C programming

Array :-

Array is a collection of Homogenius data type.

Array is Allocate static Memory organization (we can't Manuplate Memory).

In Array we're using sequencial Memory location.

Type of Array :-

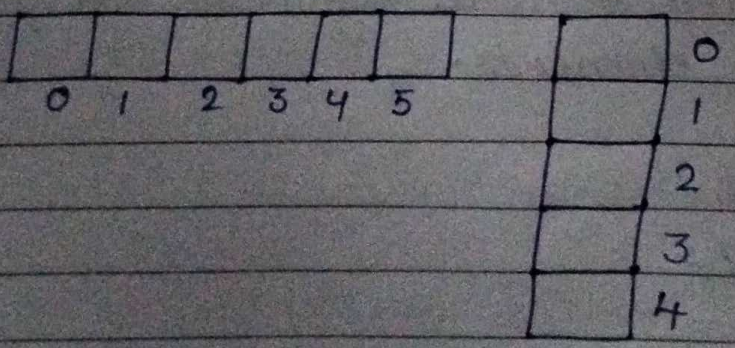
(i.) One dimensional Array
(ii.) Two dimensional Array
(iii.) Multi dimensional Array

one dimensional Array :- In this Array we can represent our data either in a row or column only.

Syntax :-

data type < array_name > [size];

example :- int arr[5];



Que WAP to show the element of array.

Ans. #include<stdio.h>

```
#include<conio.h>
  void Main()
  {
   int i;
  int a[5] = {11,12,13,14,15};
  clrscr();
  printf ("\n show the element of Array");
   for(i=0; i<5; i++)
    {
   Printf ("\n a[%d] = %d", i, a[i]);
    }
   getch();
    }
```

Output
a[0] = 11
a[1] = 12
a[2] = 13
a[3] = 14
a[4] = 15

Que WAP to input data to the keyboard in Array, and show the elements.

=)
```
#include<stdio.h>
#include<conio.h>
  void Main()
  {
  int i, a[5];
   clrscr();
  printf ("Enter the value in Array \n");
```

```
for (i=0; i<5; i++)
{
scanf ("%d", &a[i]);
}
for (i=0; i<5; i++)
{
printf ("\n a[%d] = %d", i, a[i]);
}
getch();
}
```

Output :- a[0] = 11
a[1] = 12
a[2] = 13
a[3] = 14
a[4] = 15

Que WAP for addition of array elements

Ans.
```
#include <stdio.h>
#include <conio.h>
void main()
{
int i, a[5], sum=0;
clrscr();
printf ("Enter the value in Array \n");
for (i=0; i<5; i++)
{
scanf ("%d", &a[i]);
sum = sum + a[i];
}
```

printf ("\n the sum of Array element is = %d",
sum);

getch();
}

Sorting in Array :-

we have a two types of sortings in Array.

(1) Ascending order
(2) Descending order

For the sorting we're using a different techniques and algorithms like bubbles shot, Insertion sort, Selection sort, Merge sort, ~~&~~ Heap sort etc.

∴ sorting :- Arranging data in proper way.

Bubble sort :-

step 1 :-

1	2	3	4	5	6
42	35	12	77	5	101

step 2 :-

1	2	3	4	5	6
35	12	42	5	77	101

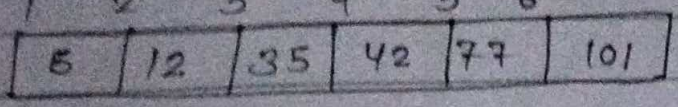
step 3 :-

1	2	3	4	5	6
12	35	5	42	77	101

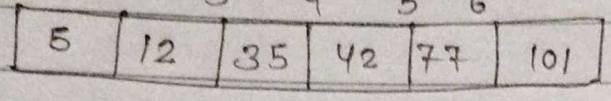
step 4 :-

1	2	3	4	5	6
12	5	35	42	77	101

Step 3:-



Step II:-



Que WAP to show the 5 subjects Marks :-

=)
```
#include<stdio.h>
#include<conio.h>
  void main()
  {
  int i, a[5], tot=0, per; //m1, m2, m3, m4, m5=
                                    a[5]

   clrscr();
   printf("Enter the Marks of five subjects");
   for(i=0; i<5; i++)
    {
   scanf("%d", &a[i]);
   tot = tot + a[i];
   }
    per = (tot/5.0);
    printf("total Marks = %d", tot);
   printf("\n Percentage = %d", per);
    getch();
    }
```

Output :- Enter the Marks of Five subjects : 78
89
56
45
34
total Marks = 302
percentage = 60

Note :- Array using Homogenius Data type.

Note :- position = Index + 1

Bubble.c

```
#include<stdio.h>
#include<conio.h>
#define MAX 20
 void Main()
  {
     int arr[MAX], i, j, k, temp, n, xchanges;
    Clrscr();
    printf("enter the number of elements:");
    scanf("%d", &n);
    for (i=0; i<n; i++)
    {
      printf("enter element %d:", i+1);
      scanf("%d", &arr[i]);
    }
    /* Middle class */
   For (i=0; i<n-1; i++)
   {
    xchanges = 0;
    For (j=0; j<n-1-j; j++
    {
    if (arr[j] > arr[j+1])
    {
       temp = arr[j];
   arr[j] = arr[j+1];
   arr[j+1] = temp;
```

```
xchanges ++;
}/* End of if */
}/* End of inner for loop*/
if (xchanges == 0) /* If list is sorted */
   break;
printf ("After pass %d elements are :", i+1);
 for (k=0; k<n; k++)
      printf ("%d", arr[k]);
  printf ("\n");
}/* End of outer for loop*/
printf ("Sorted list is : \n");
for(i=0 ; i<n; i++)
      printf ("%d ", arr[i]);
  printf ("\n");
  getch();
}/* End of main()*/
```

Output :

