LECTURE NOTE 20

TWO DIMENSIONAL ARRAYS

Arrays that we have considered up to now are one dimensional array, a single line of elements. Often data come naturally in the form of a table, e.g. spreadsheet, which need a two-dimensional array.

Declaration:

The syntax is same as for 1-D array but here 2 subscripts are used.

Syntax:

data_type array_name[rowsize][columnsize];

Rowsize specifies the no.of rows Columnsize

specifies the no.of columns.

Example:

int a[4][5];

This is a 2-D array of 4 rows and 5 columns. Here the first element of the array is a[0][0] and last element of the array is a[3][4] and total no.of elements is 4*5=20.

	col 0	col 1	col 2	col 3	col 4
row 0	a[0][0]	a[0][1]	a[0][2]	a[0][3]	a[0][4]
row 1	a[1][0]	a[1][1]	a[1][2]	a[1][3]	a[1][4]
row 2	a[2][0]	a[2][1]	a[2][2]	a[2][3]	a[2][4]
row 3	a[3][0]	a[3][1]	a[3][2]	a[3][3]	a[3][4]

Initialization:

2-D arrays can be initialized in a way similar to 1-D arrays.

Example:

int m[4][3]={1,2,3,4,5,6,7,8,9,10,11,12};

The values are assigned as follows:

m[0][0]:1 m[0][1]:2 m[0][2]:3
m[1][0]:4 m[1][1]:5 m[3][2]:6

m[2][0]:7 m[2][1]:8 m[3][2]:9

m[3][0]:10 m[3][1]:11 m[3][2]:12

The initialization of group of elements as follows:

int m[4][3]={{11},{12,13},{14,15,16},{17}};

The values are assigned as:

m[0][0]:1 1 m[0][1]:0 m[0][2]:0

m[1][0]:12 m[1][1]:13 m[3][2]:0

m[2][0]:14 m[2][1]:15 m[3][2]:16

m[3][0]:17 m[3][1]:0 m[3][2]:0

Note:

In 2-D arrays it is optional to specify the first dimension but the second dimension should always be present.

Example: *int*

m[][3]={

{1,10},

{2,20,200},

{3},

{4,40,400} };

Here the first dimension is taken 4 since there are 4 roes in the initialization list. A 2-D array is known as matrix.

Processing:

For processing of 2-D arrays we need two nested for loops. The outer loop indicates the rows and the inner loop indicates the columns.

Example:

```
int a[4][5];
```

a) Reading values in a

```
for(i=0;i<4;i++)
    for(j=0;j<5;j++)
        scanf("%d",&a[i][j]);
```

b) Displaying values of a

```
for(i=0;i<4;i++)
    for(j=0;j<5;j++)
      printf("%d",a[i][j]);
```

Example 1:

Write a C program to find sum of two matrices

```
#include <stdio.h>
#include<conio.h>
void  main()
{
  float a[2][2], b[2][2], c[2][2];
   int i,j;
   clrscr();
  printf("Enter the elements of 1st matrix\n");
/* Reading two dimensional Array with the help of two for loop. If there is an array of 'n' dimension, 'n' numbers of loops are needed for inserting data to array.*/
 for(i=0;i<2;I++)
  for(j=0;j<2;j++)
  {
     scanf("%f",&a[i][j]);
   }
  printf("Enter the elements of 2nd matrix\n");
   for(i=0;i<2;i++)
   for(j=0;j<2;j++)
   {
```

```
      scanf("%f",&b[i][j]);
      }
/* accessing corresponding elements of two arrays. */
for(i=0;i<2;i++)
       for(j=0;j<2;j++)
          {
                c[i][j]=a[i][j]+b[i][j];  /* Sum of corresponding elements of two arrays. */
          }
 /* To display matrix sum in order. */
printf("\nSum Of Matrix:");
for(i=0;i<2;++i)
     {
       for(j=0;j<2;++j)
         printf("%f", c[i][j]);
      printf("\n");
     }
getch();
}
```

