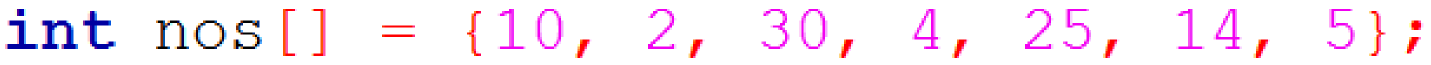
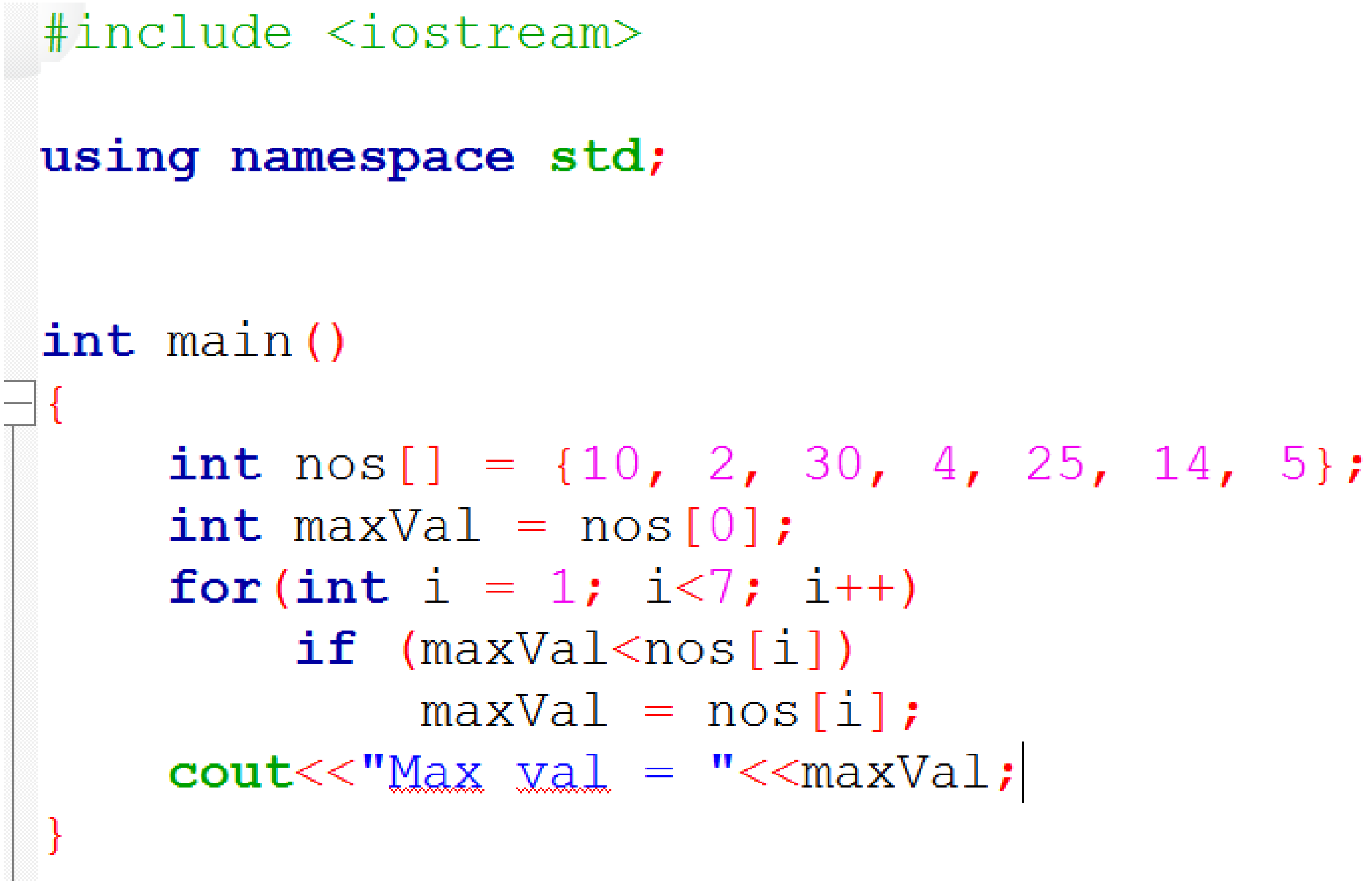
Arrays - Exercises

