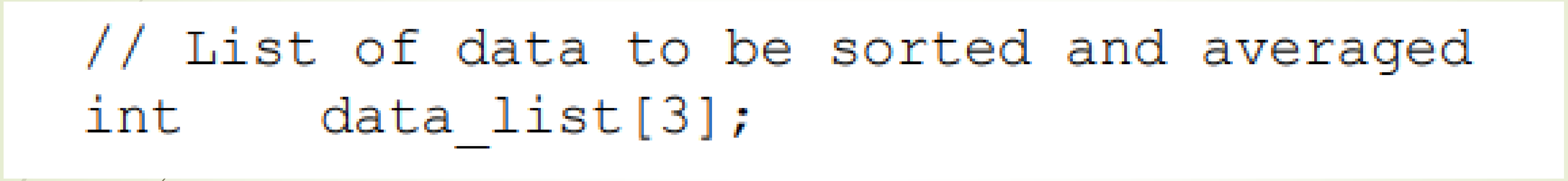
Lecture_6

Dr. Sara Mohamed

Array

Arrays

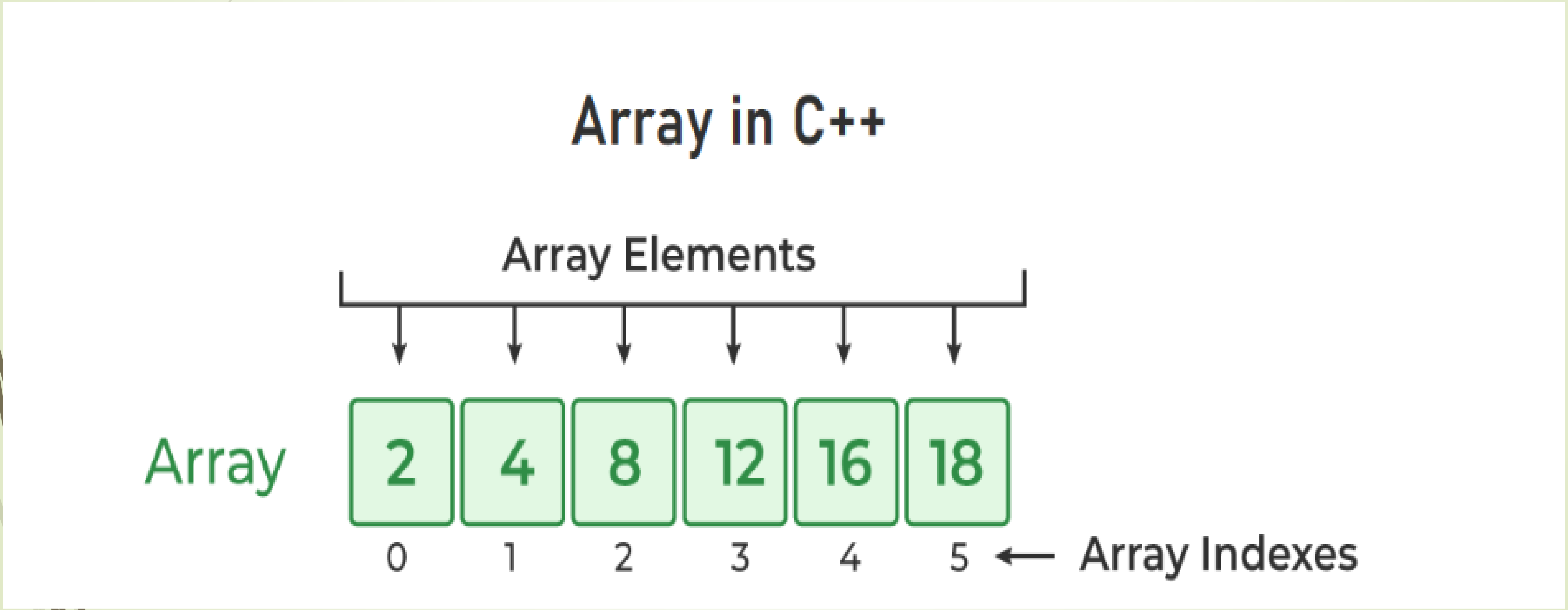
- An array is a set of consecutive memory locations used to store data. Each item in the array is called an element. The number of elements in an array is called the dimension of the array. A typical array declaration is:



```
// List of data to be sorted and averaged
int     data_list[3];
```

- This declares data_list to be an array of the three elements data_list[0], data_list [1], and data_list[2], which are separate variables.
- To reference an element of an array, you use a number called the **index** (the number inside the square brackets []).
- C++ is a funny language and likes to start counting at 0, so these three elements are numbered 0-2.

```
data_type array_name[size_of_array];
```



Properties of Arrays in C++

- An Array is a collection of data of the same data type, stored at a contiguous memory location.
- Indexing of an array starts from 0. It means the first element is stored at the 0th index, the second at 1st, and so on.
- Elements of an array can be accessed using their indices.
- Once an array is declared its size remains constant throughout the program.
- An array can have multiple dimensions.

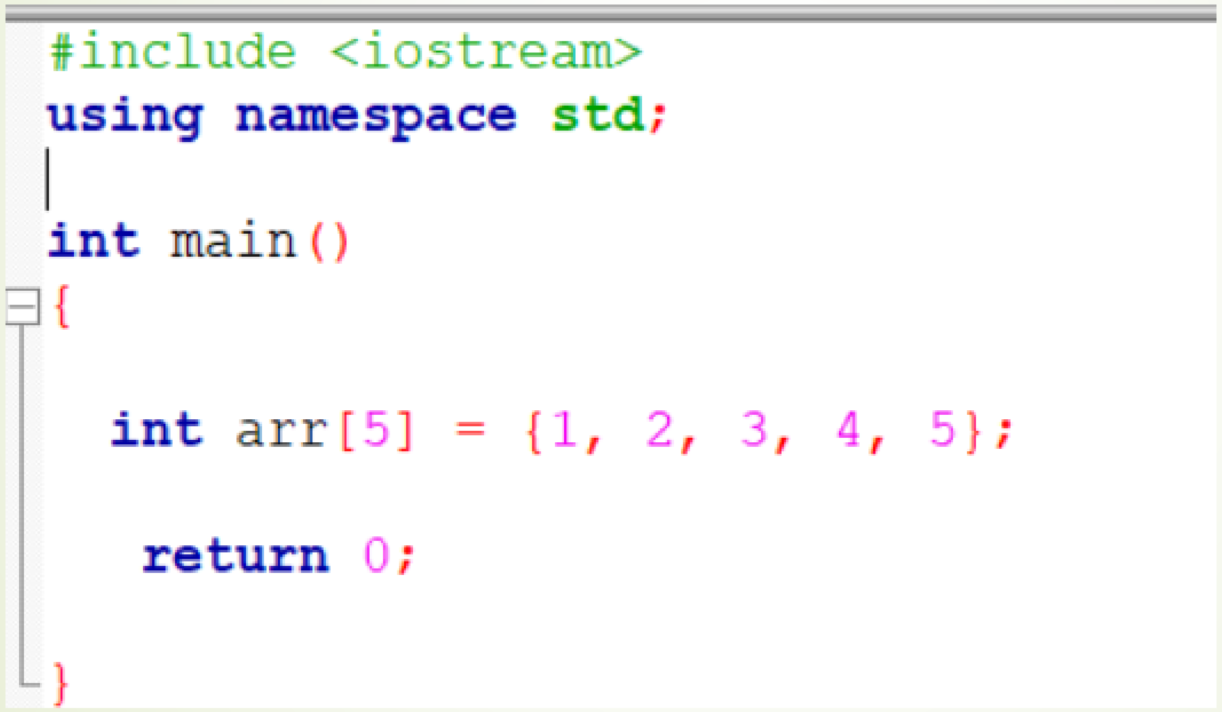
```
#include <iostream.h>

float data[5];      // data to average and total
float total;        // the total of the data items
float average;      // average of the items

main ()
{
    data[0] = 34.0;
    data[1] = 27.0;
    data[2] = 46.5;
    data[3] = 82.0;
    data[4] = 22.0;

    total = data[0] + data[1] + data[2] + data[3]  + data[4];
    average = total / 5.0;
    cout << "Total "<< total << " Average " << average << '\n';
    return (0);
}
```

