O/P

Enter the elements of an array:

10
20
30
40
50
60
70
80
90
100.

The array elements are:

array[0] = 10
array[1] = 20
array[2] = 30
array[3] = 40
array[4] = 50
array[5] = 60
array[6] = 70
array[7] = 80
array[8] = 90
array[9] = 100.

## Two-dimensional array:-

The 2D array is organized as matrix which can be represented as the collection of rows and columns.

Syntax: data_type array_name [rows] [columns]

→ int two dimen [4][2];

Here 4 is the number of rows, and 2 is the no. of columns.

ex:

| | item 1 | item 2 | item 3 |
|---|---|---|---|
| Sales girl #1 | 310 | 275 | 365 |
| Sales girl #2 | 210 | 190 | 325 |
| Sales girl #3 | 405 | 235 | 240 |
| Sales girl #4 | 260 | 300 | 380 |

→ In mathematics, we represent a particular value in a matrix by using two subscripts such as Vij.

Here Vij → table label
i - is row
j - is column.

for ex: V23 = 325

→ C - allows us to define such tables of items by using two dimensional array.

V[4][3]

→ 2D arrays are declared as:

Type array_name [row_size][column_size];

→ C - places each size in its own set of brackets.

## 2D-array Stored in memory.

| | Column 0 | Column 1 | Column 2 |
|---|---|---|---|
| | [0][0] | [0][1] | [0][2] |
| Row 0 → | 310 | 275 | 365 |
| | [1][0] | [1][1] | [1][2] |
| Row 1 → | 210 | 190 | 325 |
| | [2][0] | [2][1] | [2][2] |
| Row 2 → | 405 | 235 | 240 |
| | [3][0] | [3][1] | [3][2] |
| Row 3 → | 310 | 275 | 365 |

## Initialization of 2D-array

1D-arrays & 2D-arrays may be initialized by following their declaration with a list of initial values enclosed in braces.

ex: `int table[2][3]={0,0,0,1,1,1};`

initialize the elements of the 1st row to zero and 2nd row to one.

→ The initialization is done row by row.

The statement is equivalently written as,

`int table[2][3]={{0,0,0},{1,1,1}};` by surrounding the elements of the each row by braces

→ When the array is completely initialized with all values explicitly we need not specify the size of the 1st dimension.

```
int table [3][3] = {
                    {0,0,0},
                    {1,1,1}
                    };
```

is permitted.

→ If the values are missing in an initializer, they are automatically set to zero.

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

→ When all the elements are to be initialized to zero, the following short cut may be used.

```
int m[3][5] = { {0}, {0}, {0} };
```

→ The 1st element of each rows is explicitly initialized to zero, while other elements are automatically initialized to zero.

```
int m[3][5] = {0,0};
```

eg:-

```
#include <stdio.h>
#include <conio.h>
#define maxgirls 4
#define maxitems 3

void main
{
```

```
int value [max girls][max items];
int girltotal [max girls], itemtotal [max items];
int i,j, grandtotal;
Printf ("\n Enter Value:");
for (i=0; i< max girls; i++)
{ girl total [i]=0;
for (j=0; j< max items; j++);
{ scanf ("%d", & value [i][j]);
girl total [i] = girl total [i] + value [i][j];
}}
Printf ("\n Computing item total :");
for (j=0; j< max items; j++)
{ item total [j]=0;
for (i=0; i< max girls; j++)
item total [j] = item total [j] + value [i][j];}
Printf ("\n Girls total ");
for (i=0; i< max girl; i++)
printf ("Sales girl [%d] = %d \n", i+1, girltotal [i]);
Printf ("\n item total ");
for (j=0; j< max items; j++)
Printf ("item [%d] = %d \n", j+1, itemtotal [j]);
Printf ("\n Grand total = %d \n", grandtotal);
getch();
}
```

O/p:

enter Values : 310 275 365

210 190 325

405 235 240

260 300 330

Item_total :

Girls totals Sales girl [1] = 950

Sales girl [2] = 705

[3] = 880

[4] - 940

item total item [1] = 1185

item [2] = 1000

[3] = 1310

Grand total : 3495