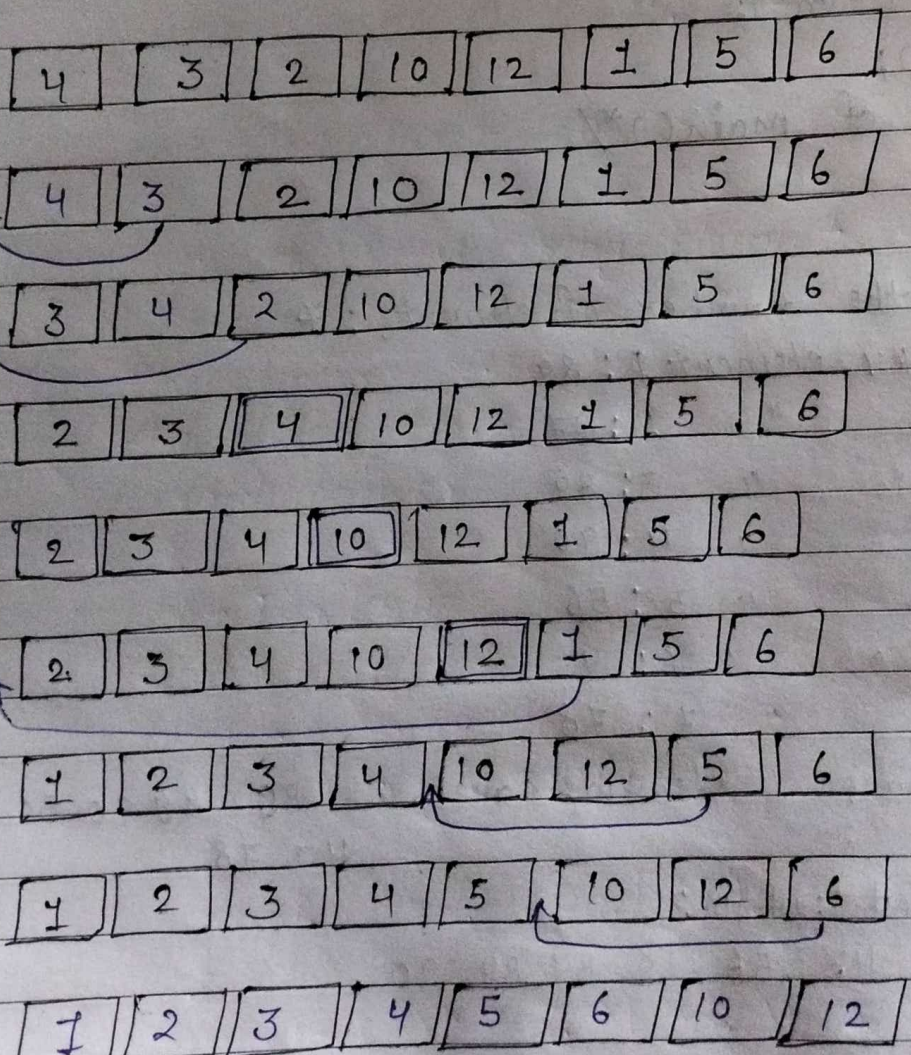
```
Enter the number of elements : 7
   enter element 1 : 89
   "      "    2 : 87
   "      "    3 : 89
   "      "    4 : 90
   "      "    5 : 56
      "      "  6 : 45
   "      "    7 : 78
After pass 1 elements are : 87 89 89 90 56
                            45 78

Sorted list is :
87 43 56 78 87 89 90
```

Insertion sort (left side se check)

- Insertion sort keeps making the left side of the array sorted until the whole array is sorted. It sorts the values seen far away and repeatedly inserts unseen values in the array into the left sorted array.
- It is the simplest of all sorting algorithms.
- Although it has the same complexity as Bubble sort, the intention sort is a little over twice as efficient as the bubble sort.

Insertion sort execution example :-



Insertion.c :-

```
#include<stdio.h>
#define max 20
  void main()
  {
    int arr[max], i, j, k, n;
    printf("Enter the number of elements:");
    scanf("%d", &n);
   For (i=0; i<n; i++)
    {
      printf ("Enter element %d:", i+1);
      scanf ("%d", &arr[i]);
    }
    printf ("unsorted list is : \n");
    For (i=0; i<n; i++)
    printf ("%d", arr[i]);
    printf ("\n");
    /* insertion sort */
   For (j=1; j<n; j++)
   {
    K= arr[j]; /*K is to be inserted at proper place*/
    For (i=j-1; i>=0 && K<arr[i]; i--)
          arr[i+1] = arr[i];
     arr[i+1]=K;
     printf ("pass %d, element inserted in proper place:
                  %d\n", j, K);
     For (i=j-1; i>=0 && k<arr[i]; i
     For (i=0; i<n; i++)
     printf ("%d", arr[i]);
```

```
    printf("\n");
}
printf("sorted list is: \n");
for(i=0; i<n; i++)
printf("%d", arr[i]);
printf("\n");
}/* end of main() */
```

Output :-

```
Enter the number of elements: 4
    Enter element 1: 56
     "      "     2: 64
     "      "     3: 49
     "      "     4: 32

sorted list :- 64  56  49  32
```

Que WAP to Access Array element randomly.

=>

```
#include<stdio.h>
#include<conio.h>
void main()
{
int i, a[5];
clrscr();
printf("Enter the element in Array \n");
for(i=0; i<5; i++)
{
scanf("%d", &a[i]);
}
printf("\t %d \t %d \t %d", a[3], a[2], a[0]);
```

```
getch();
}
```

Output: Enter the element:
53
46
65
52
93
⇒ 53 65 46

After changing :-

```
Int i, a[5];
clrscr();
Printf("Enter the element in Array \n");
For (i=0; i<5; i++)
{
scanf("%d", &a[i]);
}
For (i=0; i<5; i++)
{
printf("\n %d", a[i]);
}
Printf("\t \n\n %d \t %d \t %d", a[3],
                a[2], a[0]);

Output: getch();
}
```

Output :-

Enter the element in Array :-

56

78

65

42

43

=)

