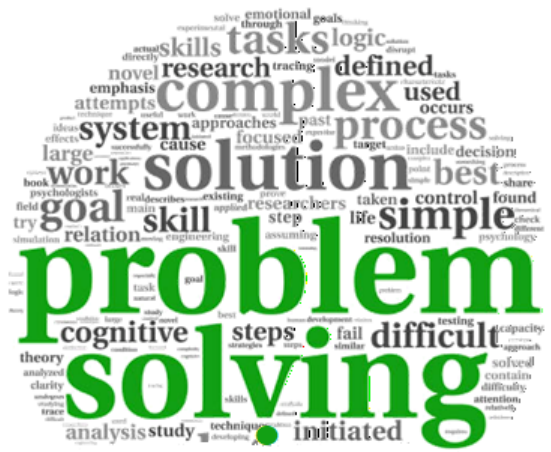


using computers

CSE 1051

2 D ARRAYS

S15-1

Objectives

To learn and appreciate the following concepts

- Programs using 2D arrays

Session outcome

At the end of session student will be able to

→ Write programs using 2D array

Syntax Recap

Declaration:

data-type array_name[row_size][column_size];

Initialization of two dimensional arrays:

type array-name [row size] [col size] ={list of values};

Reading a Matrix

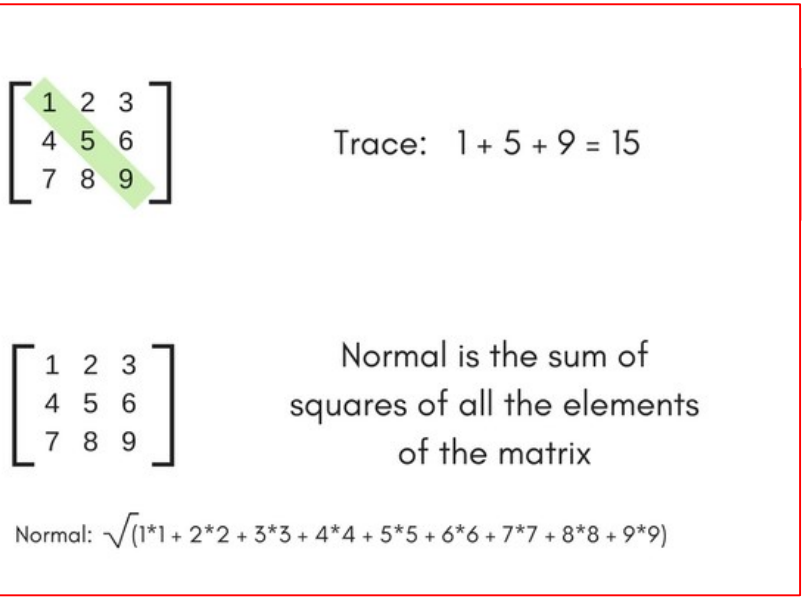
```
int a[100][100];
for(i=0;i<m;i++)
{
   for(j=0;j<n;j++)
      scanf("%d", &a[i][j]);
}
```

Display a Matrix

```
int a[100][100];
for(i=0;i<m;i++)
{
   for(j=0;j<n;j++)
      printf("%d\t",a[i][j]);
  printf("\n");
}
```

Trace and Norm of a Matrix

Trace is sum of principal diagonal elements of a square matrix.
Norm is Square Root of sum of squares of elements of a matrix.




```
int trace=0, sum=0,i,j,norm;
int m=3,n=3;
printf("enter elements for a \n");
for (i=0;i<m;i++){
 for(j=0;j<n;j++)
  scanf("%d",&a[i][j]);
}

for(i=0; i<m; i++)
   trace=trace + a[i][i];

for(i=0;i<m; i++)
{
for(j=0;j<n;j++)
  sum=sum+a[ i ][ j]*a[ i ][ j ];
}
norm=sqrt(sum);

printf(" trace is %d", trace );
printf(" norm is %d", norm );
```

```
enter the limits:
 3 3
enter elements:
 1 2 3 4 5 6 7 8 9
 trace is 15 norm is 16
```

