);

for(i=0; i<n; i++)
{
printf ("Enter details of student no. = %d", i+1);
scanf ("%d%c%[^\n] %f", &stud[i].roll, stud[i].name
display(stud, n);          &stud[i].fees);
}
Void display (struct student *stud, int n)
{
int i;
for(i=0; i<n; i++)
{
printf ("Roll = %d \n Name = %s \n
```

Date : / / Page No.

UNION :-

→ It is also a user defined datatype.
→ It is created using keyword union

Note:- In union memory is allocated for the field which is largest in size

```
#include<stdio.h>
union student
{
 int roll;
 float marks;
 char name[30];
}s;

void main()
{
printf("Enter roll no.");
scanf("%d", &s.roll);
printf("%d", s.roll);
scanf printf("Enter marks");
      scanf("%f", &s.marks);
      printf("%f", s.marks);
printf("Enter name");
gets(s.name);
puts(s.name);
}
```

Date : / / Page No.

	Structure	Union
①	Structure is created using keyword struct	Union is created using keyword union
②	In structure memory is allocated for all the members	In union memory is allocated for largest members
③	In structure we can simultaneously access all the members	We cannot access all the members simultaneously

④	Syntax
	struct student
	{
	};
	
		Syntax
		union student
		{
		};

```
#include<stdio.h>
union student
{
 int roll;
 float marks;
char name[30];
}s[10];

void main()
{
for(i=0; i<10; i++)
{
```

Date : __/__/____ Page No.: ____

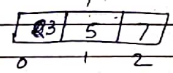
```
Printf ("Enter details of student %d", i+1);

Printf ("Enter roll no. of student %d", i+1);
scanf ("%d", &s[i].roll);
Printf ("Roll no. of student %d is %d", i+1,
                                    s[i].roll);

Printf ("Enter marks of student %d", i+1);
scanf ("%f", &s[i].marks);
Printf ("Marks of student %d is %f", i+1,
                                    s[i].marks);

Printf ("Enter name of student %d", i+1);
gets (s[i].name);
Printf ("Name of student %d", i+1);
puts (s[i].name);
}
}
```

Date : __/__/____ Tut-XII Page No.: ____

1. 6, 6 — a_{1b} [6]

2) 5 6

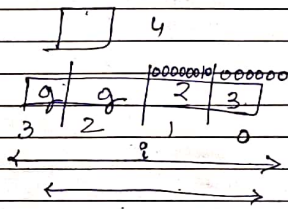
3) 3, 5, 7



6. (3) (2)

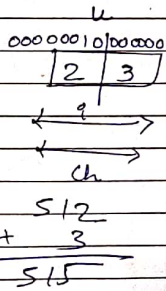
$u.ch[0] = 3$
$u.ch[1] = 2$
$u.i$ = garbage

64 bit



32 bit (3) (2)

$u.ch[0] = 3$
$u.ch[01] = 2$
$u.i = 515$



512
+ 3
515

Date : Page No. 9

(9.) Yes, it work

(10.) error

(11.) 5

Date : / / Page No.:

Q. W.A.P. to enter either name, roll no of a student and his marks in 6 subjects & display it using union within structure.

```
#include <stdio.h>
struct A
{
int marks[6];
union B
{
char name[50];
int roll;
};
};
void main()
{
int choice;
printf("1. Enter name \n 2. Enter roll no.");
scanf("%d", &choice);

switch(choice)
{
case 1 : printf("Enter name");
         gets(c.d.name);
         break;

case 2 : printf("Enter roll no.");
         scanf("%d", &c.roll);
```

```
        break;
default : Printf ("Invalid choice");
}

Printf ("Enter marks");
for(i=0; i<6 ; i++);
Scanf (" %d", c.marks[i]);

Printf for(i=0; i<6; i++)
{
Printf (" Marks of subject %d is %d", i+1, marks[i]);
if (choice == 1)
puts (c.D.name);
else
Printf (" %d", c.D.roll);
}
```

Q. W.A.P. to input either roll no. or name and DOB of a student structure with in union

Ans

```
#include <stdio.h>
Union A
{
int roll;
Char name[50];
struct A
{
int dd, mm, yy
}a;
}b;
void main()
{
int choice;
Printf ("1. Enter name \n 2. Enter roll no.");

void main()
{
Int choice
Printf (" 1. Enter name \n 2. Enter roll no.");
Printf (" Enter choice");
scanf (" %d", & choice);

switch (choice)
{
Case 1 : Printf (" Enter name");
         gets (b.name);
         breaks;

Case 2 : Printf (" Enter roll no.");
         scanf (" %d", & b.roll);
         break;

default : Printf (" invalid choice");
}
Printf (" Enter date of birth");
Scanf (" %d %d %d", & b.a.dd, & b.a.mm, & b.a.yy);
}
If
```

Date : __/__/____ Page No. :

(2.)