- Find the max value of a given array



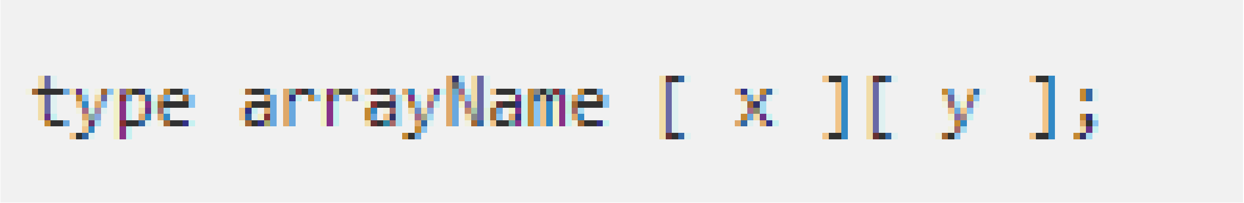
```
int nos[] = {10, 2, 30, 4, 25, 14, 5};
```



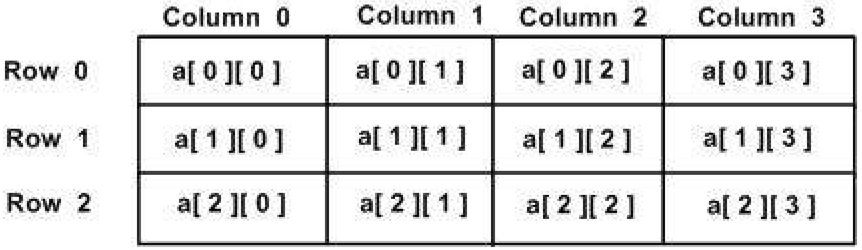
```
#include <iostream>

using namespace std;


int main()
{
    int nos[] = {10, 2, 30, 4, 25, 14, 5};
    int maxVal = nos[0];
    for(int i = 1; i<7; i++)
        if (maxVal<nos[i])
            maxVal = nos[i];
    cout<<"Max val = "<<maxVal;
}
```