Check whether a given Matrix is Symmetric or not

```
printf("enter dimension \n");
scanf("%d %d",&m, &n);
if(m!=n)
printf("it is not a square \n");
else
{
printf("enter elements \n");
for(i=0;i<m;i++)
   for(j=0;j<n;j++)
     scanf("%d", &a[i][j]);
for(i=0;i<m;i++){
   for(j=0;j<n;j++){
     if (a[ i ][ j ]!=a[ j ][ i ]
     {
       printf("\n matrix is not symmetric \n");
       exit(0);
     }
   }
}
printf("\n matrix is symmetric");
}
```

$$\begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 5 \\ 3 & 5 & 8 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 5 \\ 3 & 5 & 8 \end{bmatrix}^T$$

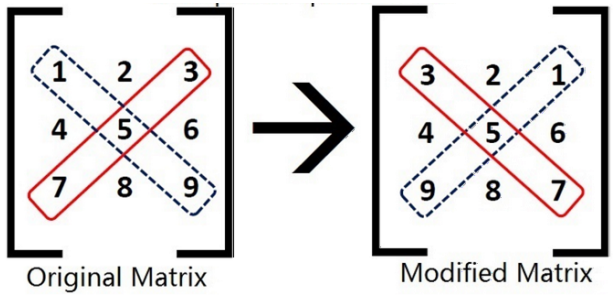
Symmetric matrix

Go to posts/chat box for the link to the question

submit your solution in next 2 minutes

The session will resume in 3 minutes

Exchange the elements of principal diagonal with secondary diagonal in an N dimensional Square matrix



Original Matrix Modified Matrix

```
int main() {
int i, j, temp, arr[4][4],n;

printf("\nEnter dimension: ");
scanf("%d",&n);

printf("\nEnter elements:\n");
for(i=0; i<n; i++)
for(j=0; j<n; j++)
scanf("%d", &arr[i][j]);
```

```
for(i=0;i<n;i++)
for(j=0;j<n;j++)
   if(i==j){
      temp=arr[i][j];
      arr[i][j]=arr[i][n-i-1];
      arr[i][n-i-1]=temp;
    }
printf("\nModified Matrix:\n");
for(i=0;i<n;i++){
  for(j=0;j<n;j++)
    printf("%d\t",arr[i][j]);
  printf("\n");
}
return 0;}
```

```
Enter dimension: 3

Enter elements:
 1 2 3 4 5 6 7 8 9

Modified Matrix:
3       2       1
4       5       6
9       8       7
```

Exchange the Rows and Columns of a ‘mxn’ matrix

```
/*read ‘mxn’ matrix */
printf("\nEnter the rows to exchange: “);
scanf(“%d %d”,&r1,&r2);

/*Row exchange r1 ⇔ r2 */
for(j=0;j<n;j++) {
   temp=arr[r1-1][j];
   arr[r1-1][j]=arr[r2-1][j];
   arr[r2-1][j]=temp;
 }
```

```
enter the limits:
 3 3
enter elements:
 1 2 3
 4 5 6
 7 8 9

Enter the rows to exchange: 1 3
Matrix after Rows exchanged:
7       8       9
4       5       6
1       2       3
```

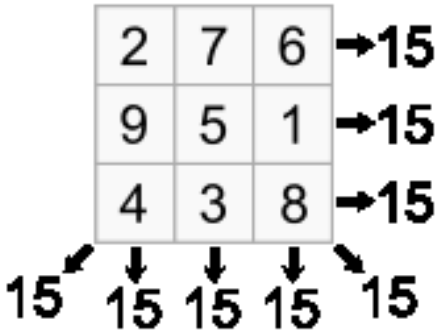
```
printf(“\nEnter the cols to exchange: “);
scanf(“%d %d”,&c1,&c2);

/*Column exchange : c1 ⇔ c2 */
for(i=0;i<m;i++) {
   temp=arr[i][c1-1];
   arr[i][c1-1]=arr[i][c2-1];
   arr[i][c2-1]=temp;
 }
```

```
Enter the cols to exchange: 1 3
Matrix after Columns exchanged:
9       8       7
6       5       4
3       2       1
```

Tutorials

- Write a program to check whether the given matrix is sparse matrix or not.
- Write a program to find the sum of the elements above and below diagonal elements in a matrix.
- Write program to check the given matrix is a magic square or not

(A magic square of order n is an arrangement of n^2 numbers, usually distinct integers, in a square, such that the n numbers in all rows, all columns, and both diagonals sum to the same constant. A normal magic square contains the integers from 1 to n^2.)