This program outputs:

```
    Total 211.5 Average 42.3
```

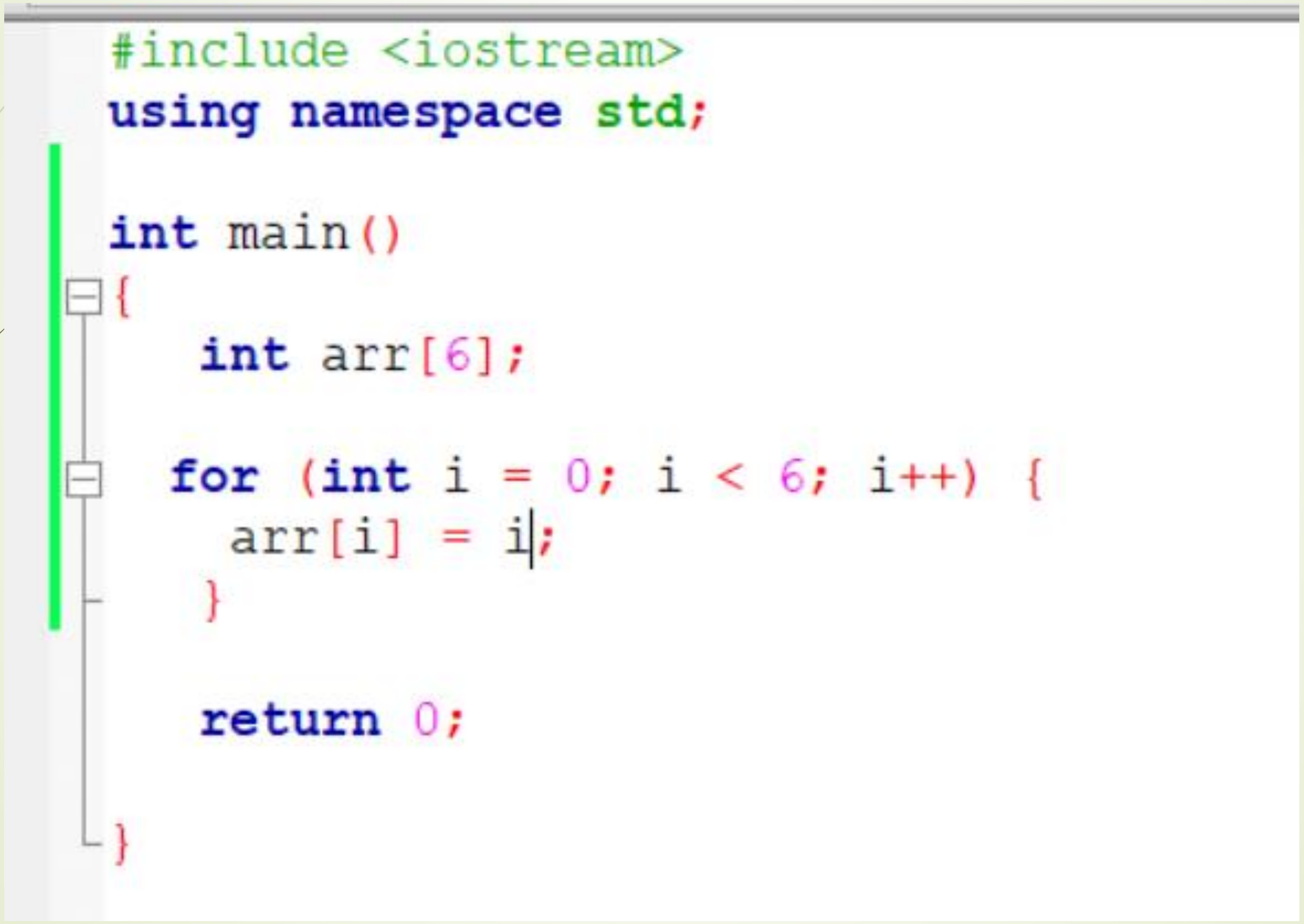
Initialization of Array in C++

- We have initialized the array with values. The values enclosed in curly braces ‘{}’ are assigned to the array:



```
#include <iostream>
using namespace std;

int main()
{

  int arr[5] = {1, 2, 3, 4, 5};

   return 0;

}
```

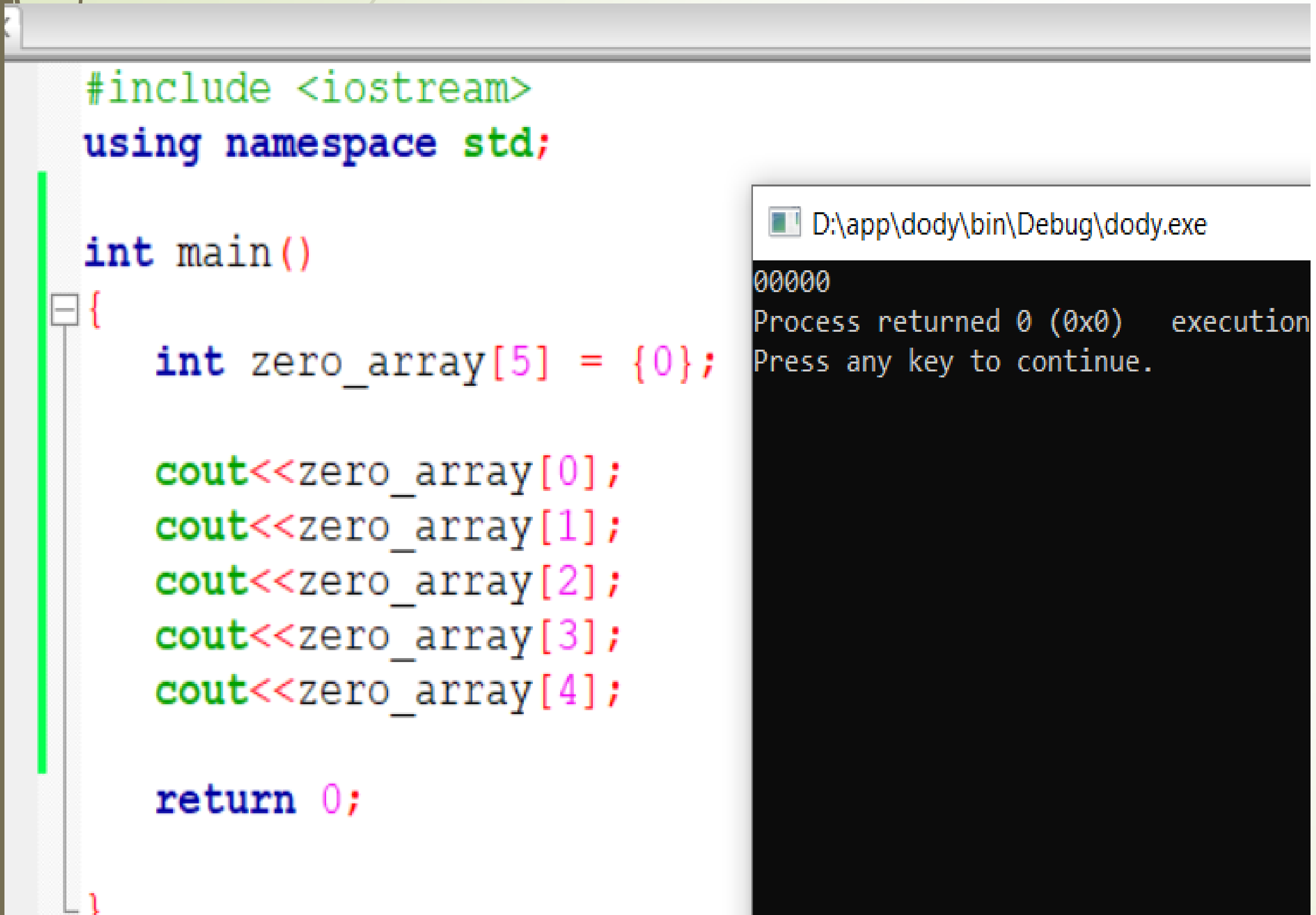
Initialize Array after Declaration (Using Loops)



```
#include <iostream>
using namespace std;

int main()
{
   int arr[6];

  for (int i = 0; i < 6; i++) {
    arr[i] = i;
   }

   return 0;

}
```

Initialize the array with zero in C++

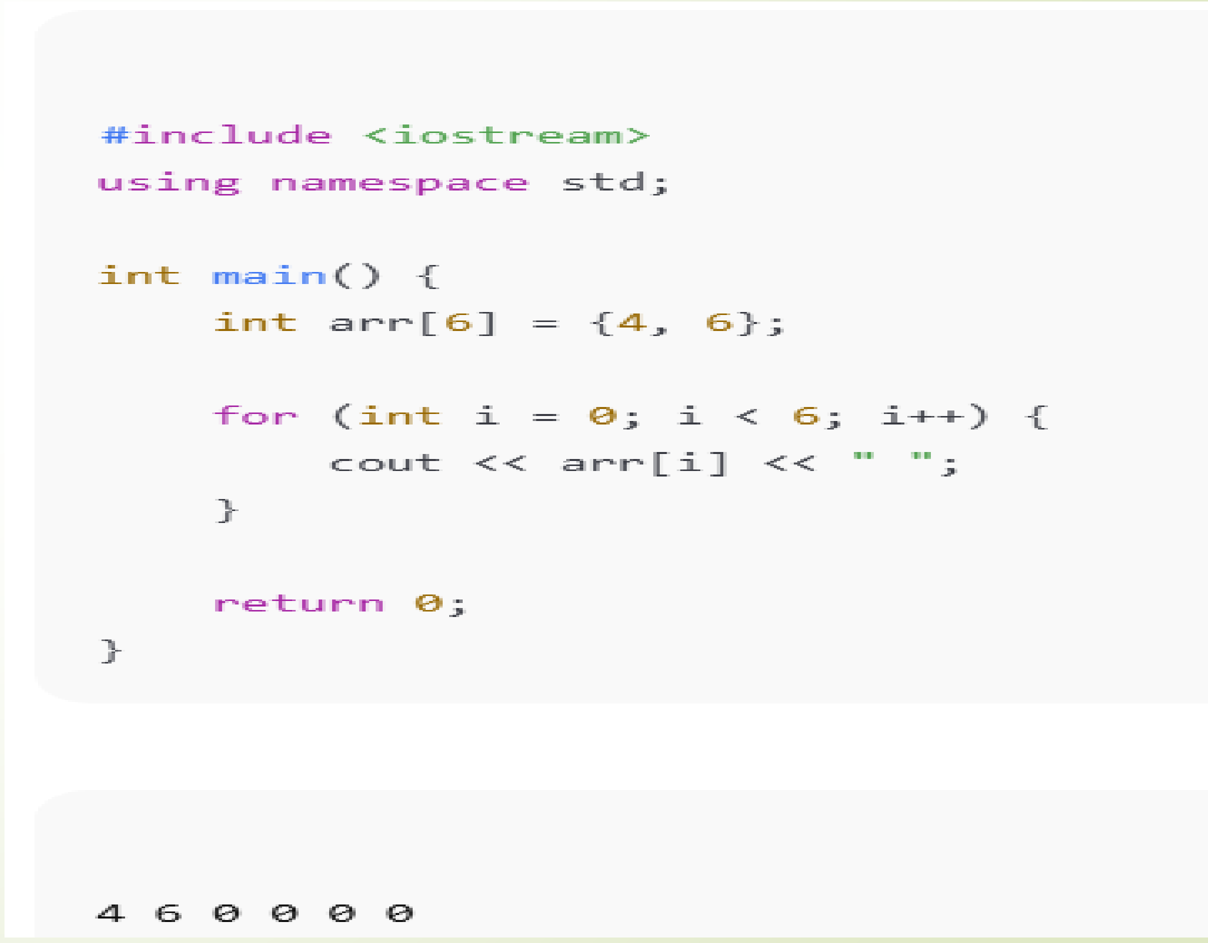
- We can initialize the array with all elements as '0' by specifying '0' inside the curly braces. This will happen in case of zero only



```cpp
#include <iostream>
using namespace std;

int main()
{
    int zero_array[5] = {0};

    cout<<zero_array[0];
    cout<<zero_array[1];
    cout<<zero_array[2];
    cout<<zero_array[3];
    cout<<zero_array[4];

    return 0;
}
```

