Entity elements

[ ] * → degree.

int a[5][6] → two degree

int a[1][2] → 2 degree

printf("%d", *(a+0)); → o/p → 3

(*(a+0)) [ ] = same degree

- If degree → is less than declared → Print address
  Ex → int a[] ⇒ printf("%d", a+0) → o/p
  printf("%d", *(a+0)) → o/p

- If degree is more than declared → Error
  Ex → int a[] ⇒ printf("%d", **(a+0)) → o/p

- If degree is matched = to the declared ⇒ Print value
  Ex → int a[] ⇒ printf("%d", *(a+0))
  ⇒ printf("%d", a[0])

→ For 1 D array -

a[a] ⟶ *(a+a)

** Print address using %p in place of %d :-

→ printf("%p, %d, %d", i, a[i], a+i);

o/p ⇒ (0), (4), (700) → Address
index value

Dynam

```
int a[5] = { };
```
→ garbage value

```
int a[5] = {1, 2};
```
→ if even 1 value is written then other will be 0.

Dynamic array:-

```
int a[n];
```
→ Garbage value always

Static Array :-

```
int a[5] = {1, 2, 3}
for(i=0; i<5; i++)
```
Always

```
{
    printf("%d", a+i);   → Address printed (No degree needed)
    printf("%d", a[0]);  → Value printed (one degree)
    printf("%d", *(a+0)); → Value printed (degree needed)
```

Dynamic Array :-

```
int a[5] = {1};
printf("Enter the element in array");
for(int i=0; i<5; i++);
{
    scanf("%d", &a[i]);
}
for(int i=0; i<5; i++)
{
    printf("%d", a[i]);
}
```

2 ways to save memory -

① int n;
scanf("%d", &n);
int a[n];

② int a[1000];
scanf("%d", &n);
→ Now run the loop till n.

Programs -

Q. WAP to take input values in an array from user and then print their sum.

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

b[i] = a[i]

Q1: WAP to read and print element of array.
Q2: WAP to print all -ve element of array. If(a[i]<0)
Q3: WAP to find max & min. element in array.
Q4: WAP to total no. of even & odd (count).
Q5: WAP to copy elements from an array to another array.

(1)
```
int a[5];
for (int i=0; i<5; i++)
{
    scanf("%d", &a[i]);
}
for(int i=0; i<5; i++)
{
    printf("%d", a[i]);
}
```

(3)
```
int a[5], min, max=0;
scanf("%d", &min);
for(int i=0; i<5; i++)
{
    scanf("%d", &a[i]);
    if(a[i] < min)
    min = a[i];
    else if (a[i] > max)
    max = a[i];
}
printf("%d\n", min);
printf("%d", max);
```

(4)
```
int a[10], c=0, d=0;
for(int i=0; i<10; i++)
{
    scanf("%d", &a[i]);
    if(a[i]%2==0)
      c=c+1;
    else if(a[i]%2==1)
      d=d+1;
}
printf("No. count of even no. = %d", c);
printf("No. count of odd no = %d", d);
```

(5)
```
int n, m;
scanf("%d%d", &n, &m);
int a[n], b[m];
for(int i=0; i<n; i++)
{
    scanf("%d", &a[i]);
}
for(int j=0; j<m; j++)
{ scanf("%d", &b[j]);
for(
```

405

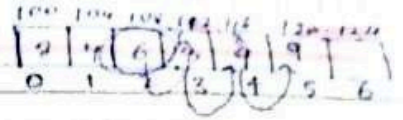
one by one

Linear search

Q: WAP to check whether a no. is present or not in a given array.

```
int main()
{ int a[5], i, f=0, k=0;
  printf("Enter an element to search");
  scanf("%d", &e);
  for(i=0; i<n; i++);
  {
    if (a[i] == e)
    { f=1;
      k= i+1;
      break;
    }
  }
  if (f==1)
  printf("Element is found at %d", k);
  else
  printf("Element is not found");
}
```