```
#include <stdio.h>
union student
{
  struct record
{
char name[50], dob[50];
}s;
long int roll;
}s;

Void main()
{
Char choice;
Printf(" Do you want to enter roll no.? ");
Scanf("%c", &choice);
If( choice == 'y' || choice == 'Y')
Scanf("%ld", &s.roll);
else
{
Printf(" Enter name");
gets(s.r.name);
Printf(" Enter dob");
gets(s.r.dob);
}
```

Date : __/__/____ Page No. :

```
If (Choice == 'y' || Choice == 'Y')
{
Printf("Roll no. = %ld", s.roll);
}
else
{
Printf("Name is ");
puts(s.r.name);
Printf(" DOB is ");
puts(s.r.dob);
}

}
```

Date : __/__/____ Page No.: ____

Enumeration (Enum) :-

It is a user-defined data type, it is used for assigning names to the integer constants, these names makes the program easy to read & maintain it

Syntax :-

enum enumeration_name {name1, name2, ---}
0 1 ---

or

enum enumeration_name {name1, name2---};

enum enumeration_name var_name;

or

typedef enum enumeration_name {name1, name2---} new_name;

new_name var_name;

Anonymous enums

enum {name1, name2 ---} var_name;

Date : __/__/____ Page No.: ____

E.g.
```
#include<stdio.h>
enum state {OFF, ON} check_state;
                0   1
check_state = OFF;   // 0
check_state = ON;    // 1
enum days {MON, TUE, WED, THUR, FRI, SAT, SUN} var_name;
           0    1    2    3    4    5    6
start, end;
start = TUE;
end = SAT;
pf("%d %d", start, end);   //15
```

Q. W.A.P. to input the choice of colour of a user among red, blue & green & display all the shades of the colour?

```
main()
{
enum colour {RED, BLUE, GREEN};
enum color col;
printf("Enter 0 for Red 1 for blue &2.Green");
scanf("%d", &col);
```

Date : __/__/____ Page No.: ____

```
switch (col)
{
case RED :  Printf ("Shades are orange, Pink, maroon\n");
            break;

case BLUE : Printf ("Shades are sky blue, Navy blue, sky blue\n");
            break;

case GREEN : Printf ("Shades are Parrot green, Dark green, Light green");
}
}
```

Some important points regarding Enum:-

① enum days {MON, TUE, WED, THU, FRI, SAT, SUN} day_name;
0 1 2 3 4 5 6

② enum days {Mon=1, TUE, WED, THU, FRI, SAT=15, SUN} day_name;
1 2 3 10 11 15 16

enum days {MON=1, TUE, WED, THU=1, FRI, SAT, SUN} day_name;
1 2 3 1 2 3 4