D:\app\dody\bin\Debug\dody.exe

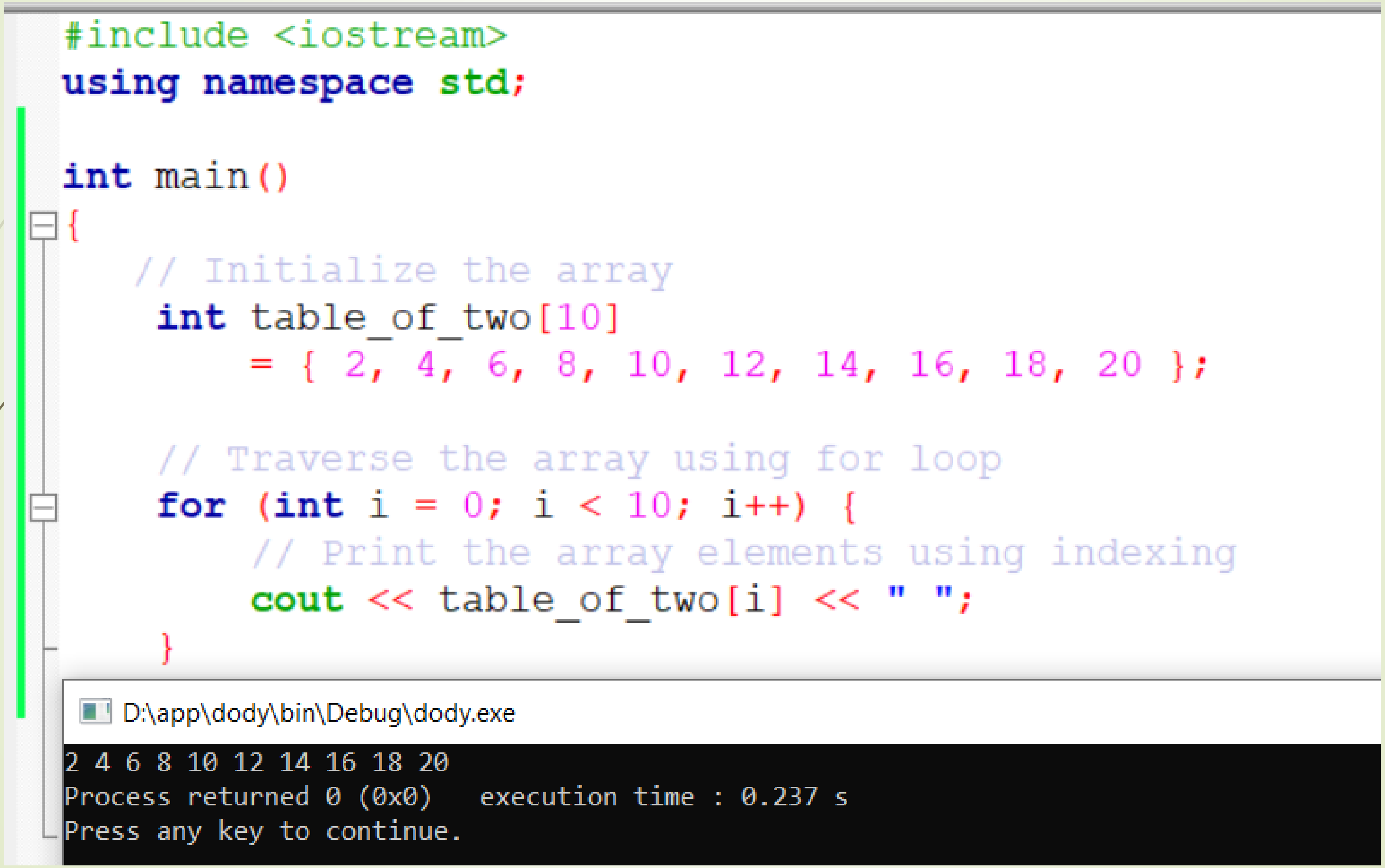
```
00000
Process returned 0 (0x0)   execution
Press any key to continue.
```



```cpp
#include <iostream>
using namespace std;

int main() {
    int arr[6] = {4, 6};

    for (int i = 0; i < 6; i++) {
        cout << arr[i] << " ";
    }

    return 0;
}
```

```
4 6 0 0 0 0
```

We can traverse over the array with the help of a loop

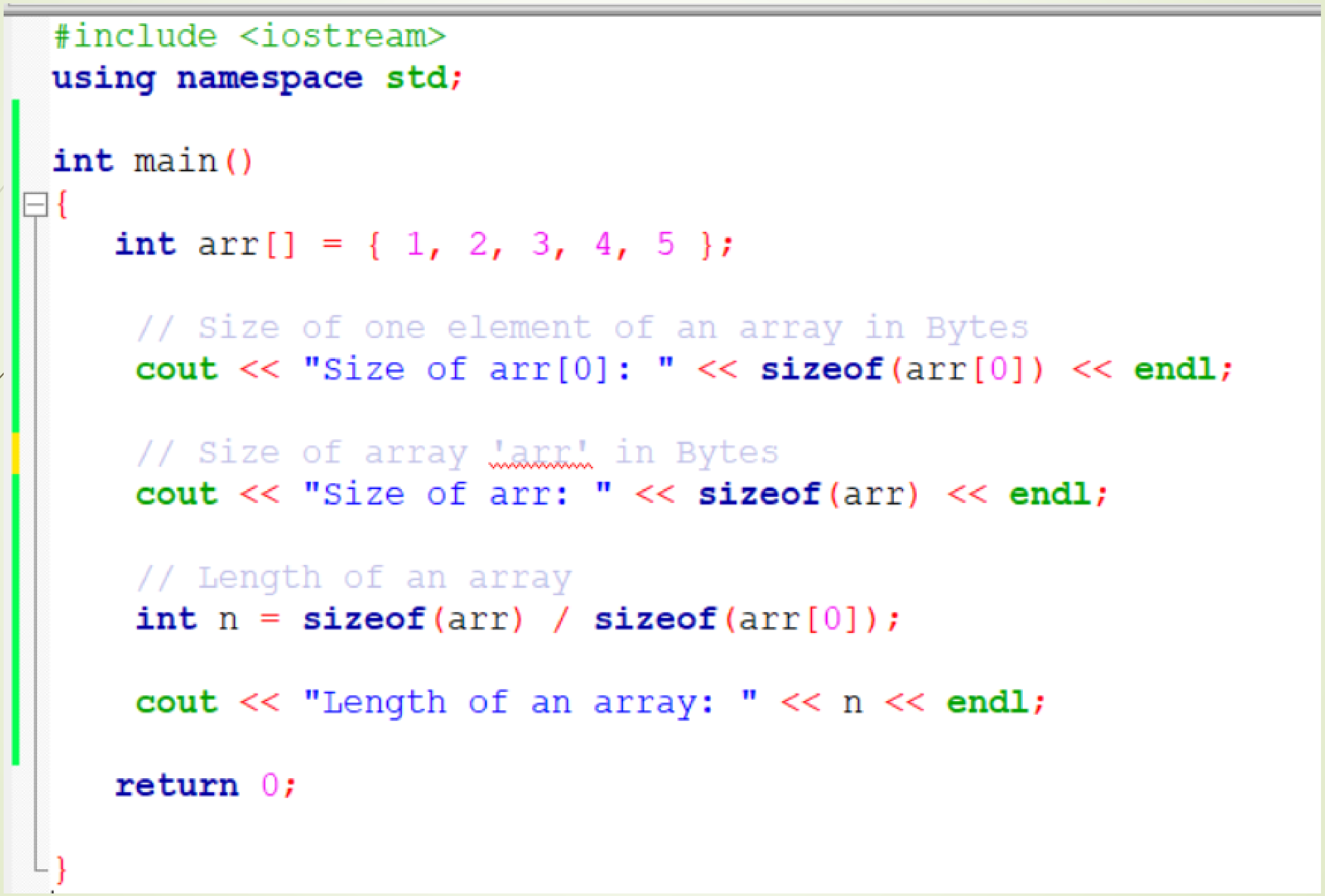


```
#include <iostream>
using namespace std;

int main()
{
   // Initialize the array
    int table_of_two[10]
        = { 2, 4, 6, 8, 10, 12, 14, 16, 18, 20 };

    // Traverse the array using for loop
    for (int i = 0; i < 10; i++) {
        // Print the array elements using indexing
        cout << table_of_two[i] << " ";
    }
```

D:\app\dody\bin\Debug\dody.exe

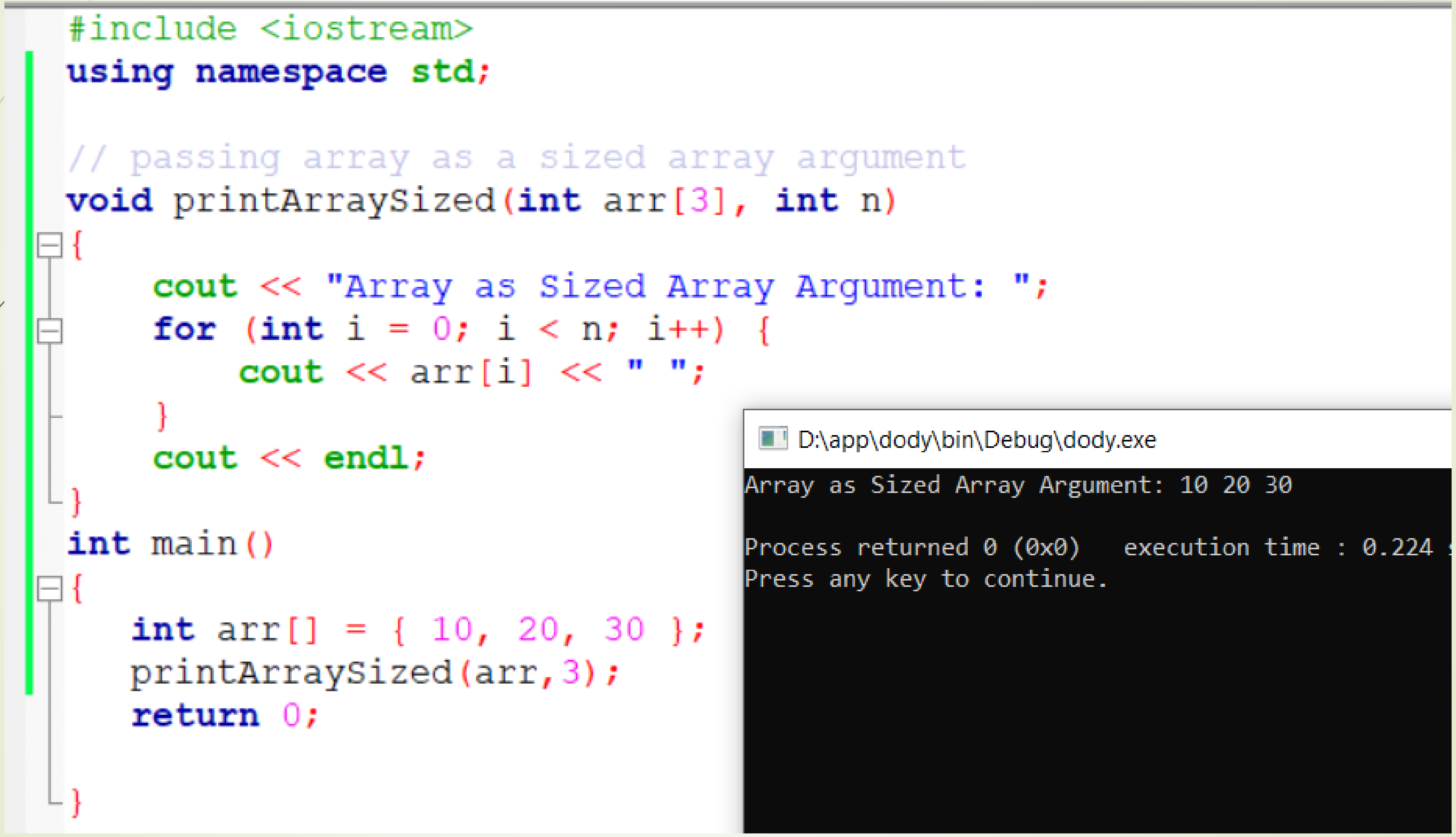
```
2 4 6 8 10 12 14 16 18 20
Process returned 0 (0x0)   execution time : 0.237 s
Press any key to continue.
```