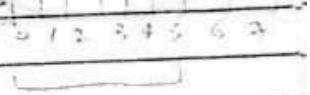
0	1	2	3	4
3	4	5	9	2



- Insertion in array :-

```
int n;
scanf("%d", &n);
int arr[n], m, ele, pos, i;
printf("Enter no of elements - ");
scanf("%d", &m);
for(int i = 0; i < m; i++);
{
    scanf("%d", &arr[i]);
}
printf("Enter the ele & pos = ");
scanf("%d%d", &ele, &pos);
for(i = m-1; i >= pos; i--)
{
    a[i+1] = a[i];
}
a[pos] = ele;
m = m+1;
for(i = 0; i < m; i++)
{
    printf("%d", a[i]);
}
```

m = no. of elements in array

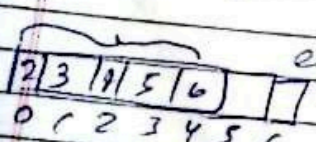


- Deletion in array :-

print

- Deletion in array:-

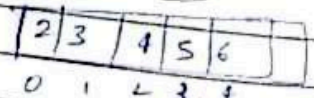
```
for
{
    scanf("%d", &arr[i]);
}
printf("Enter the pos = ");
scanf("%d", &pos);
for(i = m-1; i >= pos; i--)
{
    arr[i-1] = arr[i];
    m = m-1;
}
m = m-1;
for(i = 0; i < m; i++);
{
    printf("%d", arr[i]);
}
```



e=5.

b=e;

```
if(a[i] == e)
for(int i = m-1; i >= a[]; i--)
{
    a[i+1] = a[i];
}
m--;
```



a[i] = e.

a[2] = 4.

- **Bubble sort :-**

33	44	99	11	2	4	1
0	1	2	3	4	5	6

temp = a[1]

if (a[0] > a[1]) { a[1] = a[0];

Pass-1 :- ① 33 44 99 11 2 4 1

② 33 44 99 11 2 4 1

③ 33 44 11 99 2 4 1

④ 33 44 11 2 99 4 1

⑤ 33 44 11 2 4 99 1

⑥ 33 44 11 2 4 1 (99) → Stored and next pass will be move. work for 6 elements only.

Code -:

⑥ 5-1 = ⑤

```
for ( i=0 ; i<n - 1; i++)
{
    for(j = 0 ; j<n-1 -i ; j++)
    {      if(a[j] > a[j+1])
           { temp = a[j] ;
             a [j] = a[j+1];
             a [j+1] = temp.;
           }
    }.
```

Can be simply done as well but j one going

~~printf("%~~

```
for (i =0; i < n; i++) {
{   printf("%d", arr[i]);
```

Frequency:

```
// include <stdio.h>
# void main()
{
    int n, b[100000] = {0};
    scanf("%d", &n);
int a[n], i, max;
    for(i = 0; i<n; i++)
    {
        scanf("%d", &a[i]);
    }
    max = a[0];
    for(i = 0 ; i<n; i++)
    {
        if(a[i]>max)
            max = a[i];
    }

    max++;
// b[max] = {0};
    for(i=0; i<n; i++)
    {
        b[a[i]]++;
    }
    for(i=0; i<max; i++)
    {
        if(b[i]>0)
        printf("%d %d", i, b[i]);
    }
}
```

2-D array

static
Dynamic

Date: 12-1-22

Static: row, column

```
int a[3][2] = {{1,2},{3,4}};
```

	0	1
0	1	2
1	3	4
2		

```
int a[3][2] = {{1,2},{3,0},
               {4,5}};
```

optional

	0	1
0	1	2
1	3	0
2	4	5

Dynamic:

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

	0	1
@+0	a_{00}	a_{01}
@+1	a_{10}	a_{11}
@+2	a_{20}	a_{21}

```
int a[3][3];
```

	0	1	2
0	6	9	2
1	7	19	10
2	4	8	16
	124	128	132

&a = 100

a = 100

want to print a[0]

*(a+0).

a[1][0] = a+0 → Address.

⇒ *(a+0) → Address

⇒ (*(a+0)+0) → Address.