```
day_name=MON;
day_name = THU;
```

Date : Page No.:

④ enum state {OFF, ON} state; ✓ No error

FILE HANDLING

Need of file handling :-

* We need file for storing information permanently & retrieving it later on as per our need so that it could be processed by programs.

It is not enough to display the data on screen as memory is volatile so, if we need this data again & again it would have to entered through the keyboard or through programatically so to overcome this problem we will store the data in files & will retrieve it through file.

* Feeding or inputting a bulk amount of data again & again is a time consuming process & then is a chances of inputting erroneous data.

Date : __/__/____ Page No.: ____

File

Collection of data stored on a secondary storage device such as hard disk.

Data :-

Data is a raw fact. It may be figure, such as numbers, words, sentences, paragraph etc.

Eg: John, 3years, good boy

Information :-

Data arranged in a meaningful manner so as to make sense is called information

Eg: John aged 3 years is a good boy.

Types of file

- Text files (ASCII files)
- Binary files

Date : __/__/____ Page No.: ____

Text file

When data is stored in the form of readable & printable characters then file is known as text file & it can be accessed sequentially.

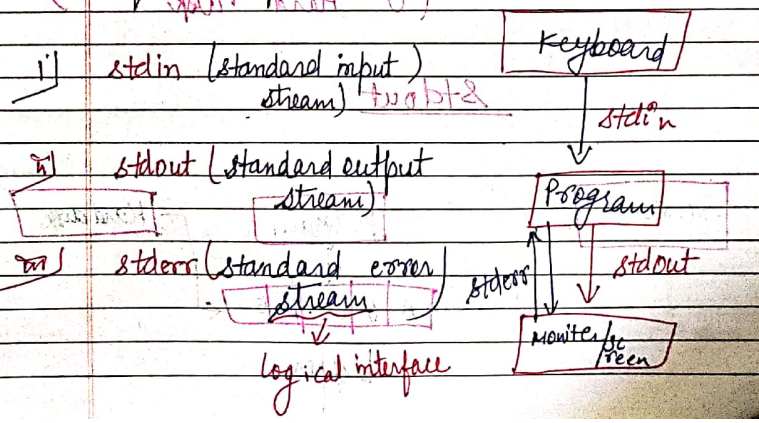
Binary file :-

If a file contains data in machine language form then the file is known as binary file & it can be accessed randomly.

Stream :-

A logical interface to different types of computer devices.

1) stdin (standard input stream)
2) stdout (standard output stream)
3) stderr (standard error stream)

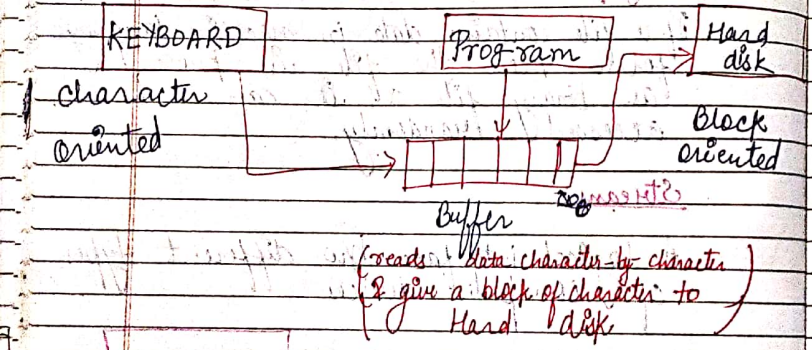


Date : / / Page No.:

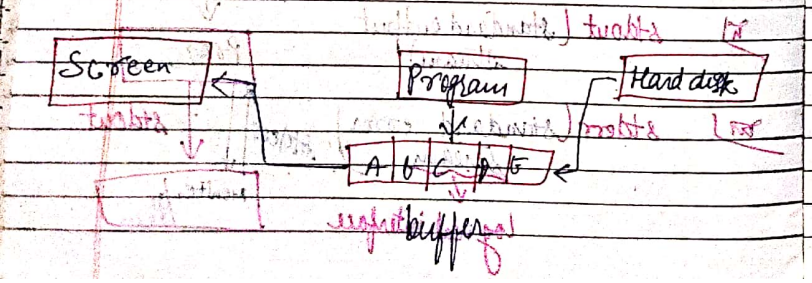
Buffer

A buffer is a temporary storage which is used for implementation of different types of stream.

stdin



stdout



Date : / / Page No.:

Steps needed to process a file:-

1. Create a FILE pointer
2. Open the file
3. process the file
4. Close the file

Different type of operations on a file

1. creation of a file
2. Read data from the file
3. Search data into file (seeking)

(1) To create a FILE pointer & open the file:-

Syntax: FILE *fp;

for open:

Syntax: fp = FILE* fopen (const char * filename, const char * file mode);

Date: Page No.:

text → r – read
w → write
a → to append
r+ → read & write
w+ → " "
a+ → " "

binary → rb, wb, ab, rb+, r+b, wb+, w+b, ab+, a+b

FILE *fp

Ex. fp = fopen ("file name.txt", "r");
PATH → read mode

If (fp == NULL) {
printf ("Can't open file");
exit(0);
}

*** Different types of file opening mode

	text file	binary
1.	r	rb
2.	w	wb
3.	a	ab
4.	r+	rb+, r+b
5.	w+	wb+, w+b
6.	a+	ab+, a+b

r :- reading from a file, if file exist file will be opened through fopen function & file pointer will point to the first character of file.

If file doesn't exist fopen will return NULL.

w :- writing into a file, if file exist the data will get overwrite, if file doesn't exist it will create the file.

If file doesn't open fopen will return NULL.

a :- appending data into a file, if file exist file pointer will point to the last character in the file & data will be written in the file from the last character. If file doesn't exist then it create a file.

If the file doesn't open it will return NULL.

r+ - reading & writing into a file.

Used for reading from a file before writting to it. Sets up a pointer which will point to the first character & return null if unable to open.

Date : / / Page No.:

w+ writing new content reading them back and modify existing type.

If the file exist contents are over written

If file doesn't exist return NULL character

a+ :- Reading existing content appending new content to the end of file cannot modify existing content

Sets up a ponter which points to the last character.

If file doesn't exist return null.

rb :- Used for reading from binary file.

wb :- writing in binary file.

ab :- used for appending data for binary file

Date : / / Page No.:

rb+ used for reading & write in binary file

wb+ " " " " "

ab+ " " " " "

2) Processing a file :-

	Input / Reading data from a file	writing data / output into file
1)	fgetc ( )	fputc ( )
2)	fgets ( )	fputs ( )
3)	fscanf ( )	fprintf ( )
4)	fread ( )	fwrite ( )

3 Closing a file :-

Syntax :- int fclose (FILE *stream);

Ex: FILE *fp, *fp1, *fp2;

fp = fopen ("ABC.txt", "r");
fp1 = fopen ("DEF.txt", "w");
fclose (fp);
fclose (fp1); or fcloseall ( );

Date : __/__/__ Page No.: ____

fclose :-

fclose does the following three steps

i) writes the data into the file from buffer to disk

ii) It will mark end of file character in the file in your disk

iii) The memory to the buffer is will be eleminated.

Troubles in opening a file :-

There are following troubles for opening a file in read mode.

1. If file doesnot exist fopen will return null
2. If file is protected

Following in write mode (Problems)

1. If disk is damaged or corrupted
2. If file is write protected
3. If there is no disk space & file also doesnot exist it will not create a file

*** **Difference b/w binary file & Text file :-**

	Text file (ASCII File)	Binary file
1.	When data is stored in the form of printable & readable characters file is called Text file	If data is stored in machine language form (0, 1)
2.	It can be accessed sequentially	It can be accessed sequentially as well as randomly.
3.	The end of file is maintained by EOF marker	There is no such character present in the binary mode file to mark the EOF. The binary mode file keep track of EOF from the no. of characters present in the directory entried in the file.

Date : __/__/____ Page No.: ____

(4.)	The size of file in disk is larger than the size of file in memory	In binary file the size of file in disk as well as in memory is same
Ex:	123465 will take 6B in disk but only 2 or 4 B in memory	Ex: 123456 → 2 or 4 B in memory, 6 B in disk
(5)	Inefficient method to store the file	Efficient method to store the file

Reading and writing functions in files

- Reading: fgets(), fgetc(), fscanf(), fread()
- Writing: fputc(), fputs(), fprintf(), fwrite()

(1) fgetc() :-

Syntax :- int fgetc (FILE *stream);

Date : __/__/____ Page No.: ____

(2) fputc() :-

Syntax :- int fputc (int char, FILE *stream);

Q. W.A.P. to read data from a file char. by char. & display it on to the screen