56

78

65

42

43

=) 56 65 78 ~

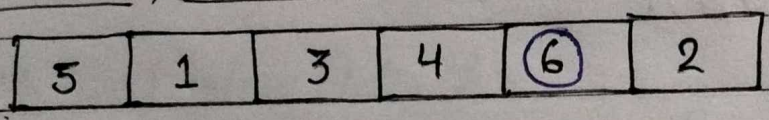
Enter the ele...
56
78
65
42
43

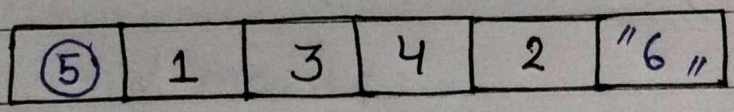
=>

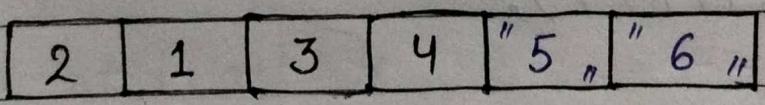
56
78
65
42
43

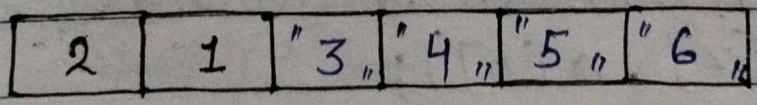
=> 56 65 78 /

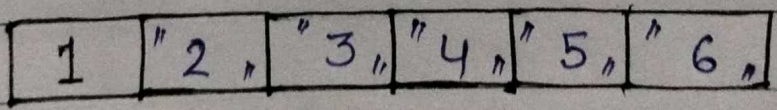
Selection sort :-

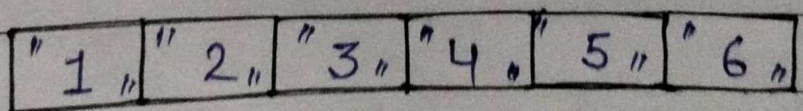












/* Program of sorting using selection sort*/

```
#include <stdio.h>
#define MAX 20
main()
{
    int arr[MAX], i, j, k, n, temp, smallest;
    printf("Enter the number of elements:");
    scanf("%d", &n);
    for(i=0; i<n; i++)
    {
        printf("Enter element %d:", i+1);
        scanf("%d", &arr[i]);
    }
    printf("Unsorting list is:\n");
    For(i=0; i<n; i++)
        printf("%d", arr[i]);
    printf("\n");
    /* selection sort */
    For(i=0; i<n-1; i++)
    {
        /* Find the smallest element */
        smallest = i;
        For(k= i+1; k<n; k++)
        {
            if(arr[smallest]>arr[k])
                smallest=k;
        }
        if(i!= smallest)
        {
            temp = arr[i];
```

```
        arr[i] = arr[smallest];
        arr[smallest] = temp;
    }
    printf("after pass %d elements are:", i+1);
    For(j=0; j<n; j++)
        printf("%d", arr[j]);
    Printf("\n");
    } /* end of for */
    printf("sorted list is:\n");
    For(i=0; i<n; i++)
        printf("%d", arr[i]);
    printf("\n");
}
```

Two dimensional Array :-

In one dimensional Array we can represent our data in either in Row or column but in two dimensional Array we can present our data in tables Form Row & Column both.

Syntax :-

```
<type-of-Array><Name-of-Array>
        [Row][cool];
```

Example :-

```
int arr[3][4];
```

By Graph :-

C ↓

R →

	C_0	C_1	C_2	C_3
R_0	R_0C_0	R_0C_1	R_0C_2	R_0C_3
R_1	R_1C_0	R_1C_1	R_1C_2	R_1C_3
R_2	R_2C_0	R_2C_1	R_2C_2	R_2C_3

C

R		C_0	C_1	C_2	C_3
→	R_0	R_0C_0	R_0C_1	R_0C_2	R_0C_3
	R_1	R_1C_0	R_1C_1	R_1C_2	R_1C_3
	R_2	R_2C_0	R_2C_1	R_2C_2	R_2C_3

Que WAP to show Matrice 3×3.

⇒
```
#include<stdio.h>
#include<conio.h>
void main()
{
int a[3][3],i,j;
clrscr();
printf("Enter the element in Array \n");
for(i=0;i<3;i++)
{
for(j=0;j<3;j++)
{
scanf("%d",&a[i][j]);
}
}
printf("\n The Matrix is:");
for(i=0;i<3;i++)
{
for(j=0;j<3;j++)
{
printf("\t%d",a[i][j]);
}
```

```
Printf ("\n");
}
getch();
}
```

Output :-

R →	j_0	j_1	j_2
i_0	11	12	13
i_1	14	15	16
i_2	17	18	19

Que WAP for addition of two Matrices and the Matrix size are 3x3.

=>
```
#include <stdio.h>
#include <conio.h>
void Main()
{
int a[3][3], b[3][3], c[3][3], i, j;
clrscr();
Printf ("Enter the Value in Ist Matrix");
For (i=0; i<3; i++)
{
for (j=0; j<3; j++)
{
&canf ("%d", &b [i][j]);
}
}
For (i=0; i<3; i++)
```

```
{
For (j=0; j<3; j++)
{
c[i][j] = a[i][j] + b[i][j];
}
}
Printf ("\n sum of two Matrix are \n");
For(i=0; i<3; i++)
{
For (j=0; j<3; j++)
{
Printf ("\t%d ", c[i][j]);
}
Printf ("\n");
}
getch();
}
```

Output:- Enter the value in 1st Matrix:

5
6
7
8
9
5
4
3
2

Enter the value in 2nd Matrix:

6
7

8
9
5
4
2
3
1

Sum of Two Matrix are :

11	13	15
17	14	9
6	6	3

Que WAP to show a subtraction of two Matrices size 的 is 3×3.

Ans.

```
#include<stdio.h>
#include<conio.h>
  void main()
  {
  int a[3][3], b[3][3] , c[3][3], i,j;
  clrscr();
  Printf ("Enter the Value in 1st Matrix ");
  For (i=0; i<3; i++)
  {
   for (j=0; j<3; j++)
   {
   &canf ("%d", &a[i][j]);
   }}
  Printf ("Enter the IInd Matrix ");
  For(i=0; i<3; i++)
  {
```

```
for (j=0; j<3; j++)
{
scanf ("%d", &b[i][j]);
}}
for (i=0; j<3; i++)
  {
  for (j=0; j<3; j++)
  {
  c[i][j] = a[i][j] - b[i][j];
  }}
  printf ("\t %d", c[i][j]);
  }
  printf ("\n");
  } getch(); }
```

Output

Enter the value in 1st Matrix:

9
8
7
7
8
9
8
7
9

Enter the value in IInd Matrix:

2 3
3 4
4
2
3
4
2

SUB of Two Matrix are:-

7 5 3
5 5 5
4 5 5

Que WAP For the Matrix Multiplication and the Matrix size is 3x3.

