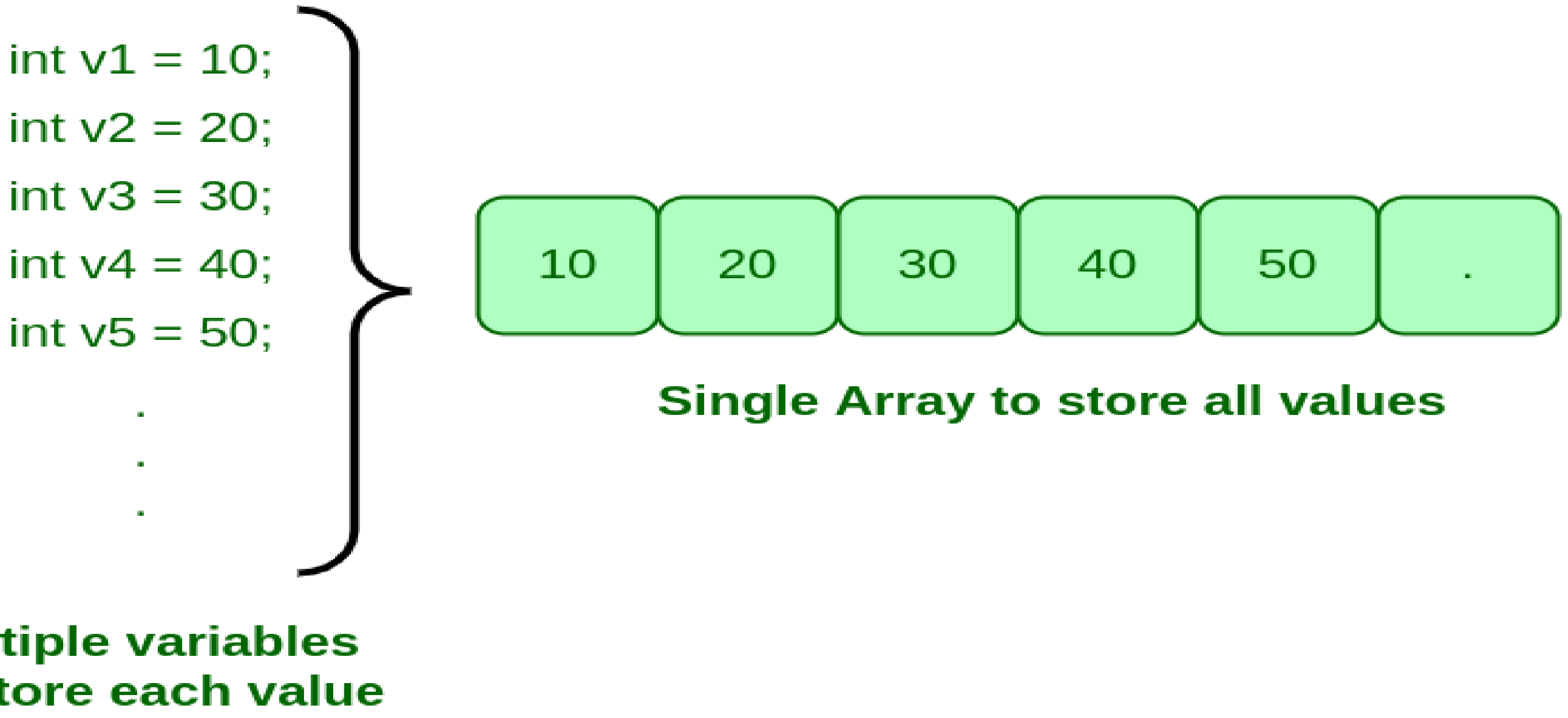
INTRODUCTION OF C++ SECTION 5 PART(1)

Dr/Ghada Maher
Eng/ Hossam Medhat

C++ Arrays

❖In C++, an array is **a variable** that can **store multiple values of the same type, instead of** declaring separate variables for each value.