```
#Include <stdio.h>
#Include <stdlib.h>
main ()
{
FILE *fp;
Char ch
fp = fopen ("ABC.txt", "r");
If (fp == NULL)
{
Printf (" Can't open file ");
exit (0);
}
ch = fgetc (fp);
While (ch != EOF)
{
Printf ("%c", ch);
ch = fgetc (fp);
}
fclose (fp);
}
```

Date : __/__/__ Page No. :

Q. W.A.P. to read a data from a file char by char & count the no. of characters, no. of space, no. of new line, no. of tabs.

```
#include <stdio.h>
#include <stdlib.h>
void main()
{
FILE *fp;
int ns=0, nn=0, nt=0, nc=0;
char ch
fp= fopen("DEF.txt", "r");

if (fp==NULL)
{
printf("Can't open file");
exit(0);
}
ch=fgetc(fp);
while (ch!=EOF)
{
nc++;
if (ch==' ')
ns++;
else if (ch=='\n')
nn++;
else if (ch=='\t')
nt++;
ch=fgetc(fp);
printf("No. of spaces=%d\n
No. of character=%d\n
No. of new line=%d\n
No. of tab=%d\n"
, ns, nc, nn, nt);
fclose(fp);
}
```

Date : __/__/__ Page No. :

Q. W.A.P. to read the data from a file char by char & print all the uppers in the lower & all the lower case letter into the upper case.

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

void main()
{
FILE *fp
char ch

fp= fopen("ABC.txt", "r");
If (fp==NULL)
{
Printf("Cant open file");
exit(0);
}

ch= fgetc(fp);
while (ch!=EOF)
{
If (ch>='A' && ch<='Z')
ch= ch+32;
else If (ch>='a' && ch<='z')
ch= ch-32;
else Printf("%c", ch);
ch= fgetc(fp);
}
fclose(fp);
}
```

Date : __/__/____ Page No.: ____

Q: W.A.P. to compare two files are identical or not character by character.

```
#include <stdio.h>
#include <stdlib.h>
void main()
{
FILE *fp1, *fp2;
char ch1, ch2;
int f=0
fp1 = fopen("ABC.txt", "r");
fp2 = fopen("DEF.txt", "r");

If (fp1==NULL || fp2==NULL)
    {
    Printf ("file can't open");
    exit(0);
    }

ch1 = fgetc (fp1);

ch2 = fgetc (fp2);

while (ch1!=EOF && ch2!=EOF && ch1==ch2)
{
ch1 = fgetc (fp1);
ch2 = fgetc (fp2);
}
```

Date : __/__/____ Page No.: ____

```
If (ch1 == ch2)
Printf ("Identical");

else if (ch1 > ch2)
Printf ("File1 > File2");
else
Printf ("FILE1 < File2");

fcloseall();
}
```

Q: W.A.P. to provide the feed back of a book from user & write it into a file character by character.

```
#include <stdio.h>
#include <stdlib.h>
main()
{
FILE *fp;
char feedback[50];
int i=0
fp = fopen("DEF.txt", "w");
if (fp==NULL)
{
Printf ("can't open file");
exit(0);
}
```

Date : __/__/____ Page No.: ____

```
Printf("Enter feedback");
gets(feed back);
for (i=0; feedback[i]!='\0'; i++)
{
  fputc(feedback[i], fp);
}
fclose(fp);
}
```

Q1: To read data from one file & write it into another file char by char

Q2: WAP to read data from a file char by char & write digit in other file, alphabet in other file & special character in other file

(1)
```
#include<stdio.h>
#include<stdio.h>
void main()
{
FILE *fp1, *fp2;
char ch;
fp1 = fopen("ABC.txt", "r");
fp2 = fopen("DEF.txt", "w");
If (fp1 == NULL)
{ Print("Can't open file");
} exit(0);
```

