Two Dimensional Array

Two Dimensional Arrays

A two dimensional array is declared as follows

datatype arrname[nrows][ncols];

Column

Row

	0	1	2	3	4
0	88	65	4	12	88
1	93	3	26	26	72
2	65	58	26	0	37

int q[3][5];

Initialization

Two Dimensional Array

```
int q[3][5]  = { {88,65,4,12,88},
                 {93,3,26,26,72},
                 {65,58,26,0,37}
               };
```

ROW

COLUMN

Initialization

Two Dimensional Array

```
int q[][5]  = { {88,65,4,12,88},
                {93,3,26,26,72},
                {65,58,26,0,37}
              };
```

ROW

COLUMN

OPTIONAL

Initialization

Two Dimensional Array

```
int q[3][] = { {88,65,4,12,88},
               {93,3,26,26,72},
               {65,58,26,0,37}
             };
```

Can we skip this

ERROR !!!!!!!!!

Access Values

Two Dimensional Array

	0	1	2	3	4
0	88	65	4	12	88
1	93	3	26	26	72
2	65	58	26	0	37

q[0][1] yields the value 65

q[2][3] yields the value 0

q[0][0] yields the value 88

Storage

- **Column major storage**

	0	1	2	3	4
0	88	65	4	12	88
1	93	3	26	26	72
2	65	58	26	0	37

Storage

- **Column major storage**

	0	1	2	3	4
0	88	65	4	12	88
1	93	3	26	26	72
2	65	58	26	0	37

88,

Storage

- **Column major storage**

	0	1	2	3	4
0	88	65	4	12	88
1	93	3	26	26	72
2	65	58	26	0	37

88,93

Storage

- **Column Major Storage**

	0	1	2	3	4
0	88	65	4	12	88
1	93	3	26	26	72
2	65	58	26	0	37

88,93,65

Storage

- **Column Major Storage**

	0	1	2	3	4
0	88	65	4	12	88
1	93	3	26	26	72
2	65	58	26	0	37

88,93,65,65

Storage

- **Column Major Storage**

	0	1	2	3	4
0	88	65	4	12	88
1	93	3	26	26	72
2	65	58	26	0	37

88,93,65,65,3

Storage

- **Column Major Storage**

	0	1	2	3	4
0	88	65	4	12	88
1	93	3	26	26	72
2	65	58	26	0	37

88,93,65,65,3,58

Storage

- **Column Major Storage**

	0	1	2	3	4
0	88	65	4	12	88
1	93	3	26	26	72
2	65	58	26	0	37

88,93,65,65,3,58,4

Storage

- **Column Major Storage**

	0	1	2	3	4
0	88	65	4	12	88
1	93	3	26	26	72
2	65	58	26	0	37

88,93,65,65,3,58,4,26,26,12,26,0,88,72,37

Storage

- **Row Major Storage**

	0	1	2	3	4
0	88	65	4	12	88
1	93	3	26	26	72
2	65	58	26	0	37

Storage

- **Row Major Storage**

	0	1	2	3	4
0	88	65	4	12	88
1	93	3	26	26	72
2	65	58	26	0	37

88,

Storage

- **Row Major Storage**

	0	1	2	3	4
0	88	65	4	12	88
1	93	3	26	26	72
2	65	58	26	0	37

88,65

Storage

- **Row Major Storage**

	0	1	2	3	4
0	88	65	4	12	88
1	93	3	26	26	72
2	65	58	26	0	37

88,65,4

Storage

- **Row Major Storage**

	0	1	2	3	4
0	88	65	4	12	88
1	93	3	26	26	72
2	65	58	26	0	37

88,65,4,12

Storage

- **Row Major Storage**

	0	1	2	3	4
0	88	65	4	12	88
1	93	3	26	26	72
2	65	58	26	0	37

88,65,4,12,88

Storage

- **Row Major Storage**

	0	1	2	3	4
0	88	65	4	12	88
1	93	3	26	26	72
2	65	58	26	0	37

88,65,4,12,88,93

Storage

- **Row Major Storage**

	0	1	2	3	4
0	88	65	4	12	88
1	93	3	26	26	72
2	65	58	26	0	37

88,65,4,12,88,93,3

Storage

- **Row Major Storage**

	0	1	2	3	4
0	88	65	4	12	88
1	93	3	26	26	72
2	65	58	26	0	37

88,65,4,12,88,93,3,26,26,72,65,58,26,0,37

Storage

- Column Major Storage
- Row Major Storage

In the C language, all multidimensional arrays are stored in row major form, i.e., with the last index changing fastest as we go through the sequence of elements.

TWODARR.C

Program demonstrates use of pointers to access elements of a 2-dimensional array.

```
#include <stdio.h>
int main()
{
    int q[][5]= { {88,65,4,12,88},
                  {93,3,26,26,72},
                  {65,58,26,0,37}
                };
    int (*qptr)[5]; /* pointer to an array of 5 ints */
    int *qp;
    qptr = q;
    qp = &q[0][0];
```

NOT SAME AS

`int *qptr[5]`

TWODARR.C

Program demonstrates use of pointers to access elements of a 2-dimensional array.

```
   printf("\n q[0][0] = %d, %d,   %d",q[0][0],**qptr,*qp);
printf("\n q[1][0] = %d, %d,
   %d",q[1][0],*(*(qptr+1)),*(qp+5));
printf("\n q[0][1] = %d, %d, d",q[0][1],*(*qptr+1),*(qp+1));
    printf("\n q[2][4] = %d, %d,
   %d",q[2][4],*(*(q+2)+4),*(qp+14));

      return 0;

}
```

Multidimensional Arrays

- Multidimensional arrays are arrays with multiple indices.
- They are declared as follows.

```
datatype arrname[n1][n2][n3][n4];
```

Points to Remember

- Multidimensional arrays are internally stored as arrays of arrays. Therefore, they need special care when being passed to functions.
- The C language stores multidimensional arrays in row-major fashion, i.e., last index varying fastest when you traverse the elements in the one-dimensional memory.

THANK YOU