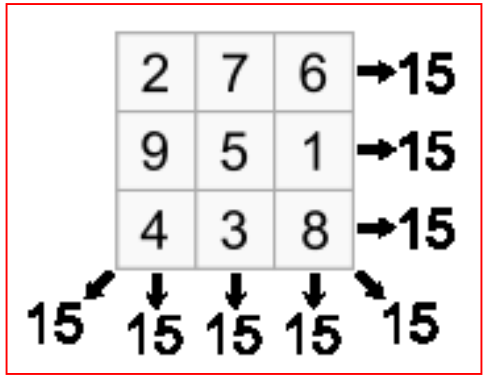


Extra problem: To be solved …

Write program to check the given matrix is a magic square or not

A **magic square** of order n is an arrangement of n^2 numbers, usually distinct integers, in a square, such that the n numbers in all rows, all columns, and both diagonals sum to the same constant.

A **normal** magic square contains the integers from 1 to n^2.



Magic Square

```
//Matrix is Magic square or not
int main()
{
int  mag[10][10], i, j, row, col, rowsum[10], colsum[10];
int pd=0, sd=0, k, x=0, b[100];
printf("enter dimension \n“);
scanf(“%d %d”,&row,&col);
if(row!=col)  // checking for square matrix
{
printf(“matrix is not square”);
exit(0);
}
```

8 1 6 3 5 7 4 9 2	6 1 8 7 5 3 2 9 4

Magic Square

```
//reading elements into the array
printf("\n enter elements for a \n“);
for(i=0;i<row;i++)
{
 for(j=0;j<col;j++)
    scanf(“%d”, &mag[i][j]);
}
```

```
//copying elements to 1D
for(i=0;i<row;i++)
        for(j=0;j<col;j++)
        b[x++]=mag[i][j];
```

```
//checking for uniqueness
for(k=0;k<x-1;k++)
for(j=k+1;j<x;j++)
        if(b[k]==b[j])
         {
printf(“elements are no distinct\n”);
printf(“matrix is not magic”);  exit(0);
         }
```

Magic Square

```
//Finding sum of elements on principal Diagonal
for(i=0; i<row; i++)
   pd=pd + mag[i][i];

//Row sum
for(i=0; i<row; i++)
{
          rowsum[i]=0;
          for(j=0;j< col; j++)
             rowsum[i]=rowsum[i]+mag[i][j];
//comparing rowsum and principal diagonal sum
          if(rowsum[i]!=pd)
          {
            printf("matrix is not magic");
            exit(0);
          }
}
```

Magic Square

```
//Finding column sum
for(i=0; i<col; i++)
{           colsum[i]=0;
            for(j=0; j<row; j++)
               colsum[i]=colsum[i]+mag[j][i];

//comparing column sum and principal diagonal sum
            if(colsum[i]!=pd){
              printf("matrix is not magic");
            exit(0);
            }
}
```

Magic Square

```
//finding secondary diagonal sum
i=row-1;
k=i;
for(j=col-1; j>=0; j--, i--)
sd=sd + mag[i][k-j];
if(sd!=pd) {
  printf("matrix is not magic");
  exit(0);
 }
printf("Matrix is magic\n");
}
```

```
enter dimension
3 3

 enter elements for a
8 1 6
3 5 7
4 9 2
Matrix is magic
```

```
enter dimension
3 3

 enter elements for a
1 2 3
4 5 6
7 8 9
matrix is not magic
```

Summary

- Declare, initialize and access 2D array
- Write programs using 2D array

Summary of 2D arrays

- Declare, initialize and access 2D array
- Write simple programs using 2D array
- Advance programming in 2D arrays