⇒
```
#include<stdio.h>
#include<conio.h>
  void main()
  {
   int a[3][3],b[3][3],c[3][3],i,j,k;
   clrscr();
   printf("Enter the value in Ist Matrix");
   For(i=0; i<3; i++)
   {
   For (j=0 ;j<3; j++)
    {
   scanf("%d", &a[i][j]);
   }}
   printf("Enter the value in IInd Matrix");
   For(i=0; i<3; i++)
   {
   For(j=0; j<3; j++)
   {
   scanf("%d", &b[i][j]);
   }
   }
   For(k=0; k<3; k++)
   {
   c[i][j]=0;
  For(i=0; i<3; i++)
   {
   For(j=0;j<3;j++)
   {
```

```
c[i][j] = c[i][j] + a[i][k] * b[k][j];
}
}
}
for (i=0 ; i<3 ; i++)
{
For (j=0; j<3 ; j++)
{
printf ("\t %d", c[i][j]);
}
printf ("\n");
}
```

Output :- Enter the value in Ist Matrix :- 2 3 4 5
2 7 3 4 2

Enter the value in 2nd Matrix :- 2 3 4 5
7 2 4 3 2

The Matrix Multiplication :-

25	39	22
48	50	38
34	43	24

Que Array with Function :-

Que WAP to pass Array element in Fn Arguement.

=) #include<stdio.h>
#include<conio.h>
void Display (int a1, int a2)
{
printf ("\n %d", a1);
printf ("\n %d", a2);
}
int main()
{
int age [] = {14, 16, 18}; clrscr();
display (age [1], age [2]);
return 0;
}

Output :- 16
18

Que WAP to pass Array as a arguement (Add fn with Array).

Ans. #include<stdio.h>
#include<conio.h>
float Calculation (float num []);
int main()
{
float result, num [] = {23.4, 55, 22.6, 3, 40, 5, 18};
result = Calculation (num);
printf ("\n Result = %.2f", result);
return 0;
}

```
float calculation (float num[])
{
float sum = 0.0;
int i;
for (i=0; i<6; i++)
{
sum = sum + num[i];
}
return sum;
}
```

Output :-

Result = 162.500000
Result = 162.500000

Que WAP to pass two dimensional Array as a arguement in a Function.

=)

```
#include <stdio.h>
#include <conio.h>
void display (int num[2][2]);
void main()
{
int num[2][2]; i;
clrscr();
printf ("Enter 4 element in Array \n");
for(i = 0; i<2; i++)
{
for (j=0; j<2; j++)
{
```

```
scanf("%d", &num[i][j]);
}
}
display(num);
return 0;
}
void display(int num[2][2])
{
printf("\n The Matrix \n");
For(i=0; i<2; i++)
{
For(j=0; j<2; j++)
{
printf("\t %d", num[i][j]);
}
printf("\n");
}
}
```

Output:-

Date
Page

Unit → IInd

Pointers

Pointer :- Pointer is variable that can hold the address of an other variable.

Que WAP to assign a value of one variable to another variable.

=>
```
#include<stdio.h>
#include<conio.h>
 void main()
 {
 int i,*j;
 clrscr();
 printf("Enter the value");
 scanf("%d",&i);
 j=&i;
 printf("The value of *j=%d",*j);
 printf("The Address of i=%u",&i);
 printf("The Address of *j=%u",&j);
 getch();
 }
```

Output :-

In pointer we're using a two symbols (*) :-

* → It's val use for accessing a value of variable.

& → In pointer this symbol is used for references of (adress) of variable.

%u → This specifier is used For Accessing variable Address. (Unsign)

Que WAP to understand the postfix, prefix, increment & Decrement in pointer's variable.

⇒
```
#include<stdio.h>
#include<conio.h>
void main()
{
int i, *j;
i = 5;
j = &i;
printf("The value of *j = %d", *j);
printf("The Address of *j = %u", &j);
printf("The value of *j = %d", ++j);
printf("The Address of ++j = %u", ++j);
printf("The Address of j++ = %u", j++);
printf("The value of --j = %d", --j);
printf("The Address of --j = %u", --j);
printf("The Address of j-- = %u", j--);
getch();
}
```

Date
Page

Output :-

Output :- The value of *j = 5
The address of *j = 65522
The value of *j = -10
The address of ++j = 65528
The " " j++ = 65530
The value of --j = -8
The value of --j = 65526 , The address of j-- = 65526

Pointer Airthmatic :-

It is a different from ordinary airthmatic. All Airthmatic is perform related to the size of base type of pointer.

For example :- If we have integer pointer (pi), which contain the Address 1000.

pi = 1000

then, on implementing we get pi = 1002

This is because the size of integer data type in 2 bytes.

For example :-

```
Int a = 5, * pi = &a;
float b = 2.2, * pf = &b;
char c = 'M', * pc = &c;
```

The Address of var are a = 1000
b = 4000
c = 5000

Int - 2 byte
float = 4 byte
char = 1 byte

Then,

(1.) pi++ or ++pi
(prefix) (postfix)
=) 1002

(2.) pi - 3
=) 998

(3.) pf++ =) 4004
(4.) pf-- =) 4000
(5.) pc++ =) 5001
(6.) pc-- =) 5000

pi = &a = 1000
↓ ↓ ↳ real addr
pointer Address
Variable of a

Que WAP to show all the Airthmatic operations in a pointer.

=)
```
#include<stdio.h>
int main()
{
    int no1, no2;
    int *ptr1, *ptr2;
    int sum, sub, mult;
    float div;
printf ("Enter number 1: \n");
scanf ("%d", &no1);
printf ("Enter number 2: \n");
scanf ("%d", &no2);
ptr1 = &no1; // ptr1 stores address of no1
ptr2 = &no2; // ptr2 stores address of no2
sum = (*ptr1)+(*ptr2);
sub = (*ptr1)-(*ptr2);
Mult = (*ptr1)*(*ptr2);
```

```
div = (*ptr1) / (*ptr2);
printf("sum = %d \n", sum);
printf ("subtraction = %d \n", sub);
printf ("Multiplication = %d \n", mult);
printf ("Division = %f \n", div);
return 0;
}
```

Output:-

```
Enter number 1: 20
Enter number 2: 10
  Sum = 30
  Subtraction = 10
  Multiplication = 200
  Division = 2.0000
```

Pointer to An Array :- we can declare a pointer that can point to the whole Array.