ADVANTAGES & DISADVANTAGES OF ARRAYS

- **Advantages:-**
  1) Code Optimization: we can retrieve or sort the data efficiently.
  2) Random access: We can get any data located at an index position.
- **Disadvantages:-**
  - Size Limit: We can store only the fixed size of elements in the array. It doesn’t grow its size at runtime.

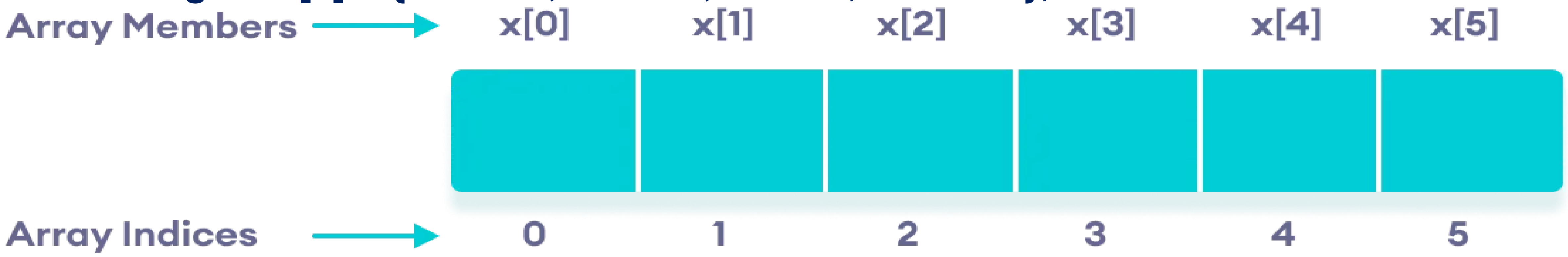
C++ ARRAYS "CONT"

❖To declare an array, define the variable type, specify the name of the array followed by square brackets and specify the number of elements it should store:

Ex : string cars[4];

❖In C++, the size and type of arrays cannot be changed after its declaration.

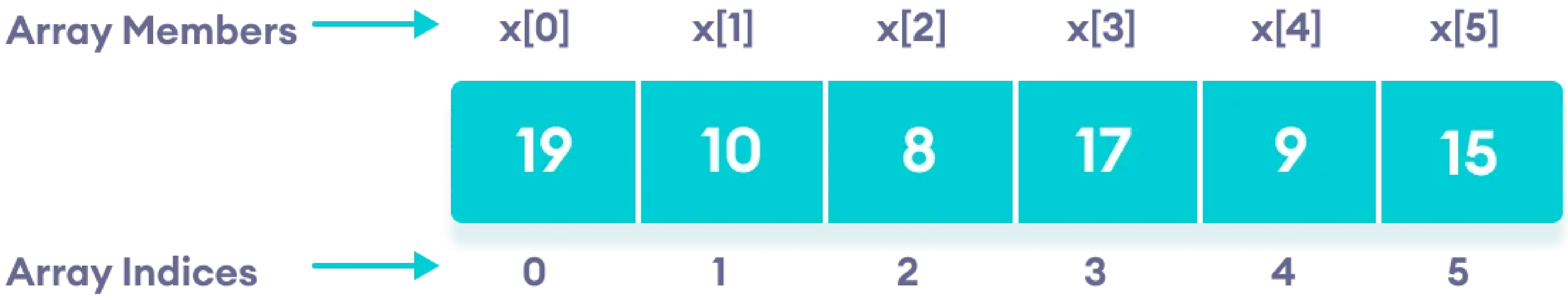
❖string cars[4] = {"Volvo", "BMW", "Ford", "Mazda"};



C++ ARRAYS "CONT"