Multi-dimensional arrays

Eg: Two dimensional array

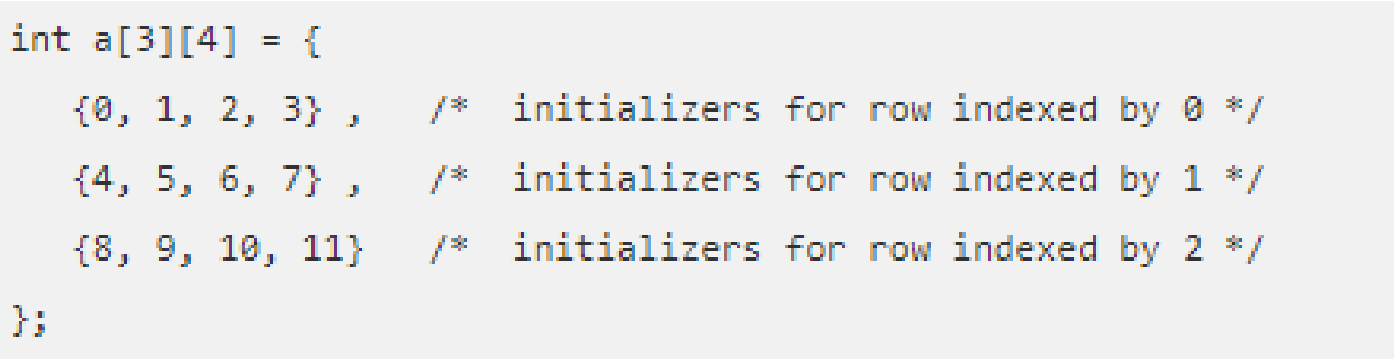


```
type arrayName [ x ][ y ];
```

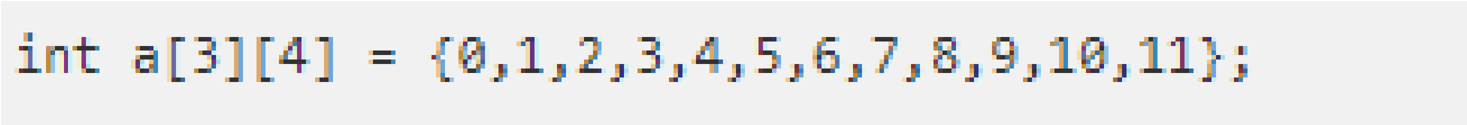


	Column 0	Column 1	Column 2	Column 3
Row 0	a[0][0]	a[0][1]	a[0][2]	a[0][3]
Row 1	a[1][0]	a[1][1]	a[1][2]	a[1][3]
Row 2	a[2][0]	a[2][1]	a[2][2]	a[2][3]

Initializing

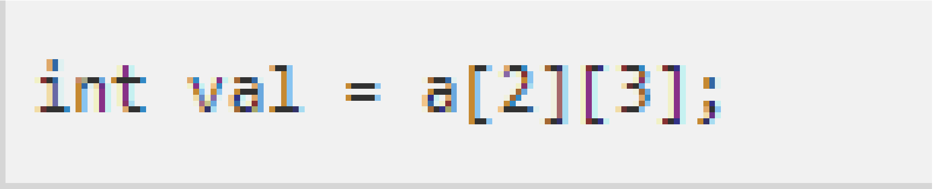


```
int a[3][4] = {
   {0, 1, 2, 3} ,   /*  initializers for row indexed by 0 */
   {4, 5, 6, 7} ,   /*  initializers for row indexed by 1 */
   {8, 9, 10, 11}   /*  initializers for row indexed by 2 */
};
```



```
int a[3][4] = {0,1,2,3,4,5,6,7,8,9,10,11};
```