Date : __/__/____ Page No.: ____

```
ch = fgetc(fp1);
while (ch != EOF)
{
 fputc(ch, fp2);
 ch = fgetc(fp1);
}
 fcloseall();
}
```

(2)
```
#include<stdlib.h>
#include<stdio.h>
void main()
{
FILE *fp1, *fp2, *fp3, *fp4;
char ch;
fp1=fopen("ABC.txt", "r");
fp2 = fopen("Digits.txt", "w");
fp3 = fopen("alphabet.txt", "w");
fp4 = fopen("SC.txt", "w");
ch = fgetc(fp1);
while (ch != EOF)
{
If (ch>='0' && ch<='9')
 fputc(ch, fp2);
```

If (fp1 == NULL)
{Print("Can't open file");
exit(0);
}

Date : / / Page No.:

```
If (ch>='a' && ch<='z' || ch>='A' && ch<='Z')
  fputc(ch, fp3);
else
  fputc(ch, fp4);
ch = fgetc(fp1);
} fcloseall();
}
```

③ fgets( ):

Char*

Syntax:- fgets(char* str, int size, FILE *stream);

④ fputs( ):-

int

Syntax:- fputs(char *str, FILE *stream);

0 → Successful

or

EOF → -1

Date : / / Page No.:

Q.1 W.A.P. to display the content of a file on screen. multiple character simultaneously

```
#include <stdio.h>
#include <stdlib.h>
void main()
{
FILE *fp;
char s[10];
fp=fopen("abc.txt","r");
If (fp==NULL)
{
Printf("can't open file");
exit(0);
}
while (fgets(s, 10, fp)!=NULL)
{
  printf("%s", s);
} fclose(fp);
}
```

Date : __/__/____ Page No.: ____

Q. W.A.P. to copy one file into another file simultaneously.

```
#include <stdio.h>
#include <stdlib.h>
Void main()
{
FILE *fp1, *fp2;
Char ch[5]; int f=0;
fp1 = fopen("ABC.txt", "r");
fp2 = fopen("DEF.txt", "w");

If (fp1==NULL || fp2==NULL)
{
Printf("Can't open file");
    exit(0);
}

while (fgets(ch, 5, fp1) != NULL)
{
   fputs(ch, fp2);
}
fcloseall();
}
```

Date : __/__/____ Page No.: ____

Q. W.A.P. to compare two file are identical or not.

```
#include<string.h>
#include <stdio.h>
#include <stdlib.h>
void main()
{
FILE *fp1, *fp2;
Char c[5], s[5]; int f=1;
fp1 = fopen("ABC.txt", "r");
fp2 = fopen("DEF.txt", "r");

If (fp1==NULL || fp2==NULL)
{
Printf("Can't open file");
exit(0);
}

while (fgets(c,5,fp1) != NULL && fgets(s,5,fp2) != NULL && strcmp(ch1,ch2)==0)
{
   f=0;
}
If (strcmp(ch1, ch2)==0)
Printf("Identical");
else
Printf("Not identical");
}
```

Date : __/__/____ Page No.: ____

```
fgets (ch1, 5, fp1);
fgets (ch2, 5, fp2);
while (ch1 != NULL && ch2 != NULL &&
       strcmp (ch1, ch2) == 0)
{
  fgets (ch1, 5, fp1);
  fgets (ch2, 5, fp2);
}
If (strcmp(ch1, ch2) == 0)
  printf ("Identical");
else
Printf ("Not identical");
```

Q: W.A.P. which receives multiple strings from the user & write them in file.

```
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
void main()
{
FILE *fp1;
Char ch[20];
fp1 = fopen ("ABC.txt", "w");

if (fp1 == NULL)
{printf ("Can't open file");
exit(0); }
```

Date : __/__/____ Page No.: ____

```
while (strlen (gets(ch)) > 0)
 {
 fputs (ch, fp1);
}
fclose (fp1);
}
```

Q. W.A.P. to merge two file into the third file multiple characters simultaneously

```
#include <stdio.h>
#include <stdlib.h>
void main()
{
FILE *fp1, *fp2, *fp3;
char ch1[20], ch2[20]

fp1 = fopen ("ABC.txt", "r");
fp2 = fopen ("DEF.txt", "r");
fp3 = fopen ("Merge.txt", "w");

If (fp1 == NULL || fp2 == NULL || fp3 == NULL)
{
Printf ("Can't open file");
exit(0);
}
```

Date : / / Page No.:

```
while (fgets(ch1, 20, fp1) != NULL)
{
fputs(ch1, fp3);
}
while (fgets(ch2, 20, fp2) != NULL)
{
fputs(ch2, fp3);
}
fcloseall();
}
```

5] fscanf :- Used to read formatted data from file

Syntax :- int fscanf(FILE *stream, const char *format, address of variables);

→ EOF (-1) → EOF / → error
→ 0 (successful)

6] fprintf() :- used for writing formatted data into file

Syntax :-

int fprintf(FILE *stream, const char *format, Name of variable);