❖ **Declare and initialize and array :**

➢ **int x[6] = {19, 10, 8, 17, 9, 15};**

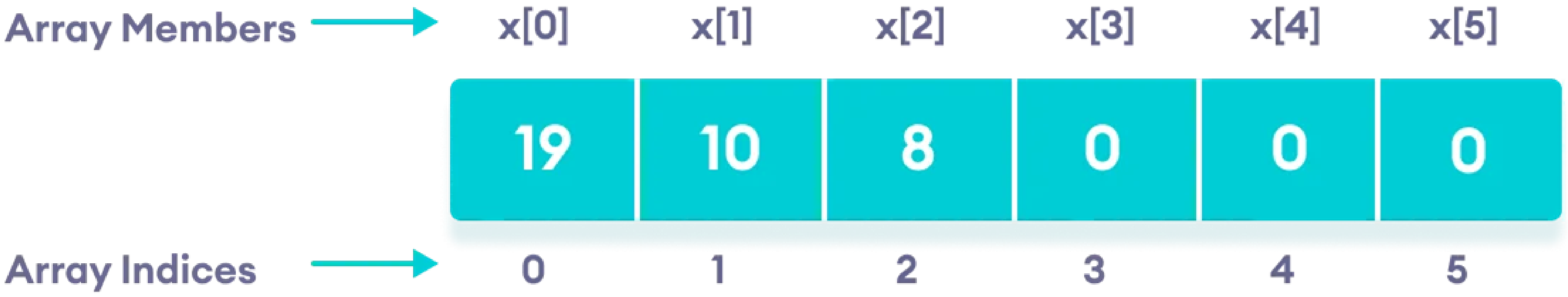


C++ Array With Empty Members

❖In C++, if an array has a size n, we can store n number of elements in the array. However, what will happen if we store less than n number of elements.

➢ Example :

➢int x[6] = {19, 10, 8}; // store only 3 elements in the array



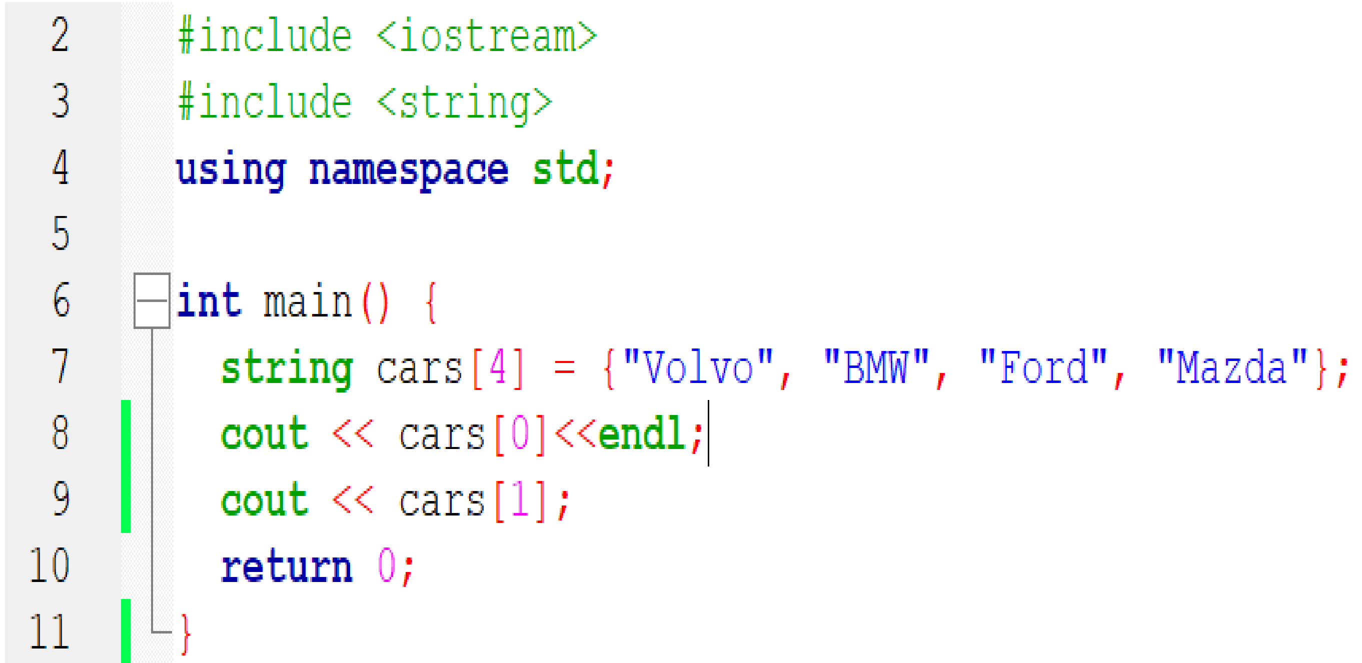
➢The compiler assigns random values to the remaining places. Oftentimes,