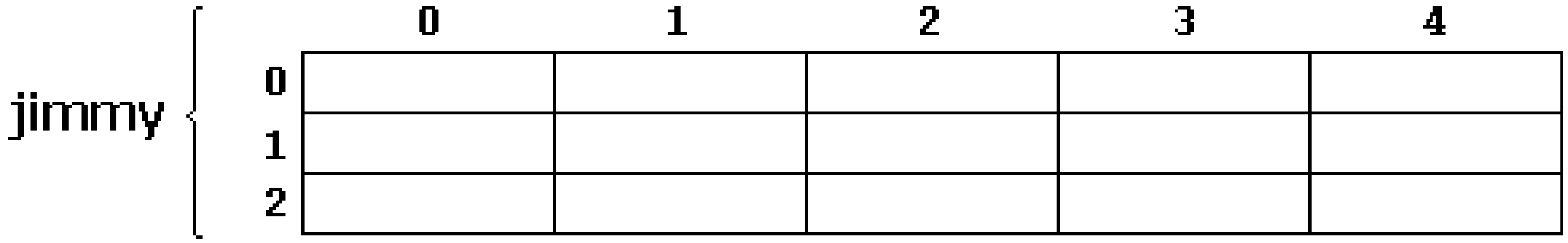
Accessing elements



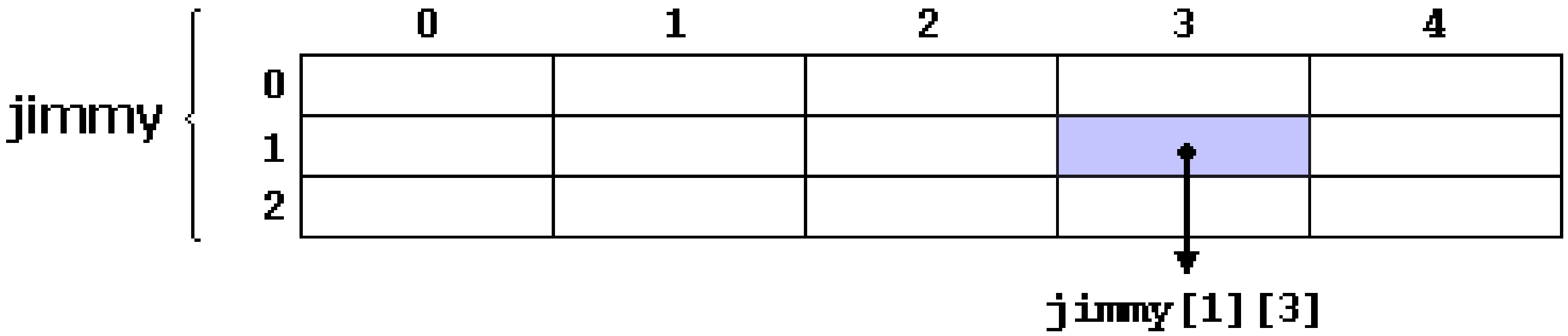
```
int val = a[2][3];
```

Multi-dimensional arrays

- Declaration `int jimmy [3][5];`

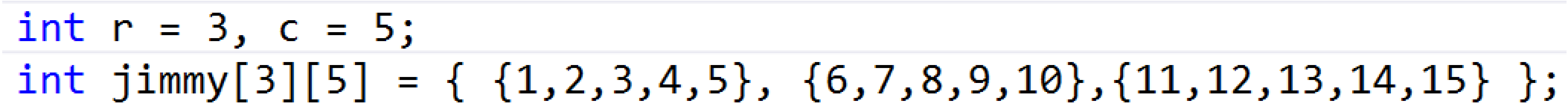


- Accessing elements `jimmy[1][3]`

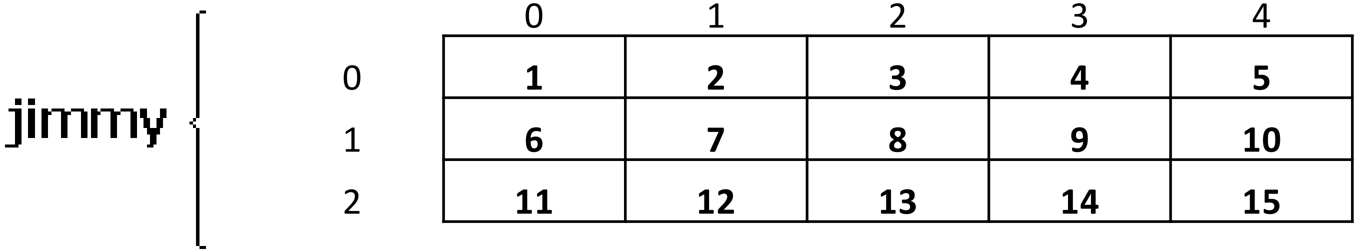


Array indices always start from 0