→ (-1) EOF → error
→ 0 → (successful)

Q. W.A.P. to input the name & roll no. of a student from keyboard & display it & then input the name & roll no. of the student from file & display it.

```
#include<stdio.h>
void main()
{ char name[20]; int roll;
FILE *fp;
fp = fopen("ABC.txt", "w+");
If (fp == NULL)
{
Printf("Can't open file");
exit(0);
}
Printf("Enter name & roll");
scanf("%[^\n]%d", name, &roll);
or
fscanf(stdin, "%[^\n]%d", name, roll);
fprintf(fp, "%s\n%d", name, roll);
printf("Name = %s \n Roll = %d", name, roll);
```

Date : __/__/____ Page No.: ____

or

fprintf (stdout, "Name=%s \n Roll=%d", name, roll);

rewind (fp);

fscanf (fp, "Name = %[^\n] \n Roll=%d" name, &roll);

fprintf (stdout, "Name=%s \n Roll=%d", name, roll);

fclose (fp);

Q. W.A.P. to read details of all employees (Name, age, salary) from a file & display it on to the screen.

```
#include <stdlib.h>
#include <stdio.h>
void main
{
FILE *fp;

fp = fopen ("ABC.txt", "r");
if (fp == NULL)
{
printf ("Can't open file");
exit(0);
}
```

Date : __/__/____ Page No.: ____

```
struct employee
{
char name[50];
int age;
float salary;
}emp;
while (
fscanf (fp, "%s %d %f\n", emp.name, &emp.age, &emp.salary)
{                                  != EOF)
fprintf (stdout, "Name = %s \t Age = %d \t Salary = %f\n",
                 emp.name, emp.age, emp.salary);

fclose (fp);
}
```

Q. W.A.P. to input the name of name & salary of 10 employee & printf data into file.

```
#include <stdlib.h>
#include <stdio.h>
void main()
{
FILE *fp
int i
```

Date : __/__/____ Page No.: ____

```
fp=fopen("ABC.txt","w");
If (fp==NULL)
{
Printf("Can't open file");
exit(0);
}

struct employ
{
char name[20];
float salary;
}e[10];
for(i=0; i<10; i++)
{Printf("Enter data of employ %d",i+1);

Printf("Enter name");
gets(e[i].name);
Printf("Enter salary");
scanf("%f",&e[i].salary);
}
for(i=0; i<10; i++)
{
fprintf(fp,"%s\t %f",e[i].name,e[i].
                                salary);
}
fclose(fp);
}
```

Date : __/__/____ Page No.: ____

Q. W.A.P to read details of student & then print it on the screen as well as write it into file.

```
#include <stdio.h>
#include <stdlib.h>
void main()
{
FILE *fp;
fp=fopen("ABC.txt","w");
int i;
struct student
{
Char name[20], dob[20];
int roll;
}a[10];

for(i=0; i<10; i++)
{
Printf("Enter details of student %d",i+1)
Printf("Enter name");
gets(a[i].name);
Printf("Enter dob");
gets(a[i].dob);
Printf("Enter roll no.");
scanf("%d",&a[i].roll);
}
for(i=0; i<10; i++)
{
fprintf(fp,"%s\n%s\n%d\n",a[i].name,
                a[i].dob, a[i].roll);
}
```

Date : __/__/____ Page No.: ____

```
for(i=0; i<n; i++)
{
fscanf (fp, "%s\n %s\n %d\n", a[i].name,
                              a[i].dob, &a[i].roll);

fprintf (stdout "%s\n %s\n %d\n" a[i].name,
                              a[i].dob, a[i].roll);
}
}
```

Date : __/__/____ Page No.: 6

Random access of file having records :-

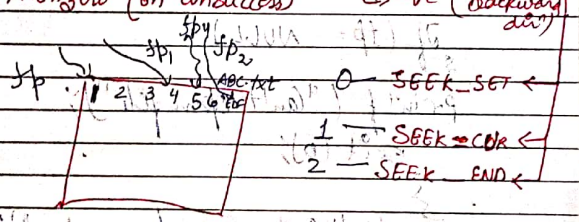
3 functions :-

1) **fseek( )** :- To reposition file stream (pointer)

syntax :- int fseek (FILE *stream, long offset, int origin);

→ 0 (on success)
→ Non zero (on unsuccess)

offset → +ve (forward dirn)
→ -ve (backward dirn)



1. fseek (fp, 0, SEEK_SET) → at the begin
2. fseek (fp1, 0, SEEK_CUR) → at the current position
3. fseek (fp2, 0, SEEK_END) → At the end of file
4. fseek (fp, 2, SEEK_SET) → 3
5. fseek (fp1, -3, SEEK_CUR) → 2
7. fseek (fp2, -4, SEEK_END) → 3

Date : __/__/____ Page No.: ____

Q1. W.A.P. to randomly read the nth record from a employee file

```
#include <stdio.h>
#include <stdlib.h>
void main()
{
FILE *fp
int r
fp=fopen ("ABC.txt", "r");
If (fp==NULL)
{
printf ("Can't open file");
exit (0);
}
Printf ("Enter record number");
Scanf ("%d", &r);
If (r>=1)
{
fseek (fp, (r-1) * size of (emp) + 3*(r-1), SEEK_SET);
fscanf (fp, "%d %s %f", &emp.emp_code,
                        &emp.name, &empsalary);
fprintf (stdout, "Empcode %d \n Name = %s \n
                  Salary = %f", emp.emp_code,
                               emp.name, emp.salary);
}
fclose (fp);
}
```

Q2. W.A.P. to edit a record in employee file

Q3. W.A.P. to print details in reverse order.