C++ ARRAYS "CONT"

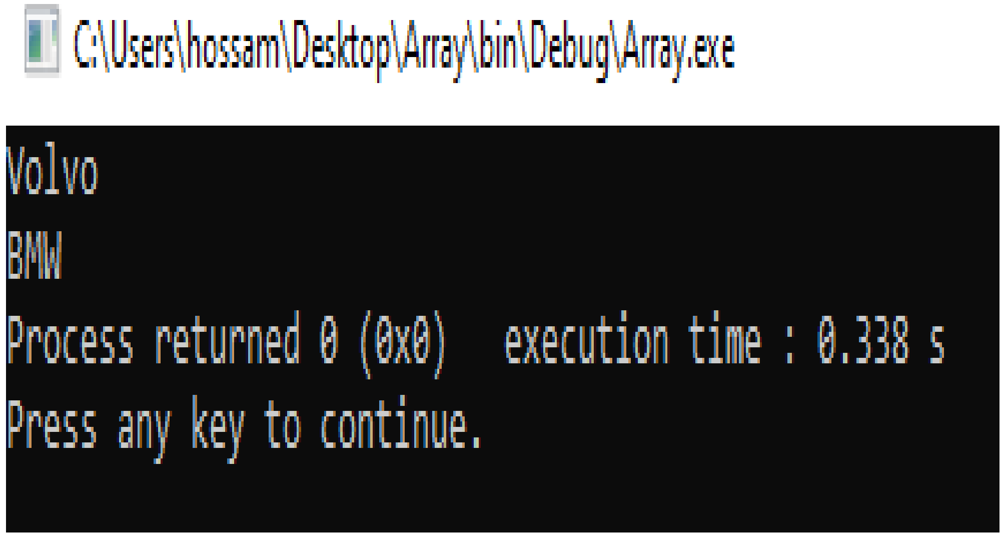
❖ **Access the Elements of an Array**

➢ **Access an array element by referring to the index number inside square brackets [].**

➢ **Note: Array indexes start with 0: [0] is the first element. [1] is the second element.**