Multi-dimensional arrays - Exercise

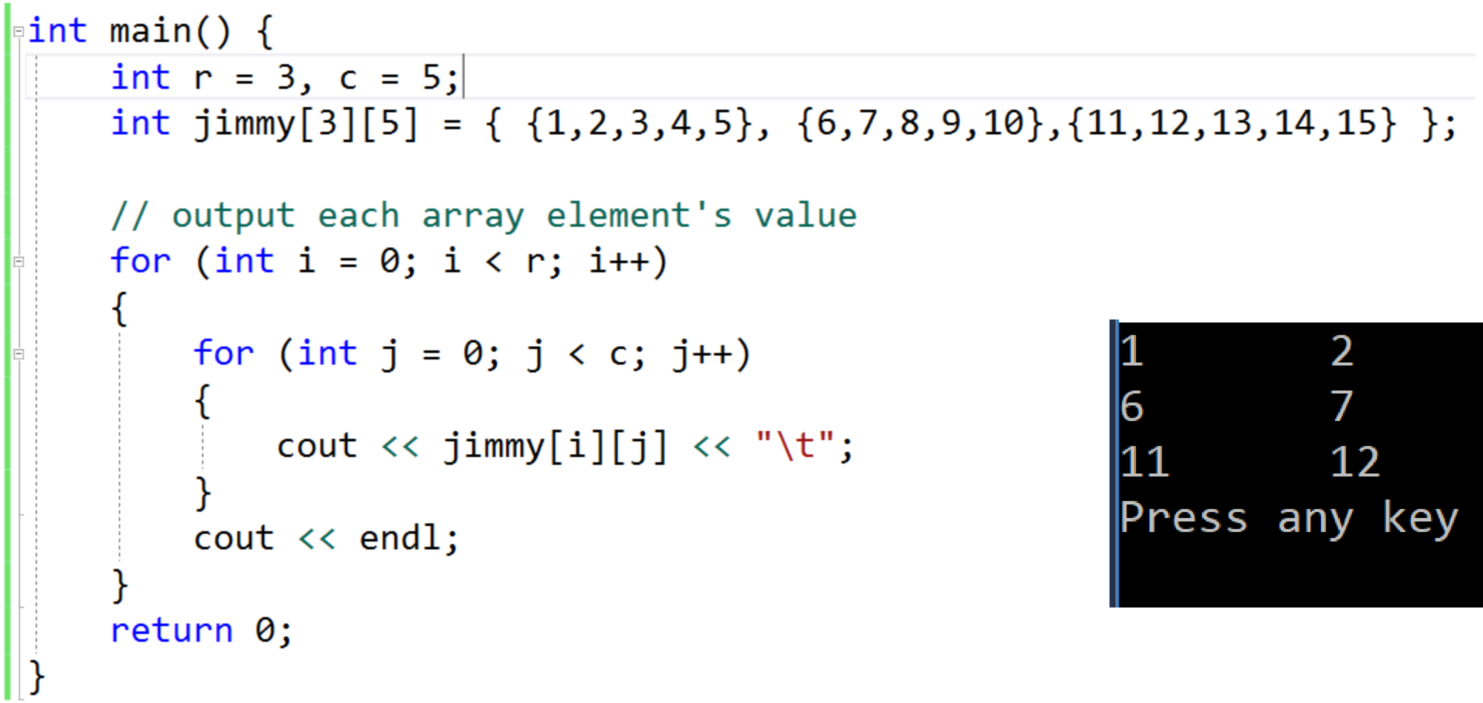


```
int r = 3, c = 5;
int jimmy[3][5] = { {1,2,3,4,5}, {6,7,8,9,10},{11,12,13,14,15} };
```

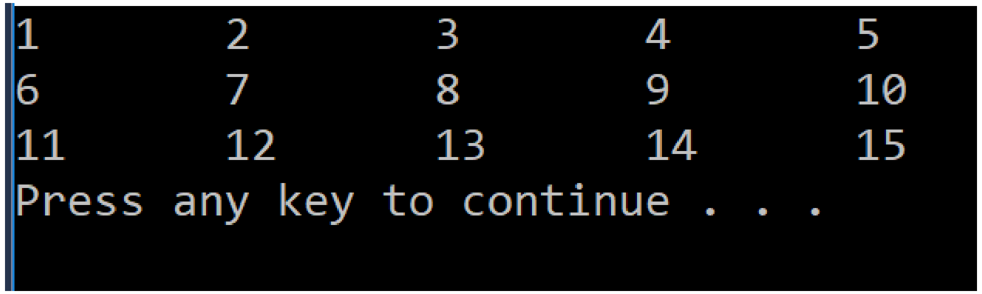


jimmy

	0	1	2	3	4
0	**1**	**2**	**3**	**4**	**5**
1	**6**	**7**	**8**	**9**	**10**
2	**11**	**12**	**13**	**14**	**15**