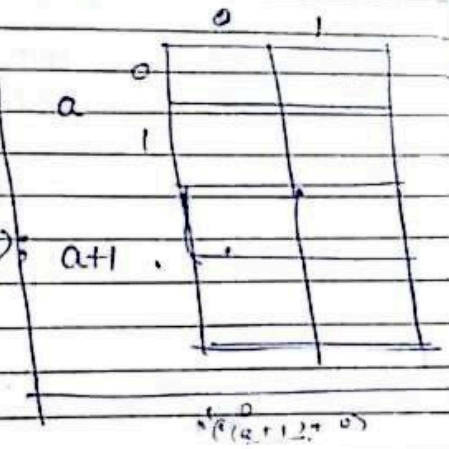
⇒ *(*(a+0)+0) → Value.

printf("%d", *(*(a+i)+j));
}
printf("\n");
}

3-D Array :-

row, column

int a[2][2][2];




```
int a[3][2] = {1,2,3,4,5,6};
for(i=0; i<2; i++)
{ for(j=0; j<2; j++)
  { printf("%d", a[j][i]);
  }
  printf("\n");
}
```

1 2
3 4
5 6

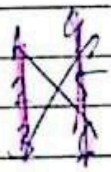
0 1 2

Output:- 1 3 5
2 4 6

Output:-

1 2 3
4 5 6

1 3 5
2 4 6

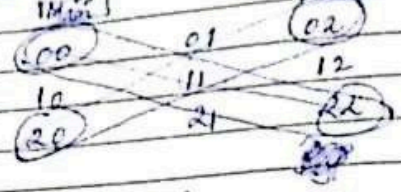


0 0 0 1
0

a[1][2]

C++ Lec-22

Q. Write a program to sum of diagonals of a matrix and interchange main and minor diagonals.



Code:

```
#include <stdio.h>
void main()
{ int n, sum=0, sum2=0;
  printf("Enter coefficient:", &n);
  int a[n][n];
  for (int i=0; i<n; i++)
  {
    for(int j=0; j<n; j++)
    {
      scanf("%d", &a[i][j]);
    }
  }
  for(int i=0; i<n; i++)
  {
    for(int j=0; j<n; j++)
    {
      if(i==j)
      { sum = sum + a[i][j]; }
      if(i+j==n-1)
      { sum2 = sum2 + a[i][j]; }
    }
  }
  printf("%d", sum);
  printf("%d", sum2);
}
```

[2-D array Programs]

Q. WAP to check a matrix is symmetric.
Q. " " " matrix is sparse symmetric.
Q. WAP to print upper triangle of a matrix (Major diagonal)
Q. WAP to print lower triangle of a matrix.

```
# include <stdio.h>
void main()
{
  int n, f=0;
  scanf("%d",&n);
  int a[n][n];
  for(int i=0; i<n; i++)
  {
    for( int j=0; j<n; j++)
    {
      scanf("%d", a[i][j]);
    }
  }
for(int i=0; i<n; i++)
  {
    for(int j=0; j<n; j++)
    {
      temp = i;
      i = j;
      j = temp;
    if(a[i][j]==a[j][i])
      f=1;
      break; break;
      break; goto lab;
    }
    lab:
    if(f==1)
    printf("Symm. matrix");
    else
    printf("Not a symm. matrix");
  }//main
```

$$\begin{Bmatrix} 1 & 2 & 3 \\ 2 & 1 & 0 \\ 3 & 0 & 1 \end{Bmatrix}$$

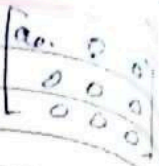
a[i][j]

$$\begin{matrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & a_{22} \end{matrix}$$

Q2-1

2) Spars matrix :-

```
#include<stdio.h>
void main()
{  int c=0;
   for
   {
      scanf(  );
   }
for(int i=0; i<m; i++)
  {
    for(int j=0; j<n; j++)
      {
        if(a[i][j]==0)
          c=c+1;
      }
  }
if(c==>(mxn)/2)
printf("Spars symmetric");
else
printf("Not a spars symmetric");
}
```

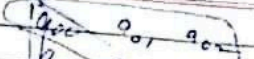


(3)

9 2x3

(3)/2

Upper Triangle



```
#include<stdio.h>
void main()
{
int
for(int i = 0; i<n; i++)
{
    for(int j=0; j<n; j++)
    {
        scanf("%d", &a[j][j]);
    }
}
for(int i=0; i<n; i++)
{
    for(int j=0; j<n; j++)
    {
        if(j>=i)
        printf("%d", a[i][j]);
    } else printf("--");
    printf("\n");
}
for(int i=0; i<n; i++)
{
    for(int j=0; j<n; j++)
    {
        if(i>=j)
        printf("%d", a[i][j]);
    } printf("--");
    printf("\n");
}
}
```

Upper Triangle

Lower Triangle