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

int main() {
  string cars[4] = {"Volvo", "BMW", "Ford", "Mazda"};
  cout << cars[0]<<endl;
  cout << cars[1];
  return 0;
}
```



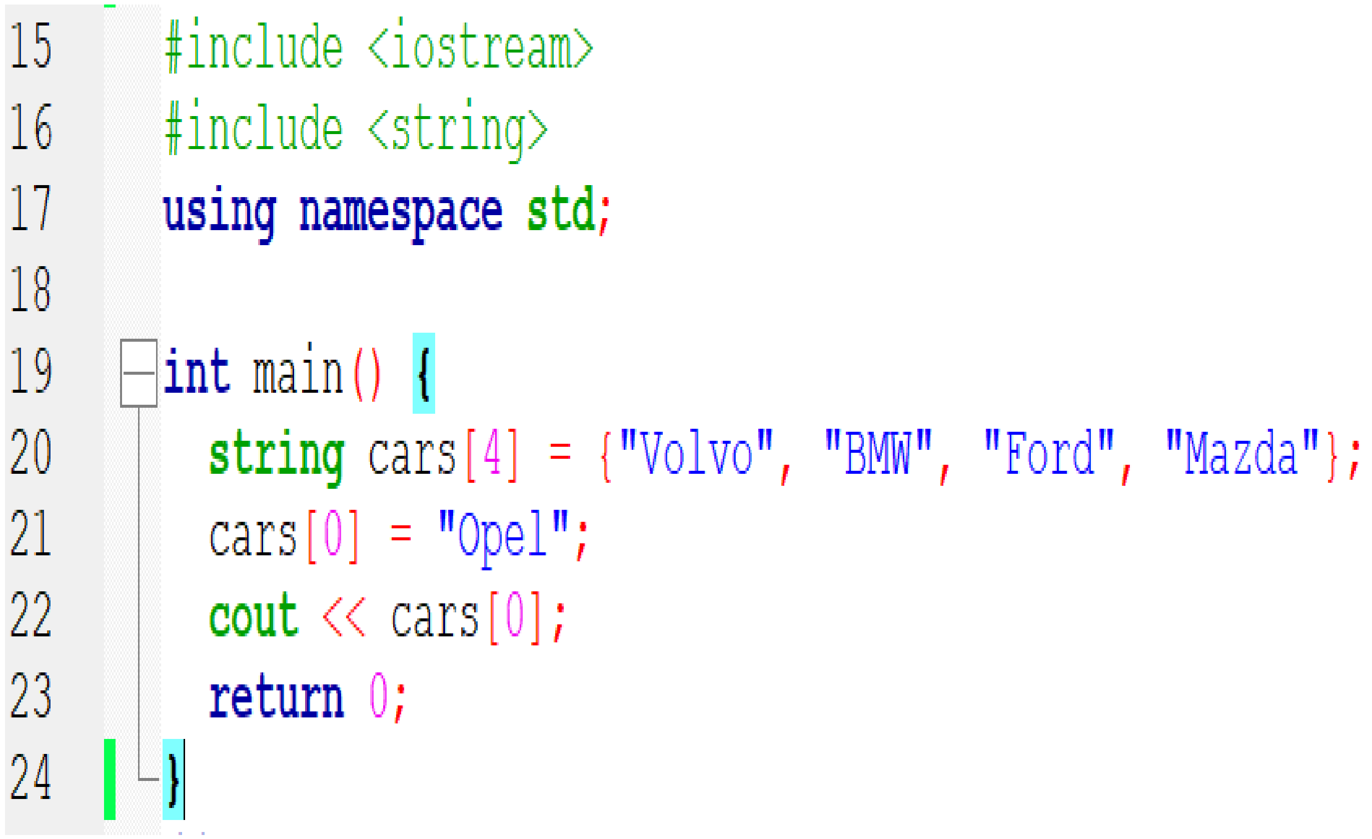
C:\Users\hossam\Desktop\Array\bin\Debug\Array.exe