Size of an Array in C++



```cpp
#include <iostream>
using namespace std;

int main()
{
    int arr[] = { 1, 2, 3, 4, 5 };

     // Size of one element of an array in Bytes
     cout << "Size of arr[0]: " << sizeof(arr[0]) << endl;

     // Size of array 'arr' in Bytes
     cout << "Size of arr: " << sizeof(arr) << endl;

     // Length of an array
     int n = sizeof(arr) / sizeof(arr[0]);

     cout << "Length of an array: " << n << endl;

    return 0;
}
```

Example_1: Illustrating Different Ways to Pass Arrays to a Function

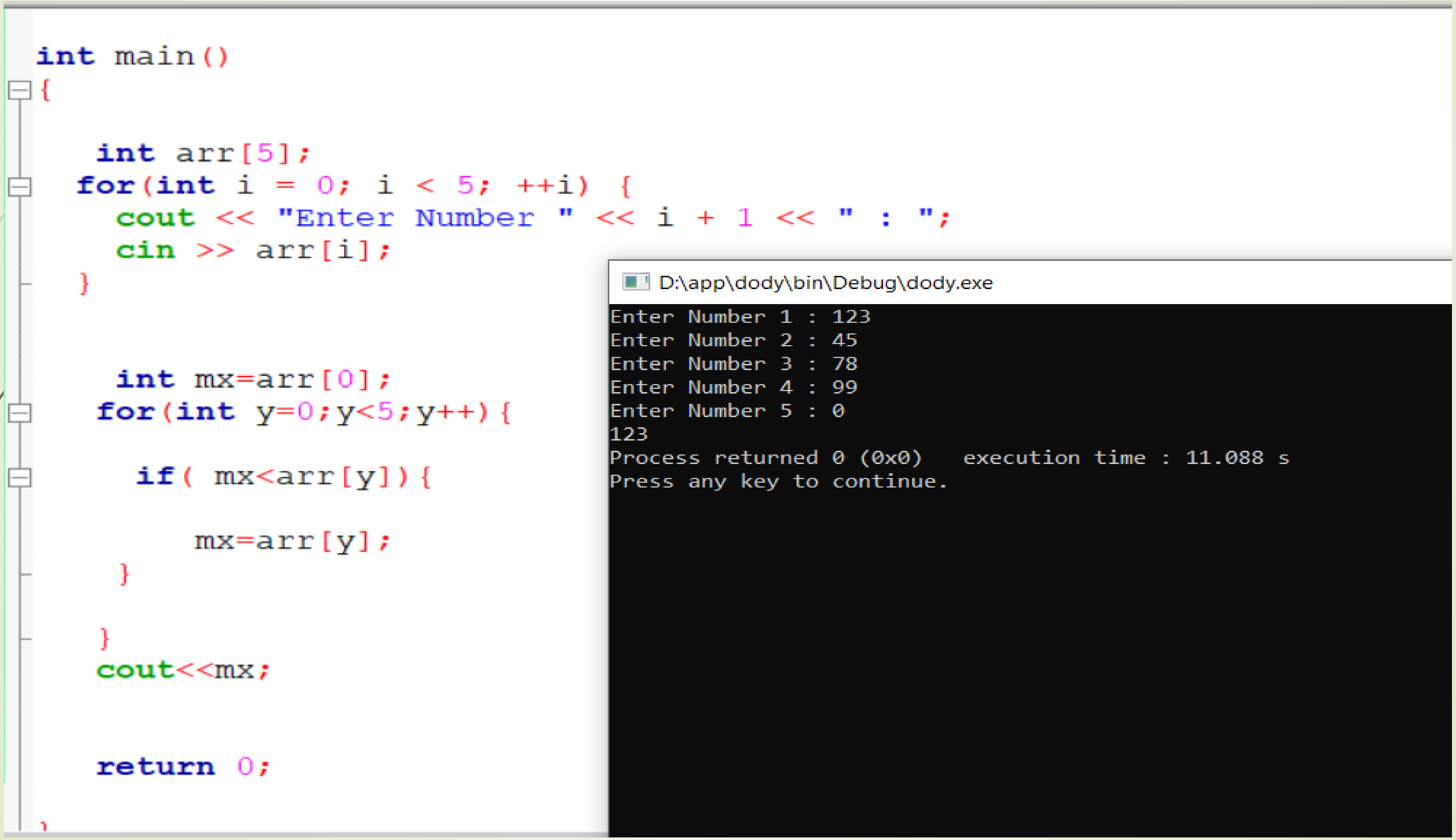


```cpp
#include <iostream>
using namespace std;

// passing array as a sized array argument
void printArraySized(int arr[3], int n)
{
    cout << "Array as Sized Array Argument: ";
    for (int i = 0; i < n; i++) {
        cout << arr[i] << " ";
    }
    cout << endl;
}
int main()
{
   int arr[] = { 10, 20, 30 };
   printArraySized(arr,3);
   return 0;

}
```

D:\app\dody\bin\Debug\dody.exe

```
Array as Sized Array Argument: 10 20 30

Process returned 0 (0x0)   execution time : 0.224
Press any key to continue.
```

Example_2: Find the largest element of a given array of integers

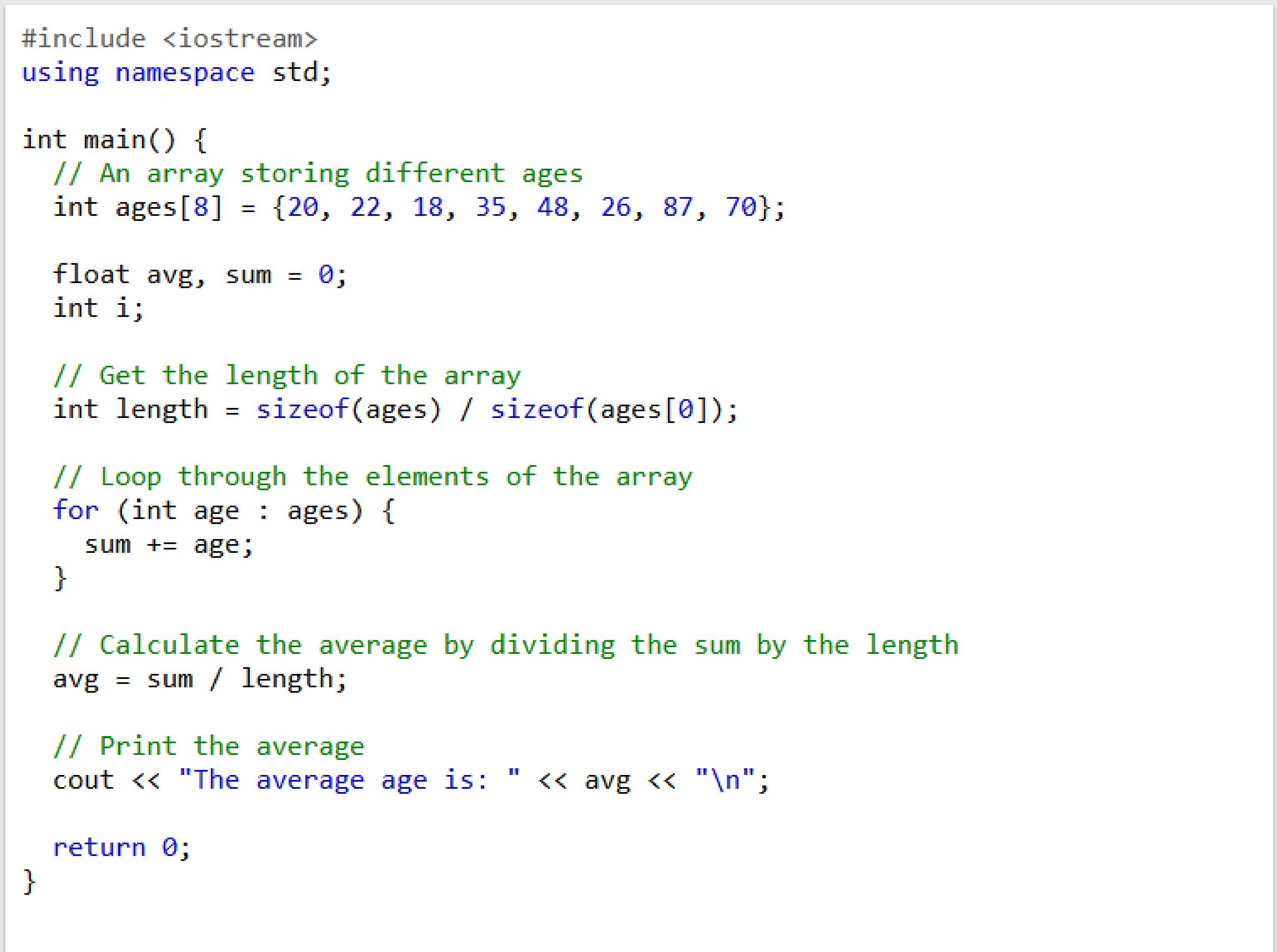


```
int main()
{

   int arr[5];
  for(int i = 0; i < 5; ++i) {
    cout << "Enter Number " << i + 1 << " : ";
    cin >> arr[i];
  }


    int mx=arr[0];
   for(int y=0;y<5;y++){

     if( mx<arr[y]){

        mx=arr[y];
    }

   }
   cout<<mx;


   return 0;
```