Que WAP to show pointer to an Array.

=>
```
#include <stdio.h>
#include <conio.h>
  void main()
  {
(only one add) int *p;   (simple pointer variable initialize)
(whole index no.) int (*ptr)[10];  (pointer size [10])
  int arr[10];
  clrscr();
  p = arr;  (only 0 Index Number Accept)
```

```
ptr = arr1; (Accept 10 Index Number)
printf("p = %u, ptr = %u\n", p, ptr);
  p++;          2020 → 2022
ptr++;          2020+2, 2022+2 ---
printf("p = %u, ptr = %u\n", p, ptr);
getch();
}
```

Output :-

Array to pointer :- we can declare an array that certain pointer as it's elements. every element of this Array is a pointer variable that can hold addr. of any variable that can hold addr. of any variable of Appropriate type. The declaration of an Array of pointer is same as declaring array as "*" is replaced before the array name.

syntax

<Array_type> * <Array_name> [size];

Ex:- int * arr[10];

Note:- Pointer is variable that can hold whole variable.

=)
```
#include<stdio.h>
#include<conio.h>
 void main()
 {
 int *p[3];
int i, a=10, b=11, c=15;
 clrscr();
p[0] = &a;
p[1] = &b;
p[2] = &c;
for (i=0; i<3; i++)
 {
printf ("p[%d] = %u \t", i, p[i]);
 printf ("p[%d] = %d \t", i, *p[i]);
 }
 getch();
 }
```

Output :-

pointer with Function :-

Que WAP to provide an increment of value using call by value Fn.

=>
```
#include <stdio.h>
#include <conio.h>
void value(int a, int b);
void main()
{
int a=5, b=6;
clrscr();
printf("\n Before calling Function the values are
    a=%d, b=%d", a, b);
value(a,b);
printf("After calling Fn The value are a=%d, b=
    %d", a,b);
getch();
}
```

Output:-

```
void value(int a, int b
{
a++;
b++;
printf("\n In Fn change values
    are a=%d, b=%d", a,b);
}
```

Q. Concept of call by References using pointer.

=>
```
#include<stdio.h>
#include<conio.h>
void value(int *x, int *y);
void main()
{
 int a=5, b=6;
 clrscr();
 printf("\n Before calling function the values are a
        =%d, b=%d", a, b);
 value(&a, &b);
 printf("\n After calling function the value are a=%d,
        b=%d", a, b);
 getch();
}
```

Output :-

```
void value(int *a, int *b)
{
 *a++;
 *b++;
  printf("\n In function
    change value are *a
    =%d, *b=%d, *x, *y);
}
```

Dynamic Memory locations in C :-

It allow program to obtain more Memory space while running. In simple terms DML allows Manually Handle Memory space For your program. The Allocation and release of this Memory space can be done with the help of some built-in-Functions whose properties are Alloc.h and stdlibh Header files the Fn take Memory Me from Memory area called HEAP and release this Memory whenever not require so that it cannot be use again For some other purpose.

1. Malloc ()→Fn
2. Calloc ()
3. Realloc ()
4. Free ()

Malloc :- It allocates required science of bytes and returnes a pointer. Iet bytes of Allocation is a space.

Syntax :-

```
ptr = (data-type*) Malloc (byte_size);
```

Malloc returns a pointer to an array of Memory with the size of byte size.

Malloc Example :-

ptr = (int *) malloc (100 * size of (int));

Calloc :- It allocates space for Array elements, Initialize to Zero and then returns pointers to Memory.

Syntax :-

ptr = (data-type *) calloc (n, element_size);

Example :-

ptr = (float *) calloc (25, (float));

25 × 4 = 100

ReAlloc :- change the size of previous Allocated space.

Syntx :- ptr = (data-type *) realloc (ptr, newsize);

Example :- ptr = (float *) realloc (ptr, new size));

Free :- free()

Allocates the previous Allocates space.

Syntax :- Free (ptr);

Example of Malloc :-

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h> (For utility) (Hardware components)
 void main()
 {
 int *ptr;
 int i,n; //Local variables
 Printf("Enter the number of element \n");
 scanf("%d",&n);
 printf("Entered number of elements : %d \n",n);
 ptr=(int *)malloc(n*size of (int)); // convert point C variable
                                        in the Form of Array
 If (ptr = NULL)
 {
 Printf("Memory not Allocated.");
 Exit(0);
 }
 else
 {
 Printf("\n Memory allocated by Malloc");
 For(i=0; i<n; i++)
 {
 ptr[i]=i+1;
 }
 Printf("\n the elements of the Array are \n");
 For(i=0; i<n; i++)
 {
 Printf("\n %d", ptr[i]);
 }}
 getch();
 }
```

Example of Calloc :-

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
 void main()
 {
  int *ptr;
  int i, n;
  n=5;
  printf("\n Entered elements are %d \n", n);
  ptr=(int *) calloc (n, size of (int));
  if (ptr = Null)
  {
  printf ("Memory not Allocated");
  exit(0);
  }
  else -
  {
  printf ("\n Memory Allocated by Calloc");
  For (i=0 ; i<n; i++)
  {
   ptr [i] = i+1;
  }
  printf ("\n the elements of the Array are \n");
  For(i=0; i<n; i++)
  {
  printf ("\n %d", ptr [i]);
  }}
  getch();
  }
```

Date
Page

Unit → 3rd

STRING

STRING

In c programming environment string Data type not support in c programming but with the help of Array we can implement string. The end of the string is marked with the special character "The null" character ('\0' → Null) which is simple the character with the value zero.

In c we have a lot of library functions in string.h header file.

A Blank space is consider as a blank character.

Que. WAP to Found the length of string.

⇒ **strlen():-** strlen() is a predefined Function in string.h header file. It's provide a Facility to count a number of characters in a string.