```
Volvo
BMW
Process returned 0 (0x0)   execution time : 0.338 s
Press any key to continue.
```

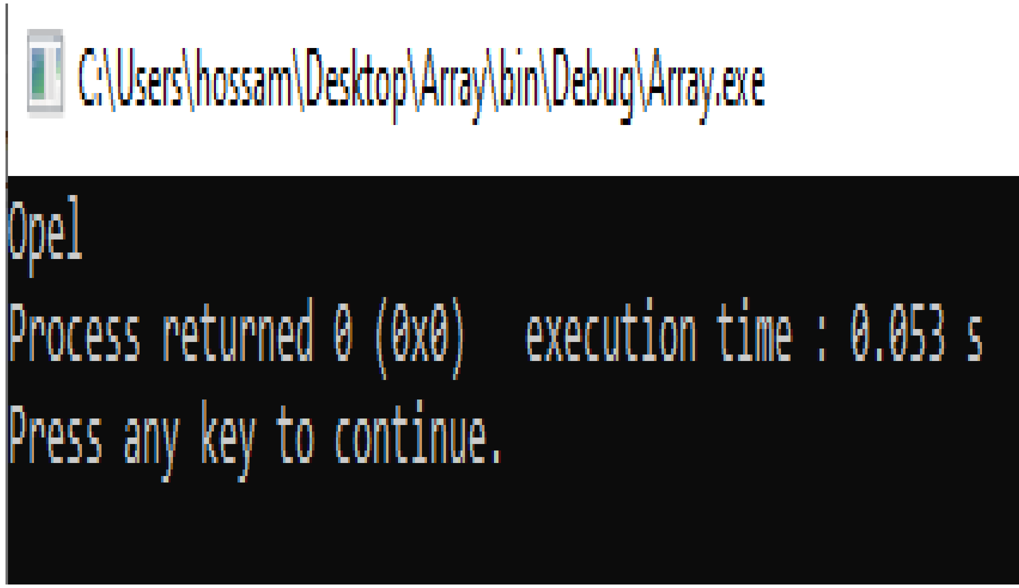
C++ Arrays "Cont"

❖ **Change an Array Element**

➢ To **change the value** of a specific element, **refer to the index number:**



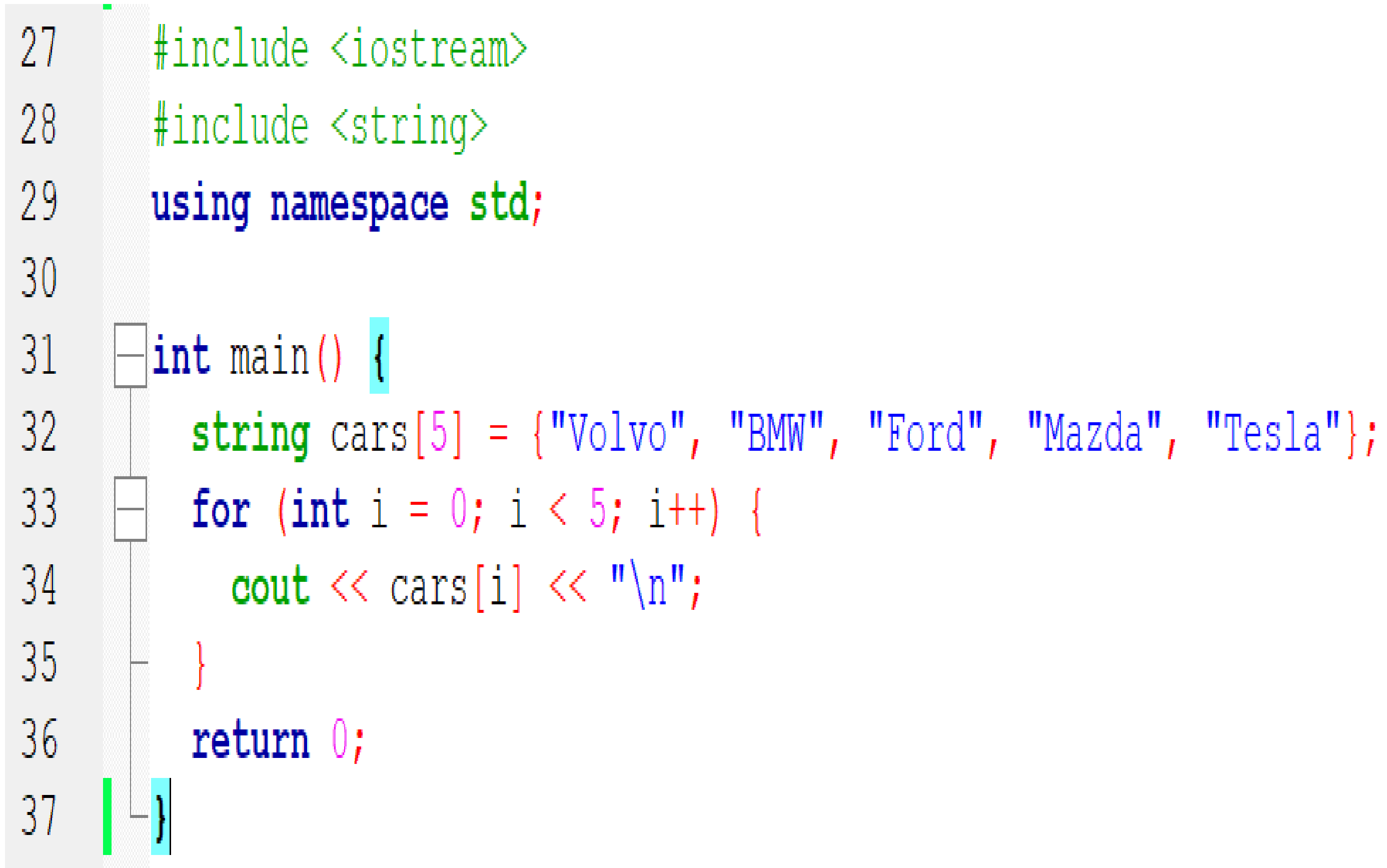
```cpp
#include <iostream>
#include <string>
using namespace std;