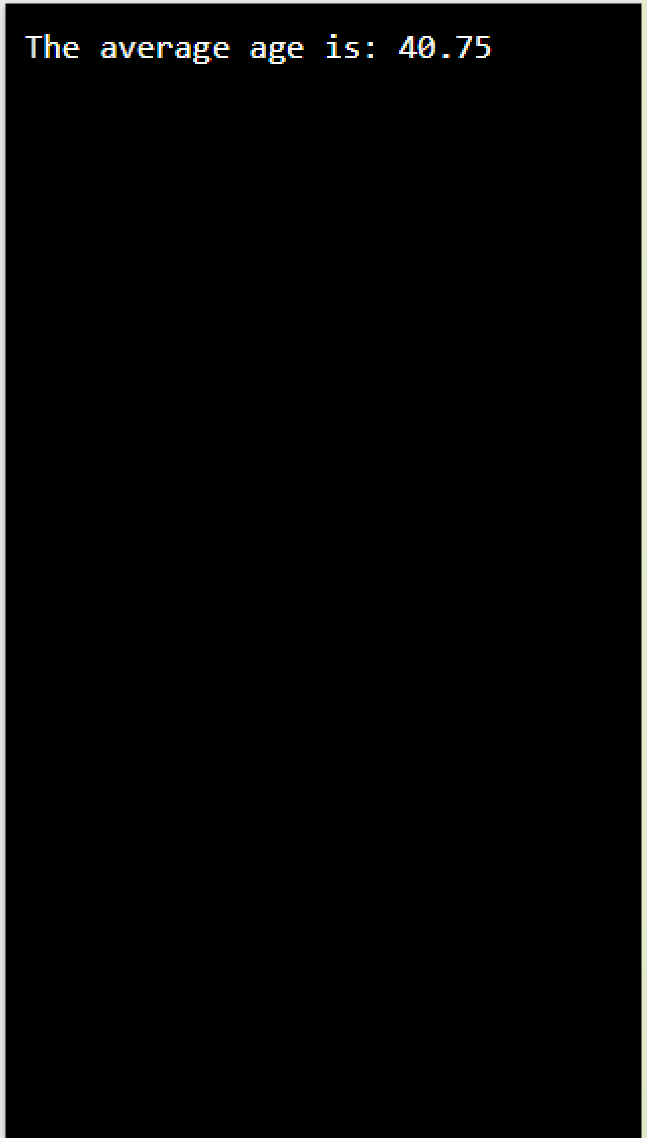
D:\app\dody\bin\Debug\dody.exe

```
Enter Number 1 : 123
Enter Number 2 : 45
Enter Number 3 : 78
Enter Number 4 : 99
Enter Number 5 : 0
123
Process returned 0 (0x0)   execution time : 11.088 s
Press any key to continue.
```

Example3: calculates the average of different ages

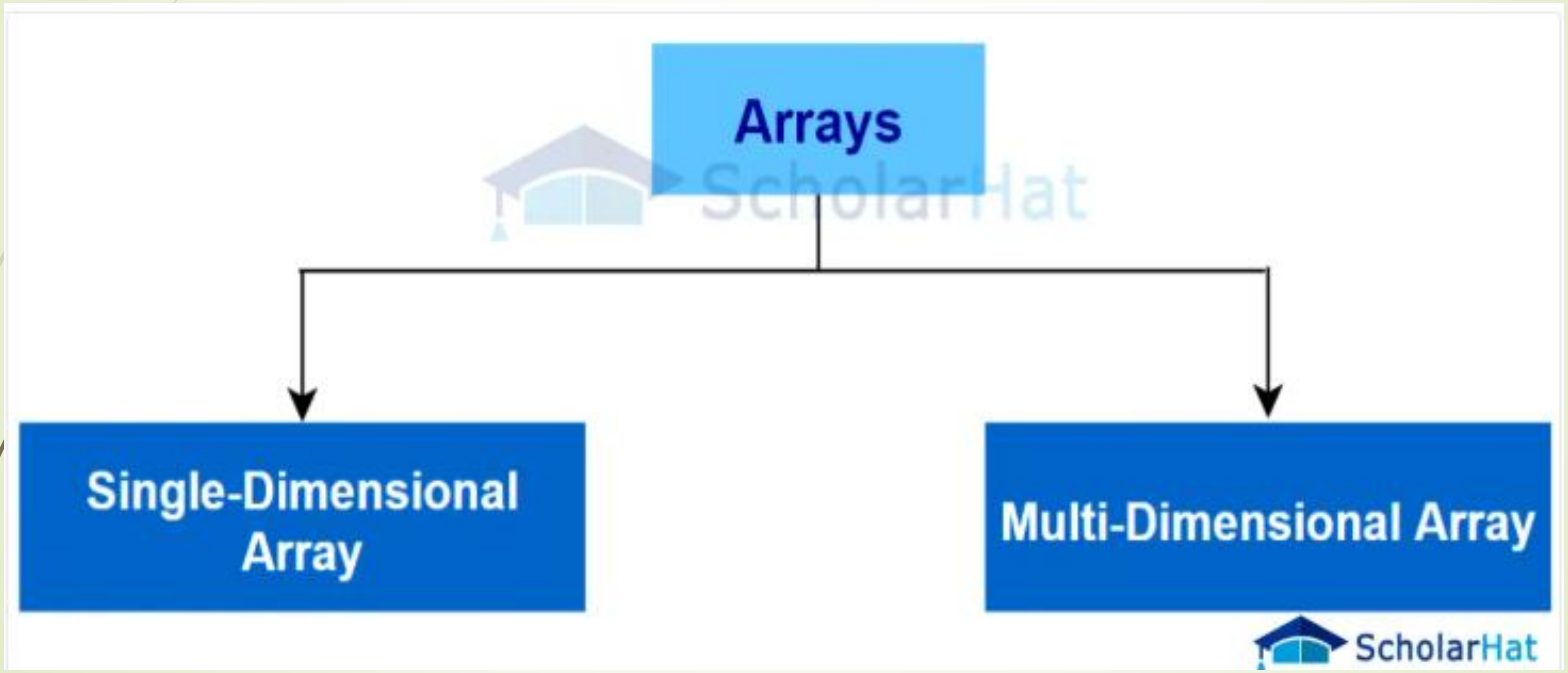


```cpp
#include <iostream>
using namespace std;

int main() {
  // An array storing different ages
  int ages[8] = {20, 22, 18, 35, 48, 26, 87, 70};

  float avg, sum = 0;
  int i;

  // Get the length of the array
  int length = sizeof(ages) / sizeof(ages[0]);

  // Loop through the elements of the array
  for (int age : ages) {
    sum += age;
  }

  // Calculate the average by dividing the sum by the length
  avg = sum / length;

  // Print the average
  cout << "The average age is: " << avg << "\n";

  return 0;
}
```



```
The average age is: 40.75
```

Types of Arrays in C++



Multidimensional Arrays in C++

- Arrays declared with more than one dimension are called multidimensional arrays. The most widely used multidimensional arrays are 2D arrays and 3D arrays. These arrays are generally represented in the form of rows and columns. **Multidimensional Array Declaration:**

```
Data_Type Array_Name[Size1][Size2]...[SizeN];
```

Where,

- **Data_Type:** Type of data to be stored in the array.
- **Array_Name:** Name of the array.
- **Size1, Size2,…, SizeN:** Size of each dimension.

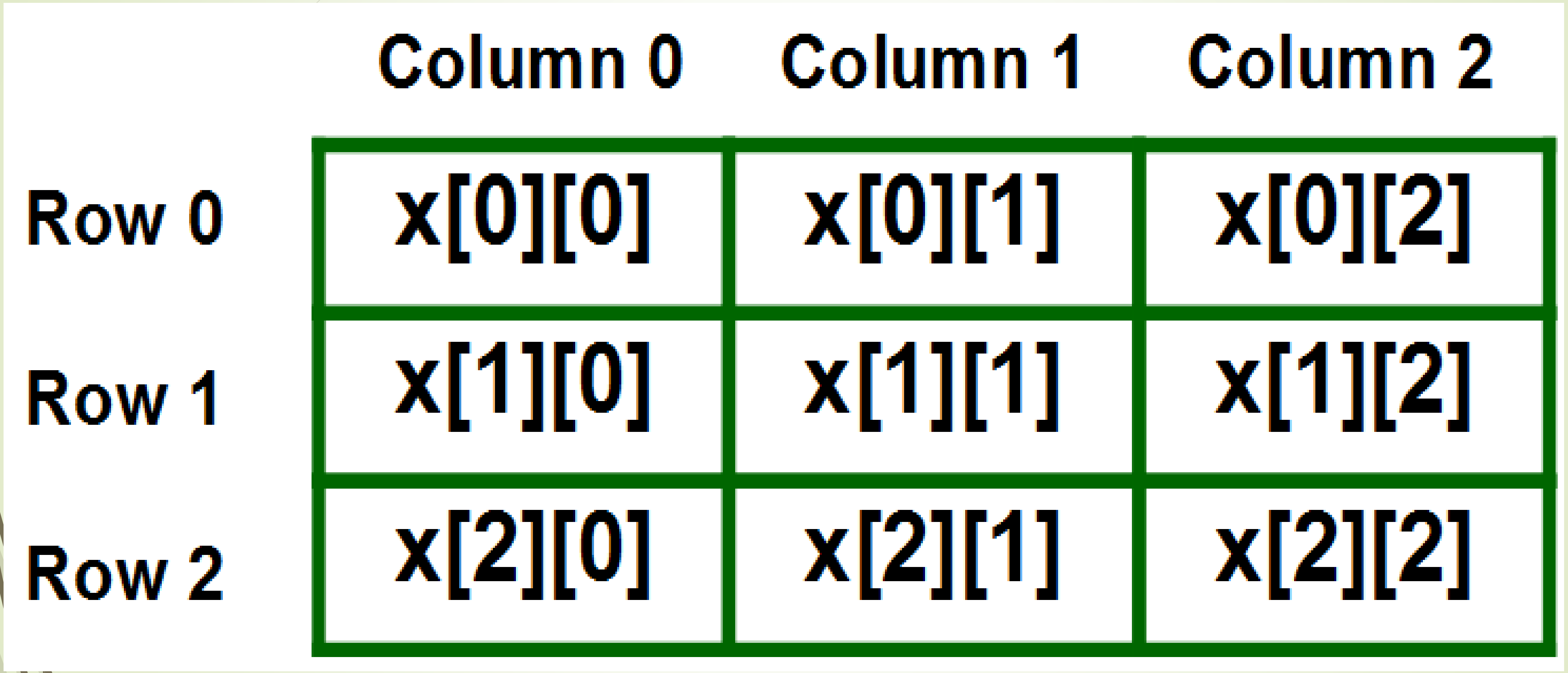
Two-Dimensional Array in C++

- In C++, a two-dimensional array is a grouping of elements arranged in rows and columns. Each element is accessed using two indices: one for the row and one for the column, which makes it easy to visualize as a table or grid.
- **Syntax of 2D array**

```
data_Type array_name[n][m];
```