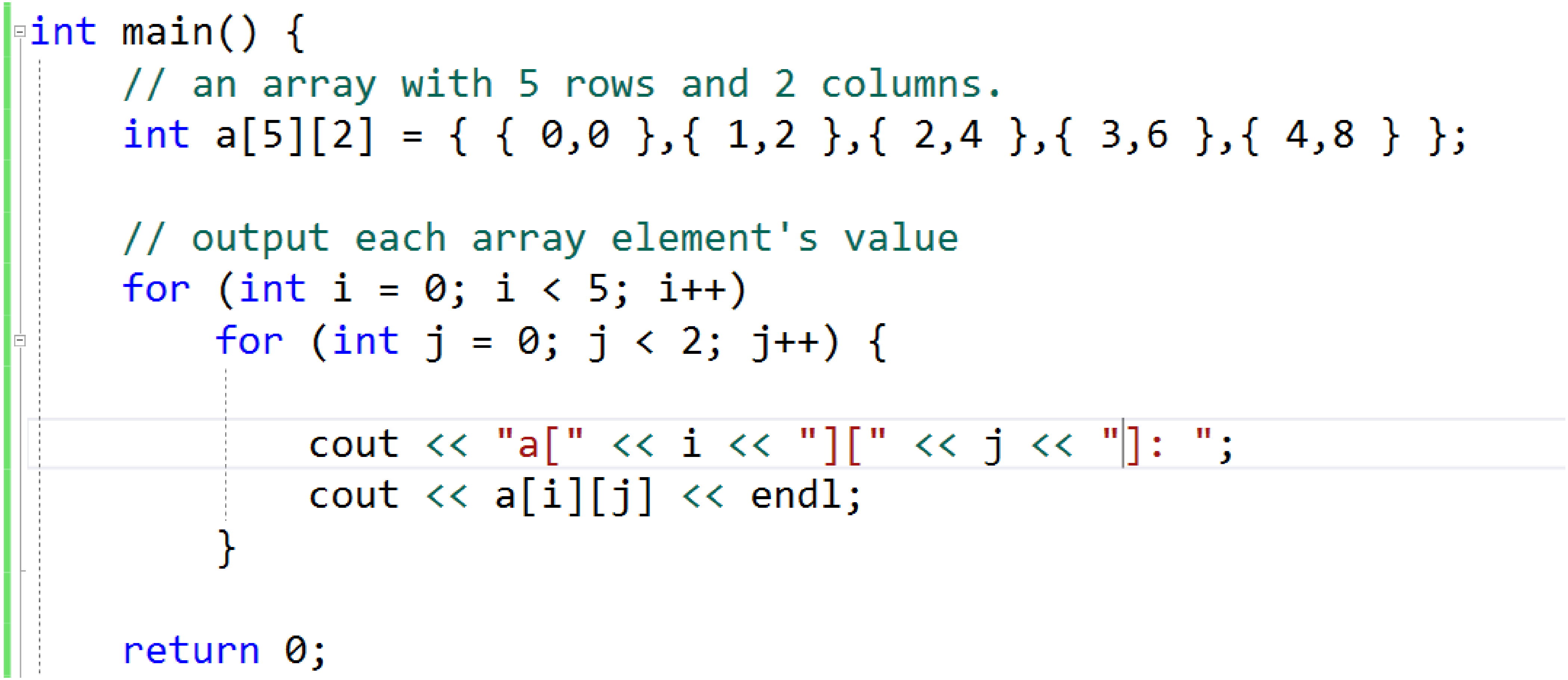
```
int main() {
    int r = 3, c = 5;
    int jimmy[3][5] = { {1,2,3,4,5}, {6,7,8,9,10},{11,12,13,14,15} };

    // output each array element's value
    for (int i = 0; i < r; i++)
    {
        for (int j = 0; j < c; j++)
        {
            cout << jimmy[i][j] << "\t";
        }
        cout << endl;
    }
    return 0;
}
```



```
1       2       3       4       5
6       7       8       9       10
11      12      13      14      15
Press any key to continue . . .
```