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
int n;
char a[20];
printf("Enter the string");
scanf("%s", &a);
n=strlen(a);
printf("\n the numbers of char is %d", n);
getch();
}
```

Que WAP to copy one string to another string.

=)
```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
char a[20], b[20];
clrscr();
printf("Enter the string");
scanf("%s", &a);
strcpy(b,a);
printf("\n The number of char is %s", b);
getch();
}
```

Que WAP to copy one string to another string.

=)
```
#include<stdio.h>
#include<conio.h>
#include<string.h>
    void main()
    {
    char a[20], b[20];
    clrscr();
    printf("enter the string");
    scanf("%s", &a);
    strcpy(b,a);
    printf("\n The number of char is %s", b);
    getch();
    }
```

[String]

1. strlen()
2. strcpy()
3. strrev()
4. strncpy()
5. strcat()

3.] strrev(): The Func^{n} proride a facility to reverse the string.

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

```
{
char a[20], b[20];
clrscr();
printf("enter the string \n");
scanf("%s", &a);
printf("\n The new string is %s", );
strrev(a));
getch();
}
```

Ques WAP to reverse the string without using strrev.

⇒

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
char str[20], temp;
int i, l;
clrscr();
printf("Enter the string");
scanf("%s", &str);
l = strlen(str)-1;
for(i=0; i<strlen(str)/2; i++)
{
temp = str[i];
str[i] = str[l];
str[l--] = temp;
}
```

printf ("\n Reverse string = %s", str);
getch();
}

Output :

Que WAP to count the length of string without using strlen.

=>
```
#include<stdio.h>
#include<conio.h>
#include<string.h>
 void main()
 {
 char str[20];
 int i;
 clrscr();
 printf("enter the string");
 scanf("%s", &str);
 For (i=0; str[i]!='\0'; i++);
 printf("The length of string is = %d", i);
 getch();
 }
```

Output :

Que WAP to Concatenation two strings without using strcat.

=)

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
 void main()
 {
 char First[20], sec[20];
 int i,j;
 clrscr();
 printf("Enter the First string");
 scanf("%s", &sec);
 For(i=0; First[i]!='\0';i++);
 For(j=0; sec[j]!='\0'; j++);
 {
 First[i]= sec[j];
 i++;
 }
 First[i]='\0';
 printf("The Concatenation of string is= %s", First);
 getch();
 }
```

Output :-

Date
Page

Unit - 4th

Data structure

In C- programming structure is a collection of different data-types in a single tag.

Syntax:-
```
struct <struct_name>
{
Data-type var1;
Data_type var2;
Data-type var3;
}tag
}<tag>;
```

Que WAP to display the information of students using structure.

⇒
```
#include <stdio.h>
#include <conio.h>
struct student
{
 int Roll no;
char name [50];
 char addr [100];
};
void Main()
{
 clrscr();
printf ("Enter the Roll no.");
scanf ("%d", s.Rollno);
printf ("\n Enter the Address name of student");
gets (s.Name);
printf ("\n Enter the Address of student");
gets (s.addr);
```

```
printf("\n\n ********** \n");
printf("\n student's Roll no = %d ", s.rollno);
printf("\n student's Name = %s ", s.Name);
printf("\n student's Address = %s", s.address);
    getch();
    }
```

Output :-

Que WAP to show the data of two students :-

Ans
```
#include<stdio.h>
#include<conio.h>
struct student
{
 int Roll no;
 Char name[50];
 Char addr[100];
} s1, s2;
Void main() :
{
 clrscr();
printf("Enter the Roll no ");
scanf("%d", s1.Roll no);
printf("\n Enter the name of student ");
scanf("%s", s1.Name);
```

```
printf("\n Enter the Address of student");
scanf("%.s", S1.Address);
printf("\n Enter the Roll no. of student 2");
scanf("%.d", &S2.roll no);
printf("\n .Enter the name of student 2");
scanf("%.s", S2.Name);
printf("\n Enter the Address of student 2");
scanf("%.s", S2.Address);
printf("\n\n Name of First student = %.s", S1.Name);
printf("\n Roll no. of First student = %.d", S1.Roll no);
printf("\n Address of First student = %.s", S1.
                                    Address);
printf("\n\n");
printf("\n Name of second student = %.s", S2.Name);
printf("\n Roll no = %.d", S2.Roll no);
printf("\n Address = %.s", S2.Addr);
getch();
}
```

Output :-

Date
Page

Qu WAP to create a result of 5 students using structure and the number of subjects are 5.

⇒
```
#include<stdio.h>
#include<conio.h>
struct student
{
 int roll no;
 char name [20];
int float Marks [5];
 float total;
 }s [2];
void Main()
{ clrscr();
 int i,j;
printf("Enter the information of student\n\n");
for(i=0; i<2 ; i++)
 {
printf ("Enter the roll no. of student");
scanf ("%d", &s[i]. roll);
printf ("\n Enter the name of student");
 scanf ("%s", &s[i]. name);
 s[i] .tot = 0.0;
For (j=0 ; j<3 ; j++)
{
 printf ("\n Enter the students Marks\n");
 scanf ("%d", &s[i]. Marks [j]);
 s[i]. tot = s[i]. tot + s[i]. Marks [j];
  }
 }
For (i=0; i<2; i++)-
```

```
{
printf ("\n student Roll no = %d", s[i].roll);
printf ("\n Name of student = %s", s[i].Name);
printf ("\n Total Marks = %d", s[i].tot);
}
getch();
}
```

Output :-

Accessing Members of the structure :-

There are two ways to access structure Members:

- By . (Member or dot operator)
- By -> (structure pointer operator)

typedef in C :- This keyboard is used to redefine the name of an already existing variable

Syntax :- typedef <existing_name><alias_name>

For ex:- typedef unsigned int Unit;

Program:-