Example 2: Program for multiplication of two matrices

```
#include<stdio.h>
#include<conio.h>
int main()
  { int i,j,k;
     int row1,col1,row2,col2,row3,col3;
     int mat1[5][5], mat2[5][5], mat3[5][5];
     clrscr();
   printf("\n enter the number of rows in the first matrix:");
  scanf("%d", &row1);
   printf("\n enter the number of columns in the first matrix:");
  scanf("%d", &col1);
   printf("\n enter the number of rows in the second matrix:");
  scanf("%d", &row2);
  printf("\n enter the number of columns in the second matrix:");
  scanf("%d", &col2);
 if(col1 != row2)
   {
      printf("\n The number of columns in the first matrix must be equal to the number of rows
      in the second matrix ");
```

```
      getch();
      exit();
   }
row3= row1;
col3= col3;
printf("\n Enter the elements of the first matrix");
for(i=0;i<row1;i++)
  {
     for(j=0;j<col1;j++)
      scanf("%d",&mat1[i][j]);
  }

 printf("\n Enter the elements of the second matrix");
 for(i=0;i<row2;i++)
 {
     for(j=0;j<col2;j++)
     scanf("%d",&mat2[i][j]);
  }
for(i=0;i<row3;i++)
  {
    for(j=0;j<col3;j++)
       {
          mat3[i][j]=0;
          for(k=0;k<col3;k++)
               mat3[i][j] +=mat1[i][k]*mat2[k][j];
         }
      }
printf("\n The elements of the product matrix are"):
for(i=0;i<row3;i++)
  {
     printf("\n");
    for(j=0;j<col3;j++)
      printf("\t %d", mat3[i][j]);
  }
return 0;
}
```

Output:

Enter the number of rows in the first matrix: 2

```
Enter the number of columns in the first matrix: 2
Enter the number of rows in the second matrix: 2
Enter the number of columns in the second matrix: 2
Enter the elements of the first matrix
 1 2 3 4
Enter the elements of the second matrix
 5 6 7 8
The elements of the product matrix are
 19 22
 43 50
```

Example 3:

Program to find transpose of a matrix.

```
#include <stdio.h>
 int main()
 {
    int a[10][10], trans[10][10], r, c, i, j;
    printf("Enter rows and column of matrix: ");
    scanf("%d %d", &r, &c);
    printf("\nEnter elements of matrix:\n");
    for(i=0; i<r; i++)
     for(j=0; j<c; j++)
    {
       printf("Enter elements a%d%d: ",i+1,j+1);
       scanf("%d", &a[i][j]);
   }
/* Displaying the matrix a[][] */
printf("\n Entered Matrix: \n");
for(i=0; i<r; i++)
   for(j=0; j<c; j++)
   {
      printf("%d  ",a[i][j]);
      if(j==c-1)
          printf("\n\n");
   }

/* Finding transpose of matrix a[][] and storing it in array trans[][]. */
for(i=0; i<r;i++)
   for(j=0; j<c; j++)
   {
```

```
      trans[j][i]=a[i][j];
   }

/* Displaying the array trans[][]. */
printf("\nTranspose of Matrix:\n");
for(i=0; i<c;i++)
   for(j=0; j<r;j++)
   {
      printf("%d  ",trans[i][j]);
       if(j==r-1)
          printf("\n\n");
   }
return 0;
}
```

Output

```
Enter the rows and columns of matrix: 2   3
Enter the elements of matrix:
Enter elements a11: 1
Enter elements a12: 2
Enter elements a13: 9
Enter elements a21: 0
Enter elements a22: 4
Enter elements a23: 7
Entered matrix:
1 2 9
0 4 7
Transpose of matrix:
1 0
2 4
9 7
```

Multidimensional Array

More than 2-dimensional arrays are treated as multidimensional arrays.

Example:

```
int a[2][3][4];
```

Here a represents two 2-dimensional arrays and each of these 2-d arrays contains 3 rows and 4 columns.

The individual elements are:

a[0][0][0], a[0][0][1],a[0][0][2],a[0][1][0]……a[0][3][2]

a[1][0][0],a[1][0][1],a[1][0][2],a[1][1][0]……..a[1][3][2]

the total no. of elements in the above array is 2*3*4=24.

Initialization:

```
int a[2][4][3]={
{
 {1,2,3},
{4,5},
{6,7,8},
{9}
},
{
{10,11},
{12,13,14},
{15,16},
{17,18,19}
}
}
```

The values of elements after this initialization are as:

a[0][0][0]:1 a[0][0][1]:2 a[0][0][2]:3

a[0][1][0]:4 a[0][1][1]:5 a[0][1][2]:0

a[0][2][0]:6 a[0][2][1]:7 a[0][2][2]:8

a[0][3][0]:9 a[0][3][1]:0 a[0][3][2]:0

a[1][0][0]:10 a[1][0][1]:11 a[1][0][2]:0

a[1][1][0]:12 a[1][1][1]:13 a[1][1][2]:14

a[1][2][0]:15 a[1][2][1]:16 a[1][2][2]:0

a[1][3][0]:17 a[1][3][1]:18 a[1][3][2]:19

Note:

The rule of initialization of multidimensional arrays is that last subscript varies most frequently and the first subscript varies least rapidly.

Example:

```
#include<stdio.h>
 main()
{
 int d[5];
int i;
for(i=0;i<5;i++)
{
 d[i]=i;
}
for(i=0;i<5;i++)
{
printf("value in array %d\n",a[i]);
}
}
```

pictorial representation of d will look like

d[0]	d[1]	d[2]	d[3]	d[4]
0	1	2	3	4