Multi-dimensional arrays – Exercise

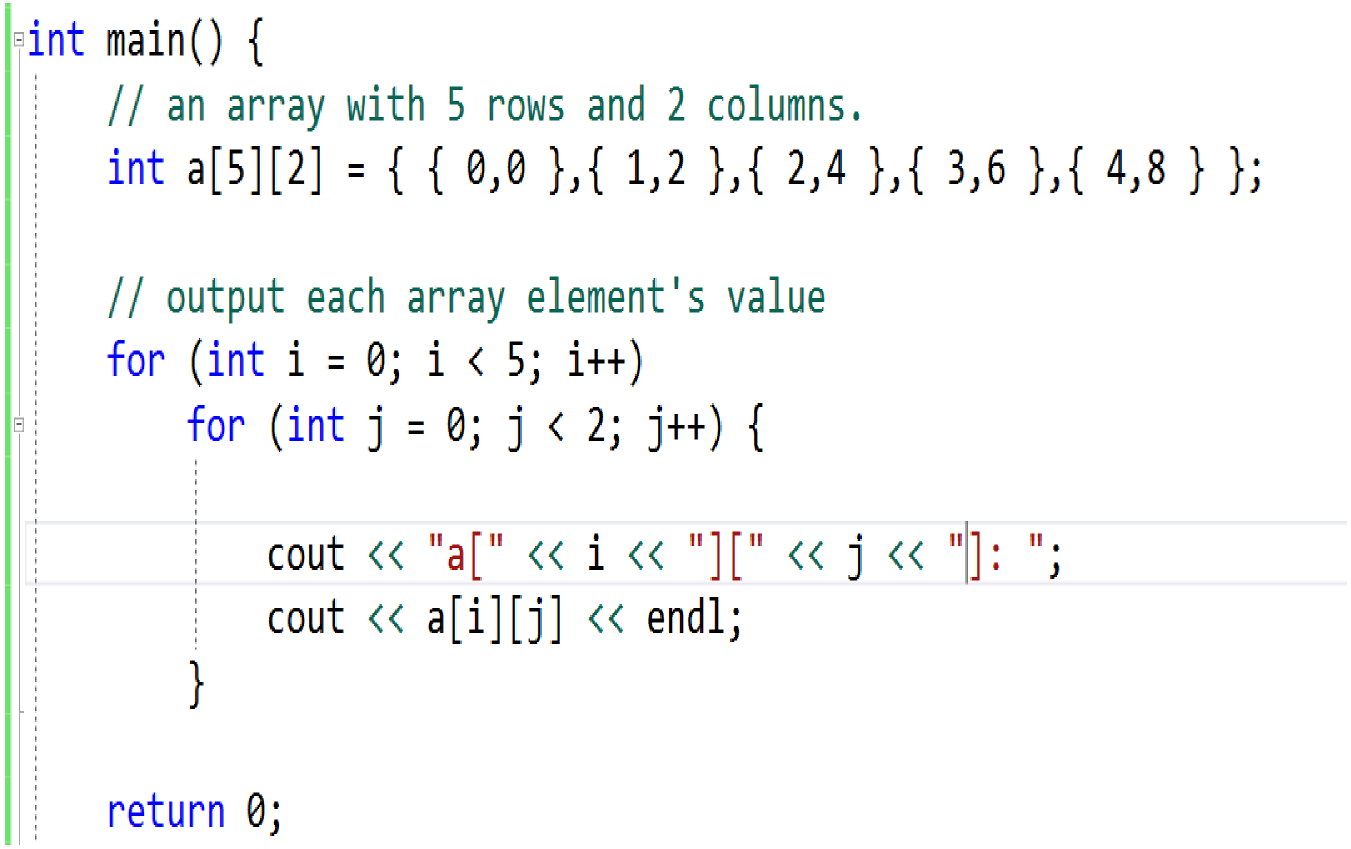
Write down the output of the following code