```
#include<stdio.h>
  int main()
  {
  typedef unsigned int Unit;
  Unit i,j;
  i=10;
  j=20;
  printf("value of i is : %d", i);
  printf("\n value of j is : %d", j);
  return 0;
  }
```

Enumeration (or enum) in C:-

Introduction:-

Enumeration is a user defined data type in a C Language. It is used to assign name to the integral constants which makes a program easy to read and maintain. The keyword "enum" is used to declare an enumeration.

Here is the syntax of enum in C language.

- enum enum_name { Const 1, Const 2, --- };

The enum keyword is also used to define the variables of enum type. There are two ways to define the variables of enum type as follows.

- enum week { sun, Mon, Tue, wed, Thu, fri, sat };
- enum week day;

Difference between structure and union in C:-

Structure	Union
Keyword:- Keyword struct is used to define a structure	The Keyword union as used to define a Union.
Size:- when a variable associated with a structure the compiler Allocates the Memory for each Member. The size of structure is greater than or equal to the sum of sizes of it's members.	when a variable is associated with a Union, the compiler alocates the Memory by consideration the size of largest Memory so, size of Union is equal to the size of largest Member.
Memory:- Each member within a structure is assigned unique storage area of location.	Memory allocated is shared by individual Members of Union.
Value Altering:- Altering the value of a Members will not alters other Member in the structure.	Altering the value of any of the member will After other members values.

Accessing Members	Individual Members can be accessed at a time.	only one member can be accessed at a time.
Initialize of Members	several members of a structure can initialize at once.	only the first member of a Union can be initialize.

similarities between structure and Union :-

(i.) Both are user-defined data-types used to store data of different types as a single unit.

(ii.) Their Members can be objects of any type, including other structure and Unions or arrays. A Member can also consist of a bit field.

(iii.) Both structure and Unions support only assignment and size of operators. The two structure or Unions in the assignment must have the same Members and Member types.

(iv.) A structure or a Union can be passed by value to functions and returned by value. By functions, the Argument must have the same type as the function parameter. A structure or Union is passed by value just like a scalar variables as a corresponding

Correspondancing parameter.

(5.) "Operator or selection operator, which has one of the highest precedence is used for accessing member variables inside both the user-defined data-types.

program :-

```
Union abc
{
int a;
char b;
}var;
int main()
{
var.a=66;
printf("\n a=%d", var.a);
printf("\n b=%d", var.b);
printf("\n %c", var.b);
}
```

Accessing members of union using pointers :-

We can Access the members of the union through pointers by using the (->) arrow operator.

```
#include<stdio.h>
union abc
{
int a;
char b;
```

```
}
int main ()
{
Union abc *ptr; // pointer variable declaration
Union abc var;
var.a = 90;
ptr = &var;
printf ("The value of a is : %.d", ptr->a);
return 0;
}
```

In the above code, we have created a pointer variable, i.e *ptr, that stores the Address of var variable. Now, ptr can access the variable 'a' by using the (->) operator. Hence, the output of the above code would be 90.

1st Unit (Last topic)

Row Major order / column Major order in C programming :-

Array always created in a single dimensional in Memory because Memory in linear and have single integer Address.

Row Major order

Array_name [M][n]
where m (row) and n (column)

(i) If Index start with '0'.

Address of Call in A[i][j] = Base Address + [i*n+j]* size of data type.

A[2][2] = 2020 + [2*4+2]*2
= 2020 + (10*2)
= 2040

(ii) If Index start with '1'
Address of Call in A[i][j] = Base Address + [(i-1)*n+(j-1)]*size

A[3][3] = 2020 + [(3-1)*4+(3-1)]*2
= 2020 + [2*4+2]*2
= 2020 + [10*2]
= 2040

Column Major order :-

(i) If column start with '0'.

Address A[i][j] = Base Address + [j*m+i]* size of data.

exp. A[2][2] = 2020 + [2*3*2]*2
= 2020 + 8*2
= 2020 + 16
= 2036

(11) If column start with '1'

Address A[i][j] = Base Add. + [j-1]*m + (i-1)]* size of data.

$$A[3][3] = 2020 + [(3-1)*3 + (3-1)]*2$$
$$= 2020 + [2*3+2]*2$$
$$= 2020 + 16$$
$$= 2036$$

Que calculate Row Major order of Matrix 4x4 and index is start with '0'. location of element is [3][2], Base addel. 2020.

$$\Rightarrow A[3][2] = 2020 + [3\times4+2]*2$$
$$= 2020 + [14]*2$$
$$= 2020 + 28$$
$$= 2048$$

Ended 9 June, 22

UNIT → V

Introduction C preprocessor
Bit-wise operators

PRE-PROCESSOR DIRECTIVES

Preprocessor :- preprocessor is a Program that process the source code Before it passes through the Compiler.

It operates under the control of preprocessor or directives. which begins with the Hash (#) symbol.

There are three types of preprocessor :-

1. Macro substitution
2. file inclusion
3. compiler Control

Macro substitution directives :-

It replace every Acquarence of identifiers by Pre-defined strings.

Syntax :-

```
#define identifier string
```

For example :-

```
#include<stdio.h>
#define wait clrscr()
void main()
{
 wait;
printf ("Hello student");
}
```

Program

#Define 1 :-

```
#include<stdio.h>
#include<conio.h>
#define NAME "Pragya Kumari"
#define AGE 18
  void main()
  {
  clrscr()
printf ("%S age is %d", Name, Age);
getch();
}
```

Output :-

PRAGYA KUMARI Of 18.

 # Define 2 :-

```
#include<stdio.h>
#include<conio.h>
#define MIN (a,b) ((a)<(b)?(a):(b))
  void main()
  {
```

```
clrscr();
printf("The minimum value between 50 and 20
    is %d \n", min(50,20));
getch();
}
```

Output :-

The MINIMUM Value between 50 and 20 is
⇒ 20

#Define3 :-

```
#include<stdio.h>
#define square(a) a*a
 int main(){
 int b,c;
printf("Enter b element :");
 scanf("%d",&b);
 c=square(b); // replaces c = b*b before execution
                                of program)
printf("%d",c)
return 0;
}
```

Output :-

Enter b element : 2

C : 4

2. File inclusion :-

External file containing function Macro, definitions can be included using

#include<directive>

Syntax :- #include<file_name>

or,

#include "file_name"

In file inclusion we create a one Header file name is "PRAGYAT.h"

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

Next Page :-

```
#include "PragyaT.h">
 void main()
 {
 int i,j;
 char a[50];
 clrscr();
printf("Enter the string ");
scanf("%s",&a);
i= strlen(a);
```

```
printf("The length of string is = %d", i);
getch();
}
```

3. Compiler Control directives :-

C preprocessor offer a feature known as Conditional Compilention which can be used to switch ON, OFF of a particular line or group of lines.

Syntax :-

```
#if
# else
# endif
```

Program :-

```
#include<stdio.h>
#define LINE 1
 void main()
 {
#if line
 Printf("This is line Number 1");
 #else
 Printf("This is line Number 2");
#endif
}
```

Bit-wise operator :-

& → AND bit-wise operator
| → OR bit-wise operator.
~ → NOT bit-wise operator
<< → LEFT shift operator
>> → RIGHT shift operator
^ → XOR

bit-wise AND operator :-

(I.) It takes 2 bites at a time and perform AND operator.
(II.) AND operator is a binary operator it takes two number and perform bit-wise AND operation.
(III.) Result of AND is (1) when, both bites are (1).

Truth table :-

No.	A	B	A & B
0	0	0	0
1	0	1	0
2	1	0	0
3	1	1	1

Ques 7 & 4 = ?