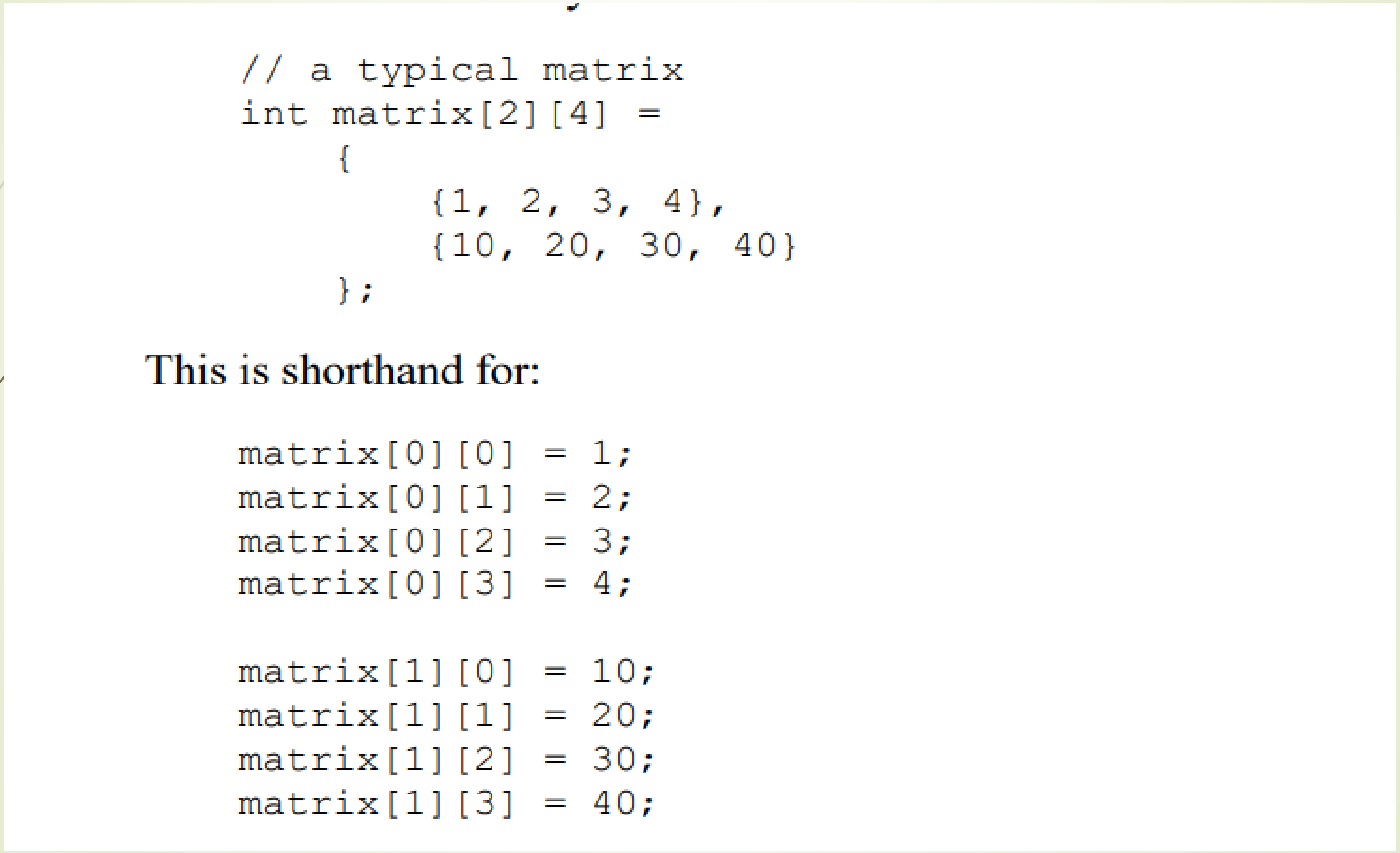
Where

- n: Number of rows.
- m: Number of columns



	Column 0	Column 1	Column 2
Row 0	x[0][0]	x[0][1]	x[0][2]
Row 1	x[1][0]	x[1][1]	x[1][2]
Row 2	x[2][0]	x[2][1]	x[2][2]

The array is initialized as follows

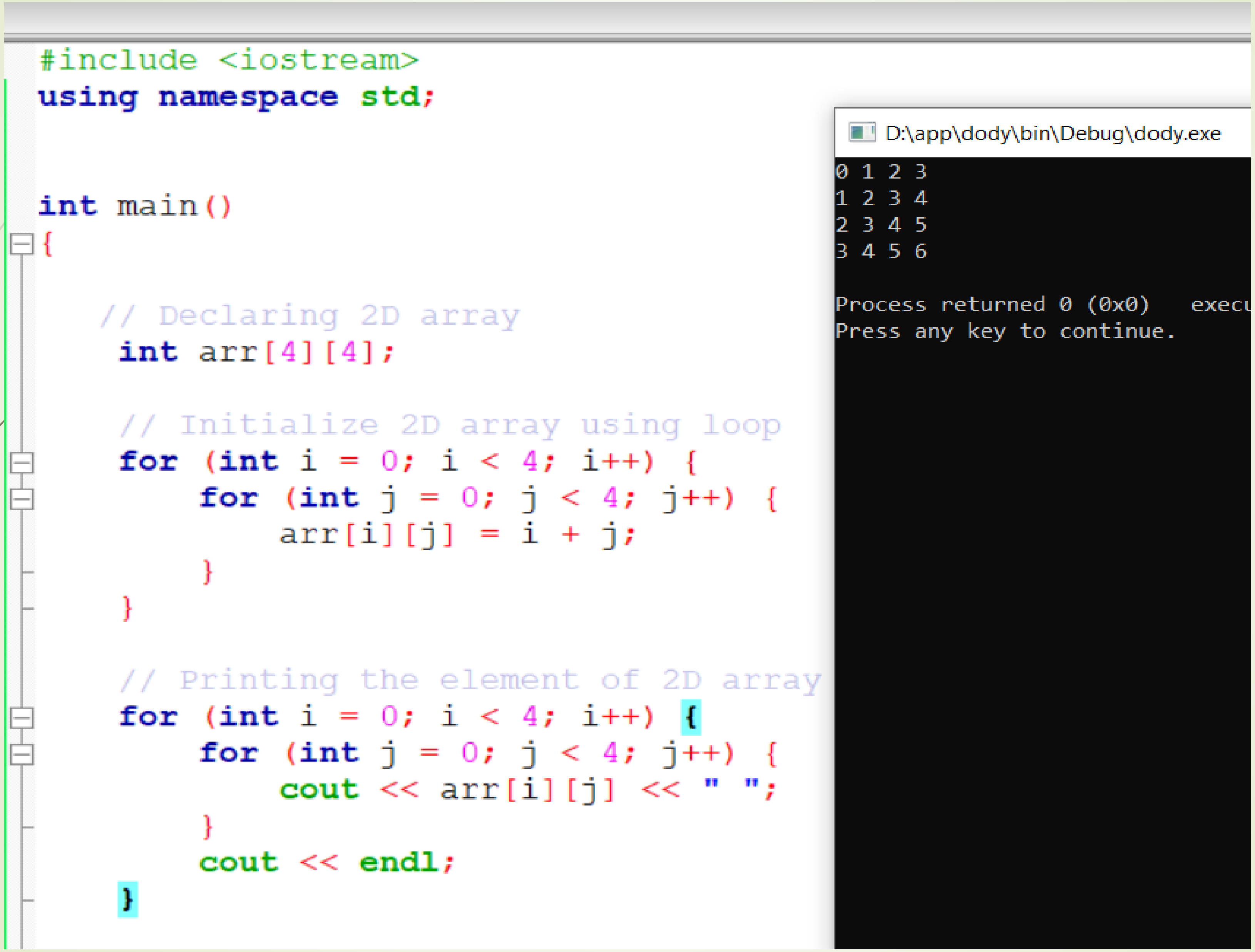


```
// a typical matrix
int matrix[2][4] =
    {
        {1, 2, 3, 4},
        {10, 20, 30, 40}
    };
```

This is shorthand for:

```
matrix[0][0] = 1;
matrix[0][1] = 2;
matrix[0][2] = 3;
matrix[0][3] = 4;

matrix[1][0] = 10;
matrix[1][1] = 20;
matrix[1][2] = 30;
matrix[1][3] = 40;
```