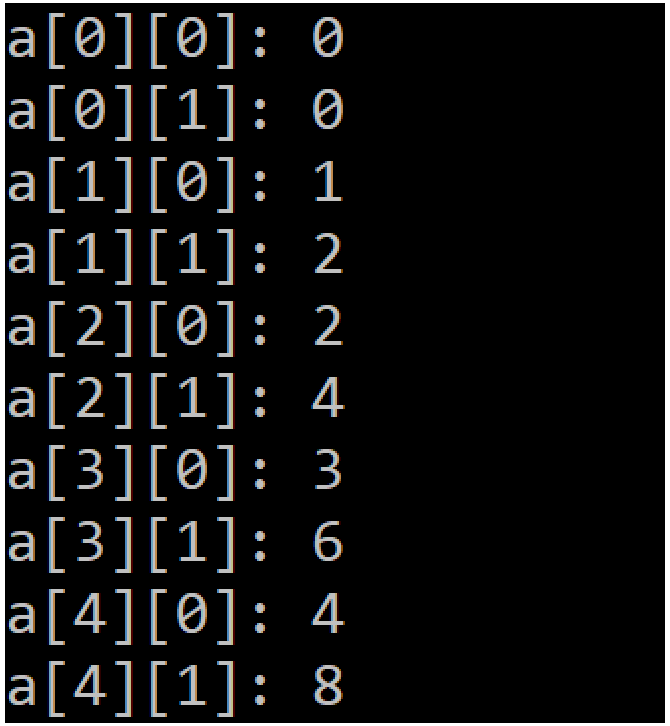
```
int main() {
    // an array with 5 rows and 2 columns.
    int a[5][2] = { { 0,0 },{ 1,2 },{ 2,4 },{ 3,6 },{ 4,8 } };

    // output each array element's value
    for (int i = 0; i < 5; i++)
        for (int j = 0; j < 2; j++) {

            cout << "a[" << i << "][" << j << "]: ";
            cout << a[i][j] << endl;
        }

    return 0;
```

Multi-dimensional arrays – Exercise

Write down the output of the following code



```
int main() {
    // an array with 5 rows and 2 columns.
    int a[5][2] = { { 0,0 },{ 1,2 },{ 2,4 },{ 3,6 },{ 4,8 } };

    // output each array element's value
    for (int i = 0; i < 5; i++)
        for (int j = 0; j < 2; j++) {

            cout << "a[" << i << "][" << j << "]: ";
            cout << a[i][j] << endl;
        }

    return 0;
```



```
a[0][0]: 0
a[0][1]: 0
a[1][0]: 1
a[1][1]: 2
a[2][0]: 2
a[2][1]: 4
a[3][0]: 3
a[3][1]: 6
a[4][0]: 4
a[4][1]: 8
```

int marks [3][10]={{80,65,42,10,100,198,76,56,15,100}, {75,-100,98,60,80,85,14,50,65,80},{125,-85,100,80,190,65,72,34,15,200}}

The given 2D array is having three students' 10 different course marks.

The following table shows the GPAs for the marks ranges.

Range	GPA
0-45	2.50
46-65	2.75
66-80	3.50
81-100	4.00

Task: Write C++ code to store the GPA of each course for each student in an array gpa. You need to do the correct validations for the marks. If there is any issue in any validation, ask the user for the new marks and replace the particular marks. Appropriate print statements should stop the marks replacement at one time.