```
  7 → 0 1 1 1
& 4 → 0 1 0 0
-------------
  4 ← 0 1 0 0
-------------
```

Bit-wise OR operator :-

Truth Table

No.	A	B	A | B
0	0	0	0
1	0	1	1
2	1	0	1
3	1	1	1

Que 7 | 4 = ?

7 → 0 1 1 1
4 → 0 1 0 0
7 ← 0 1 1 1

Bit-wise NOT Operator :-

(i) NOT is a Unary Operator.
(ii) It's job is to compliment each bit one-by-one.
(iii) Result of NOT is '0' when bit is one and 'one' when bit is 'zero'.

Truth Table :-

No.	A	~A
0	0	1
1	1	0

Que 7~ = ?

7 → ~0 1 1 1
8 ← 1 0 0 0

C shift Operators :-

→ In C, we have two types of shift Operators.
(i) Right shift operator
(ii) Left shift operator

→ It needs two operands
→ It shift each bit from left to right or right to left.

⇒ Right shift Operator :-

In right shift Operator each bit shifts from left to right For example.

· int a = 10 >> 2
→ firstly, decimal numbers gets Converted into binary numbers.
→ Then as per the instructions, the number of bits shifts from left to right.

Que. Right shift two bits of 14 i.e. 14>>2

⇒ 14 → 1110
∴ 14>>2 = 1110>>2 = 0011

Que Three bit shifting :-

⇒ int a = 10 >>3

↓

1010 → 0001 0x → 1

(1.) 10/2 = 5
(2.) 5/2 = 2
(3.) 2/2 = 1

→ Left shift Operator :-

If shift each bit from Right to left.

Ex :- Int a = 10 << 2

↓

1010 → 1000 → 8

$\frac{10}{2} = 5$

→ 8 bit operator :-

Ans. int a = 10 << 2

↓

00001010 → 00101000 → 40

010x2 = 20
2042 = 40

Unit - 6

File Handling

file Handling in C :-

File Handling in C provide a Facility to create a file and perform Multiple Operations like, update, read and delete, create file from storage data/ Area.
we can perform these operations :-

- Creating of the New file
- Opening an existing file
- Reading from the file
- writing to the file
- Deleting the file

Functions for file Handling :-

No.	Function	Description
1.	Fopen()	Opens New or existing file
2.	fprintf()	write data into the file
3.	fscanf()	Read data from the file
4.	fputc()	writes a character into the file
5.	fgetc()	Reads a character from file
6.	fclose()	closes the file
7.	fseek()	sets the file pointer to given
8.	fputw()	writes an integer to file
9.	fgetw()	Reads an integer from file
10.	ftell()	Returns current position
11.	rewind()	sets the file pointer to the beginning of the file.

* opening file : fopen()

Note :- pointer isliye kyuki ye poora dynamic memory Allocation par work karta hai.

syntax :-

file *fopen (const char * file name, const char * mode);

For exp :-

fr = * open ("file_name.txt", "r");

fopen() function :-

Mode description :-

r → opens a text file in read mode.
w → opens a text file in write mode.
a → opens a text file in append mode.
r++ → opens a text file in read and write mode.
w+++ → opens a text file in write and read mode.
a+++ → Opens a text file in write, read and append mode
rb → opens a binary file in read mode.
wb → opens a binary file in write mode.
ab → opens a binary file in append mode.
rb+ → opens a binary file in read & write mode.
wb+ → opens a binary file in write & read mode.
ab+ → opens a binary file in read, write and append mode.

Note : append → Editing

Program :-

```
#include<stdio.h>
void main()
{
 FILE *fp;
 char ch;
 fp = fopen("file_Handle.c","r");
 while(1)
 {
  ch = fgetc(fp);
  if = (ch == EOF)
  break;
  printf("%c", ch);
 } }
 fcloseall(fp);
}
```

∴ EOF ⇒ End of FILE

WAP for file Handling :-

```
#include<stdio.h>
 main()
 {
 FILE *fp;
 char buff[255]; // creating char array to store
                    data of file
 clrscr();
 fp = fopen("file.txt","r");
 while (fscanf(fp, "%s", buff) != EOF)
 {
  Printf("\t %s", buff);
 }
 flalose(fp);
}
```

C fprintf() and fscanf()

• Program :-

WAP to write a content on text file :-

```
#include<stdio.h>
#include<conio.h>
 main()
 {
  FILE *fp(); //alrscr();
 fp = fopen("file.txt","w"); //opening file
 fprintf(fp,"Hello file by fprintf....\n");
                //writing data into file
 fprintf(fp, "Hello students....\n");
 fclose(fp); //closing file
 }
```

A file Handling example to store employee info. as entered by user from Console.

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

```
FILE * emp_file
FILE * emp_info;
int id;
int name;
float salary;
emp_info = fopen ("emp.txt", "wt");
if (emp_info == NULL)
{
   printf ("file does not exist \n");
   return;
}
printf ("Enter the id : ");
scanf ("%d", &id);
fprintf (emp_info, "ID = %d \n", id);
printf ("Enter the name \n");
scanf ("%s", name);
fprintf (emp_info, "name = %s \n", name);
printf ("Enter the salary : ");
scanf ("%f", &salary);
fprintf (emp_info", salary = %f ", salary);
fclose (emp_info);
}
```

Reading file: fgetc()

→ syntax:-

int fgetc (FILE * &pem)

```
#include <stdio.h>
#include <conio.h>
void main()
{
  FILE * fp;
  char c;
clrscr();
fp (fp = fopen ("Myfile.txt", "r");
while (
while ((c = fgetc (fp)) != EOF)
{
   printf ("%c", c);
}
fclose (fp);
getch();
}
```