C++ program to illustrate the two-dimensional array

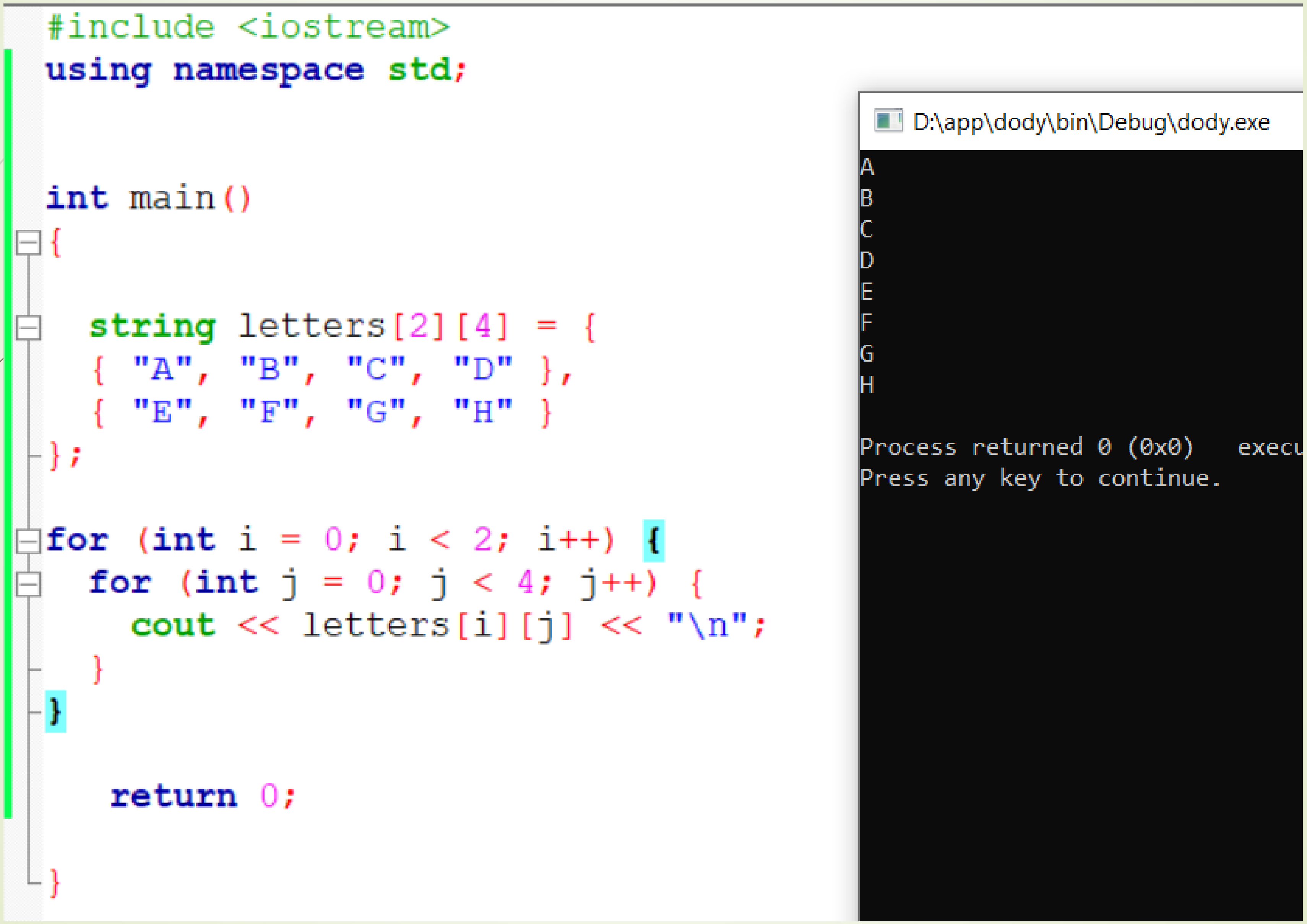


```cpp
#include <iostream>
using namespace std;


int main()
{

    // Declaring 2D array
    int arr[4][4];

    // Initialize 2D array using loop
    for (int i = 0; i < 4; i++) {
        for (int j = 0; j < 4; j++) {
            arr[i][j] = i + j;
        }
    }

    // Printing the element of 2D array
    for (int i = 0; i < 4; i++) {
        for (int j = 0; j < 4; j++) {
            cout << arr[i][j] << " ";
        }
        cout << endl;
    }
```

D:\app\dody\bin\Debug\dody.exe

```
0 1 2 3
1 2 3 4
2 3 4 5
3 4 5 6

Process returned 0 (0x0)   execu
Press any key to continue.
```

Example2



```cpp
#include <iostream>
using namespace std;


int main()
{

  string letters[2][4] = {
  { "A", "B", "C", "D" },
  { "E", "F", "G", "H" }
};

for (int i = 0; i < 2; i++) {
  for (int j = 0; j < 4; j++) {
    cout << letters[i][j] << "\n";
  }
}

   return 0;

}
```

D:\app\dody\bin\Debug\dody.exe

```
A
B
C
D
E
F
G
H

Process returned 0 (0x0)   execu
Press any key to continue.
```

Three-Dimensional Array in C++

The 3D array uses three dimensions. A collection of various two-dimensional arrays piled on top of one another can be used to represent it. Three indices—the row index, column index, and depth index are used to uniquely identify each element in a 3D array.

- **Declaration of Three-Dimensional Array in C++**
- To declare a 3D array in C++, we need to specify its third dimension along with 2D dimensions.

```
Data_Type Array_Name[D][R][C];
```

Where:

•**Data_Type:** Type of data to be stored in each element.

•**Array_Name:** Name of the array

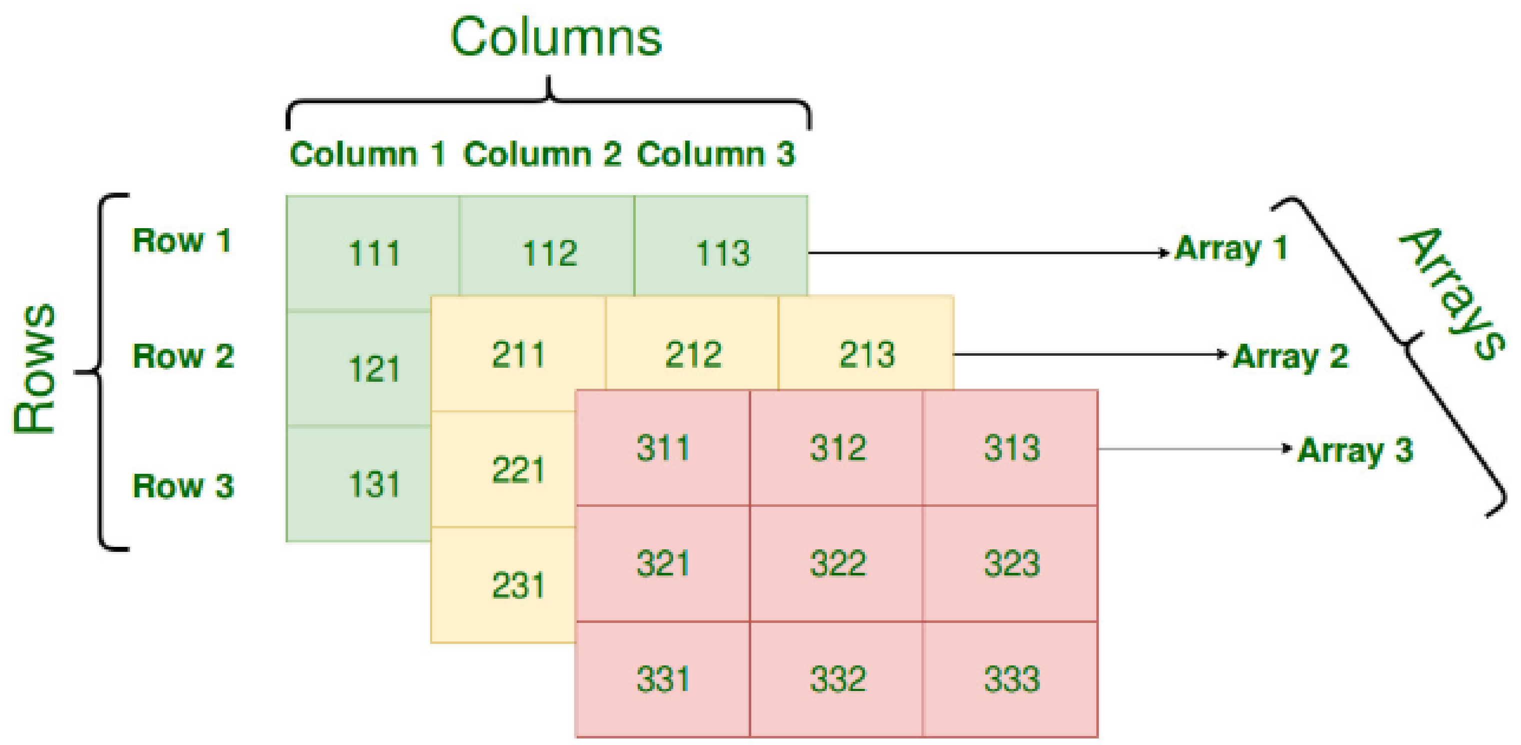
•**D:** Number of 2D arrays or Depth of array.

•**R:** Number of rows in each 2D array.

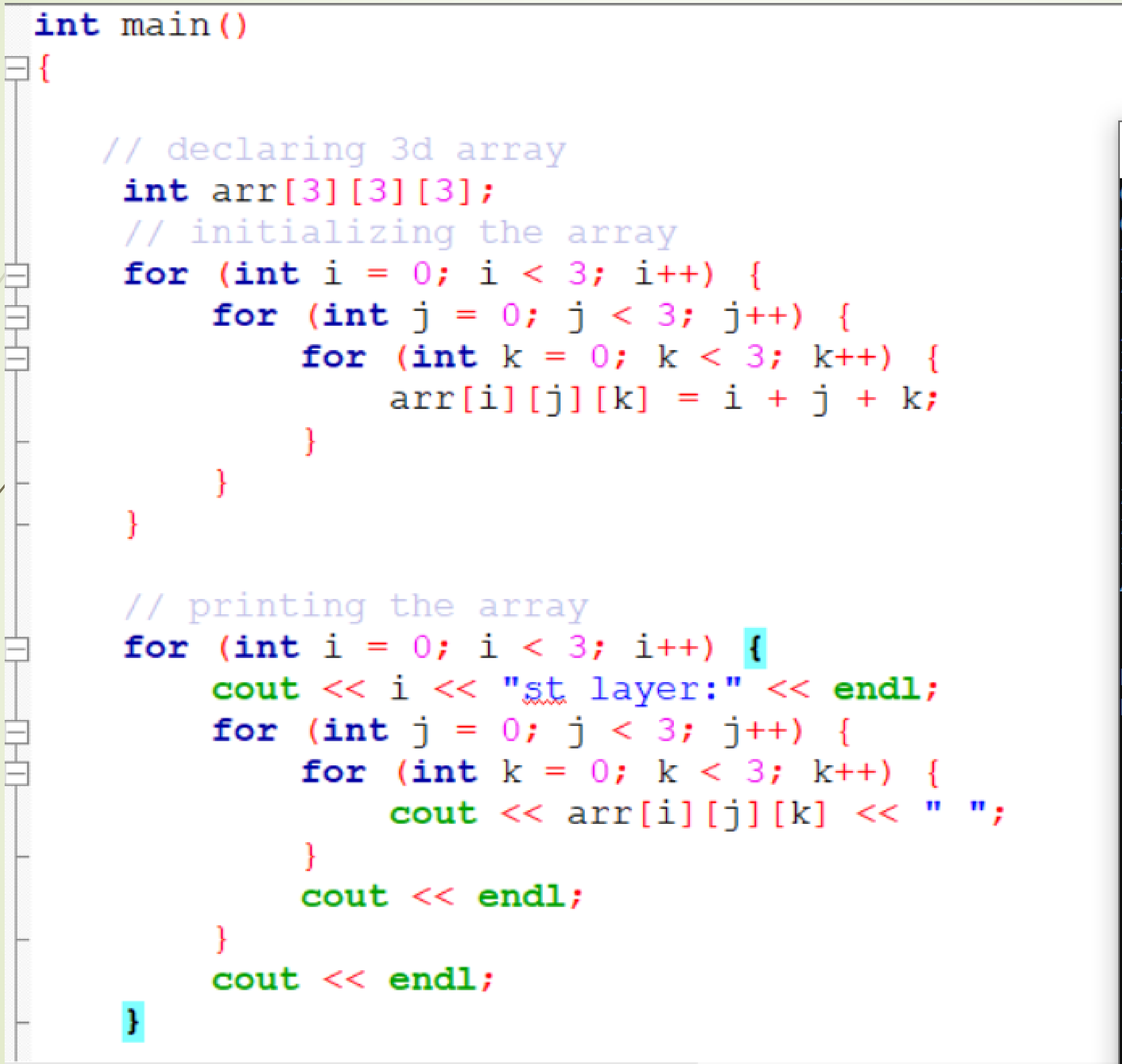
•**C:** Number of columns in each 2D array.