(3)

```
#include <stdio.h>
#include <stdlib.h>
void main()
{
FILE *fp; int c=1, a;
fp=fopen ("ABC.txt", "r");
If (fp==NULL)
{
pt ("Can't open file");
exit (0);
}
```

```
struct employee
{
int emp_code;
Char name[20];
float salary;
} emp;
```

Date : __/__/____ Page No.: ____

```
struct employee
{
int emp_code;
char name[20];
float salary;
}emp;

while(1)
{
a = fseek(fp, -1*sizeof(emp)-3*1, SEEK_END);
if(a!=0)
  break;
fscanf(fp,"%d %s %f", &emp.emp_code,
                      emp_name, &emp_salary);
fprintf("Name = %s\n Code = %d\n Salary = %f",
        &emp_name, emp.empname,
                   emp_salary);
```

ftell();

It tells current position of FILE stream/pointer in no. of bytes.

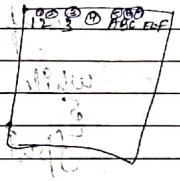
Date : __/__/____ Page No.: ____

Syntax:-

long int ftell(FILE *stream);

-1 (character no.)

long a = ftell(fp);



Drawbacks of ftell:-

1) Using ftell with a device that cannot store data. Ex:- keyboard

2) When the position is larger that cannot be represented in long int, that happens in very large file.

Q. W.A.P. which writes data to a file saves the no. of bytes stored in it in a variable n & then reopen the file to read n bytes from the file & then simultaneously display it.

```
#include <stdio.h>
#include <stdlib.h>
Void main()
{ char ch;
FILE *fp; long int n;
fp = fopen("ABC.txt", "w+");
```

Date : __/__/____ Page No.: ____

```
If (fp==NULL)
{
pf ("can't opin file");
exit(0);
}
scanf ("%c", &ch);
while (ch != '*')
{
fputc (ch, fp);
scanf ("%c", &ch);
}
n=ftell (fp);
rewind (fp);
while (ftell (fp) <= n)
{
ch = fgetc (fp);
printf ("%c", ch);
}
```

Date : __/__/____ Page No.: ____

Advance concept in C

1. Bit field
2. Command line arguments
3. Variable length argument
4. Type qualifier - const, volatile, restrict

Slag bytes || Padding bytes

```
struct A
{
char c;
int a;
int b;
float d;
char e;
}A1;
printf ("%d", sizeof (A1));
```

Output 16

```
struct A
{
char e;
char c;
int a;
float d;
}A1;
```

Output 10

Bit field

Use :- Bit fields are used with structure the main use of bit-fields are for memory utilisation / memory saving when we already know the range of that particular member.

Date : / / Page No.:

Syntax :-

```
struct structure_name
{
  data_type  bit_field_name : width;
                or
             mem_name       : no. of bits
  int, char, long, short, signed int, unsigned int etc.
  but not float
};
```

Rules for bit field :-

1. C doesn't permit array of bit field, pointer to bit field, function returning bit field
2. C only permits use of bit fields only within structure
3. The & operator cannot be applied with bit field

Eg:

```
struct A
{
  unsigned int a : 2;
  unsigned int b : 3;
  signed int c : 2;
} A1 = {4, 6, -1};
Printf("%d %d %d", A1.a, A1.b, A1.c);
```

Output 0 -2 -1

COMMAND LINE ARGUMENT

There is need to provide inputs while giving the execution command we can do it with the help of command line arguments without proving the input again & again after running the programs

Syntax :-

```
int main(int argc, char *argv[])
{
```

argc : argument count
argv : argument vector

```
}
```

Date : __/__/____ Page No.: ____

Q. W.A.P to display the argument supplied from command line

```
#include <stdio.h>
int main (int argc, char * argv[ ])
{
int i ;

for ( i=0; i<argc ,i++)
printf ("Argument %d = %s", i+1, argv[i]);
return 0;
}
```

Q. W.A.P. to find the largest amount three integer numbers supplied from command line

```
#include <stdio.h>
void main (int argc, char * argv[ ]);
{
int a = atoi (argv[1]), b= atoi(argv[2])
c=atoi(argv[3]);
if (* argv[0] > * argv[2] )
if ( * argv[0] > * argv[2])
if (a>b)
if (a >c)
Printf ("Greatest number is %d", a);
else
Printf ("Greatest number is %d", c);
}
if (b>c)
Printf (" b is greater");
}
```

Q. W.A.P. to check whether the user supplied argument less than 2, greater than 2 or equal to 2

```
#include <stdio.h>
void main (int argc, char * argv[ ]);
{
if (argc <2)
Printf ("Less than two arguments");
else if (argc ==2)
Printf ("Two arguments");
else
Printf ("More than two arguments");
}
```

Q. W.A.P. to input multiple file names & display the content of all these file on string.

```
#include<stdlib.h>
#include<stdio.h>
void main (int argc, char *argv[]);
{
int i = argc-1; j = 1;
FILE *P; char s[10];
printf ("Data of %d no. of files", argc);
while (c)
{
P = fopen (argv[j], "r");
If (P==NULL)
{ Printf ("Can't open file");
exit(0);
}
while (fgets (s, 10, P) != NULL)
{
Printf ("%s", s);
}
fClose P;
c--;
}
}
```

Variable length argument list:

<Stdarg.h>
↳ standard argument

1) va_list } object like macro
2) va_start }
3) va_arg } function like macro
4) va_end }

Sometimes the user may come across a situation when you want to have a function which can take variable no. of arguments i.e. parameters instead of pre-defined no. of parameters.

The C allowed to a soln for this & define a fun^n which can accept variable argument

```
Void fun(int num, ...)
main()
{
fun(3, 1, 2, 3)
fun(4, 1, 2, 3, 4)
```

↳ 3 dots (ellipses)

Date : __/__/____ Page No.: ____

It should be noted that the function fun has its last argument as ellipsis & first argument is the total count of arguments to perform or to implement variable length argument we will use <stdarg.h>

(1) Create a va_list type of variable which create a plot or array

(2) We will use va_start macro to initialise the va_list

(3) Use va_arg macro & va_list to access argument one by one

(4) Use va_end to macro to clean up the memory assign to va_list

Q. W.A.P. to find the average of n numbers supplied from variable (named) argument list

Date : __/__/____ Page No.: ____

```
#include<stdio.h>
#include<stdarg.h>
double average(int num, ...);
void main()
{
printf("sum = %f", average(3, 5, 8, 9));
printf("sum = %f", average(4, 5, 7, 3, 4));
}
double average(int num, ...)
{ double sum=0.0; int i;
  va_list list;
  va_start(list, num);
for(i=0; i<num; i++)
sum = sum + va_arg(list, int);
  va_end(list);
return(sum/num);
}
```

R = R+1

Date : __/__/____ Page No.: ____

Type qualifiers :-

The keywords which used to modify the properties of a variable are called Type qualifiers.

3 types

1) Const
2) restrict
3) volatile

1) Const :- (constant)

keyword → const

The object variable can only be used as R-value if qualified with const

A constant variable must be initialised when it is declared because it can't be changed later.

Ex:-

```
main()
{
const float PI;
PI = 3.1421
}
```
✗ Not valid

```
main()
{
const float PI = 3.1421;
}
```
Valid

Date : __/__/____ Page No.: ____

Using const with pointers

1) making pointer as constant
2) making value where pointer points as constant
3) making both pointer & value as constant

Eg 1)
```
main()
{ const int b = 10;
Const int a = 5;
int *const p = &a;
p = &b; ✗ not valid
*p = 10; valid
}
```

2.
```
main()
{
Const int b = 10;
Const int a = 5;
int Const *p;
p = &a;
p = &b;
*p = a ; ✗ Not valid
}
```

Date : / / Page No.:

3. main()
{
const int b=10;
const int a=5;
int const * const p = &a;

p = &b; // Not valid
*p = 10; // Not valid
a = a+2; // Not valid

2. Restrict :-

Used on pointer only. & indicates that the pointer is only the initial way to access the dereference data. It provides more help to the compiler for optimization.

main()
{
int a=10;
restrict int *p = &a;
int *q = &a;
*p = *p+4;

*q = *q * 3; // Not valid
*p = *p+5;
}

optimization
*p = *p+9

Date : / / Page No.:

3. Volatile :-

Volatile qualifiers tells the compiler that an object value may be changed by entities other than the program.

In C to optimise the C.P.U. time when compiler owns the object it can store & handle it anyway to optimise the program.

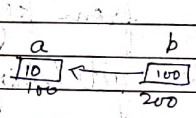
As a result C - compiler may think that it is more efficient to remove an object from RAM & put it into a register.

Ex:
main()
{
volatile int a = 10;
int *p = &a;

printf("%d", *p); → 10

*p = 100

printf("%d", *p); → 100
}



output for gcc A.C → 10, 100
" " gcc -O A.C → 10, 10
after volatile gcc -O A.C → 10, 100

Tut

	gets()	fscanf()
①	only used to read string data from keyboard.	used to read any type of data (formatted data) from file or keyboard
②	It will only read one inputs at a time	It can read multiple input at the same time
③	It will only read the data from keyboard	It will read from file, keyboard.
④	gets(string);	fscanf(FILE*stream, "format specifier", add. of variable);
⑤	Ex-	Ex-