Example

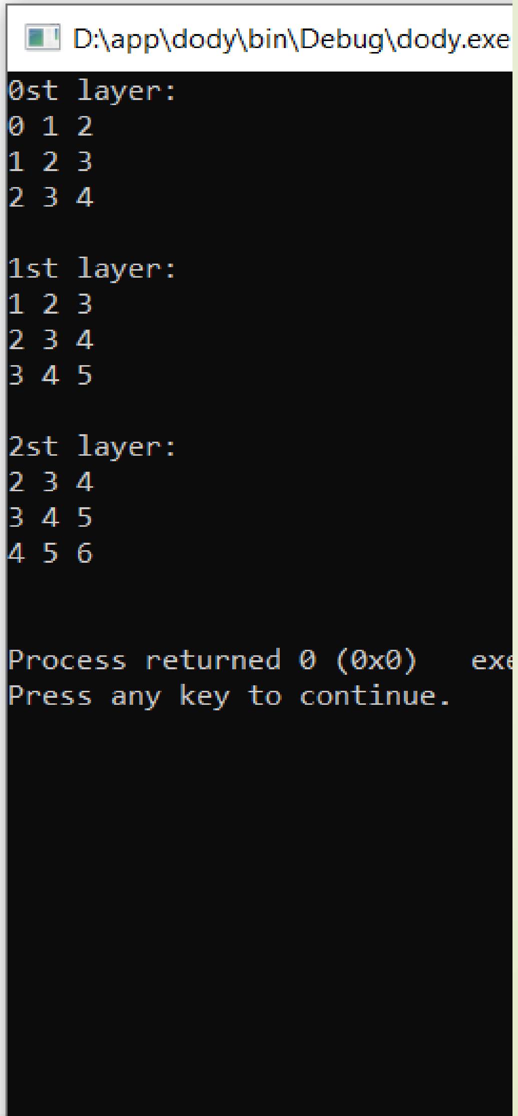
```
int array[3][3][3];
```



C++ program to illustrate the 3d array

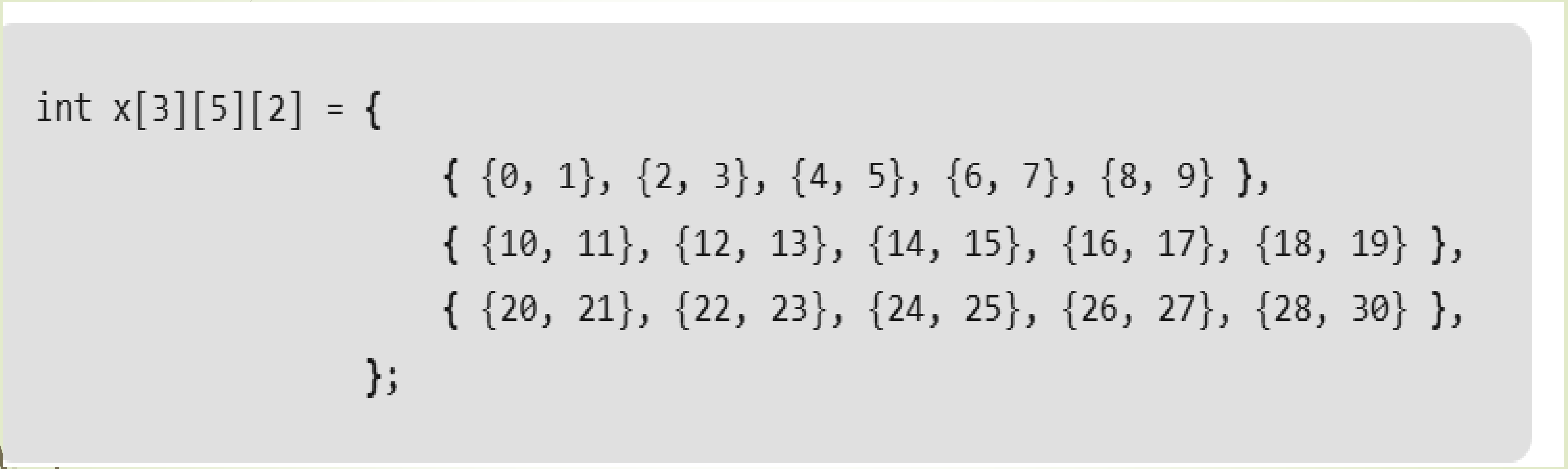


```
int main()
{

    // declaring 3d array
    int arr[3][3][3];
    // initializing the array
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            for (int k = 0; k < 3; k++) {
                arr[i][j][k] = i + j + k;
            }
        }
    }

    // printing the array
    for (int i = 0; i < 3; i++) {
        cout << i << "st layer:" << endl;
        for (int j = 0; j < 3; j++) {
            for (int k = 0; k < 3; k++) {
                cout << arr[i][j][k] << " ";
            }
            cout << endl;
        }
        cout << endl;
    }
```



D:\app\dody\bin\Debug\dody.exe

```
0st layer:
0 1 2
1 2 3
2 3 4

1st layer:
1 2 3
2 3 4
3 4 5

2st layer:
2 3 4
3 4 5
4 5 6


Process returned 0 (0x0)   exe
Press any key to continue.
```



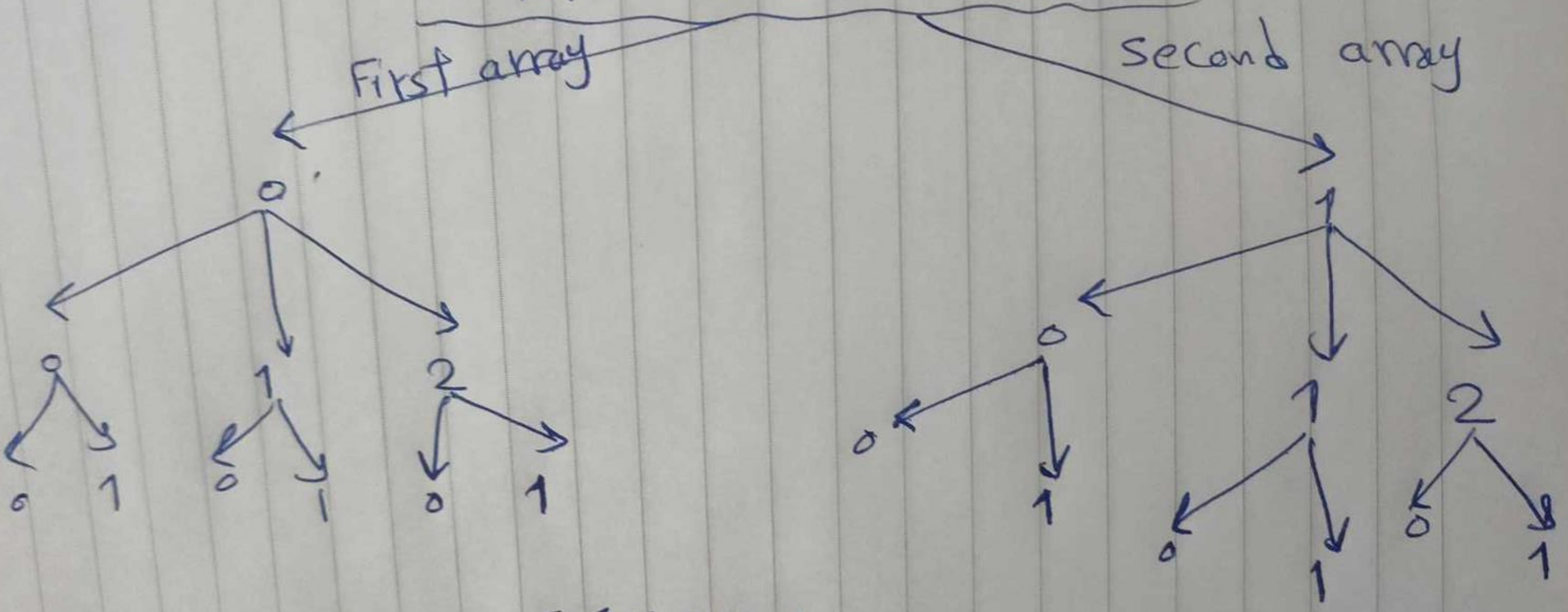
```
int x[3][5][2] = {
                    { {0, 1}, {2, 3}, {4, 5}, {6, 7}, {8, 9} },
                    { {10, 11}, {12, 13}, {14, 15}, {16, 17}, {18, 19} },
                    { {20, 21}, {22, 23}, {24, 25}, {26, 27}, {28, 30} },
                };
```

```
// Create and Initialize the 3-dimensional array
int arr[2][3][2] = { { { 1, 1 }, { 2, 3 }, { 4, 5 } },
                     { { 6, 7 }, { 8, 9 }, { 10, 11 } } };
```

```
arr[0][0][0] = 1   arr[0][0][1] = 1
arr[0][1][0] = 2   arr[0][1][1] = 3
arr[0][2][0] = 4   arr[0][2][1] = 5



arr[1][0][0] = 6   arr[1][0][1] = 7
arr[1][1][0] = 8   arr[1][1][1] = 9
arr[1][2][0] = 10   arr[1][2][1] = 11
```

Example For 3D array

int arr [2] [3] [2]



Ex: arr [0] [0] [0]
arr [0] [0] [1]
arr [0] [1] [0]
arr [0] [1] [1]

Thank you

References

- https://www.geeksforgeeks.org/cpp-arrays/
- Steve Oualline," Practical C++ Programming", O'Reilly & Associates, Inc, United States of America 1995