int main() {
    string cars[4] = {"Volvo", "BMW", "Ford", "Mazda"};
    cars[0] = "Opel";
    cout << cars[0];
    return 0;
}
```



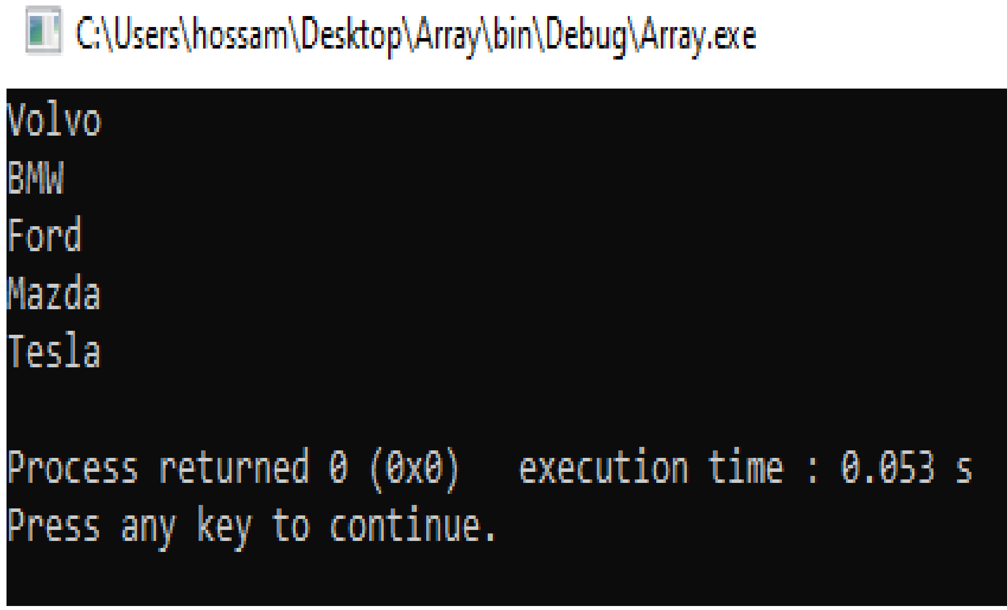
```
C:\Users\hossam\Desktop\Array\bin\Debug\Array.exe

Opel
Process returned 0 (0x0)   execution time : 0.053 s
Press any key to continue.
```

C++ Arrays and Loops

- You can loop through **the array elements with the for loop.**
- **Example :** outputs all elements in the cars array



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

int main() {
  string cars[5] = {"Volvo", "BMW", "Ford", "Mazda", "Tesla"};
  for (int i = 0; i < 5; i++) {
    cout << cars[i] << "\n";
  }
  return 0;
}
```

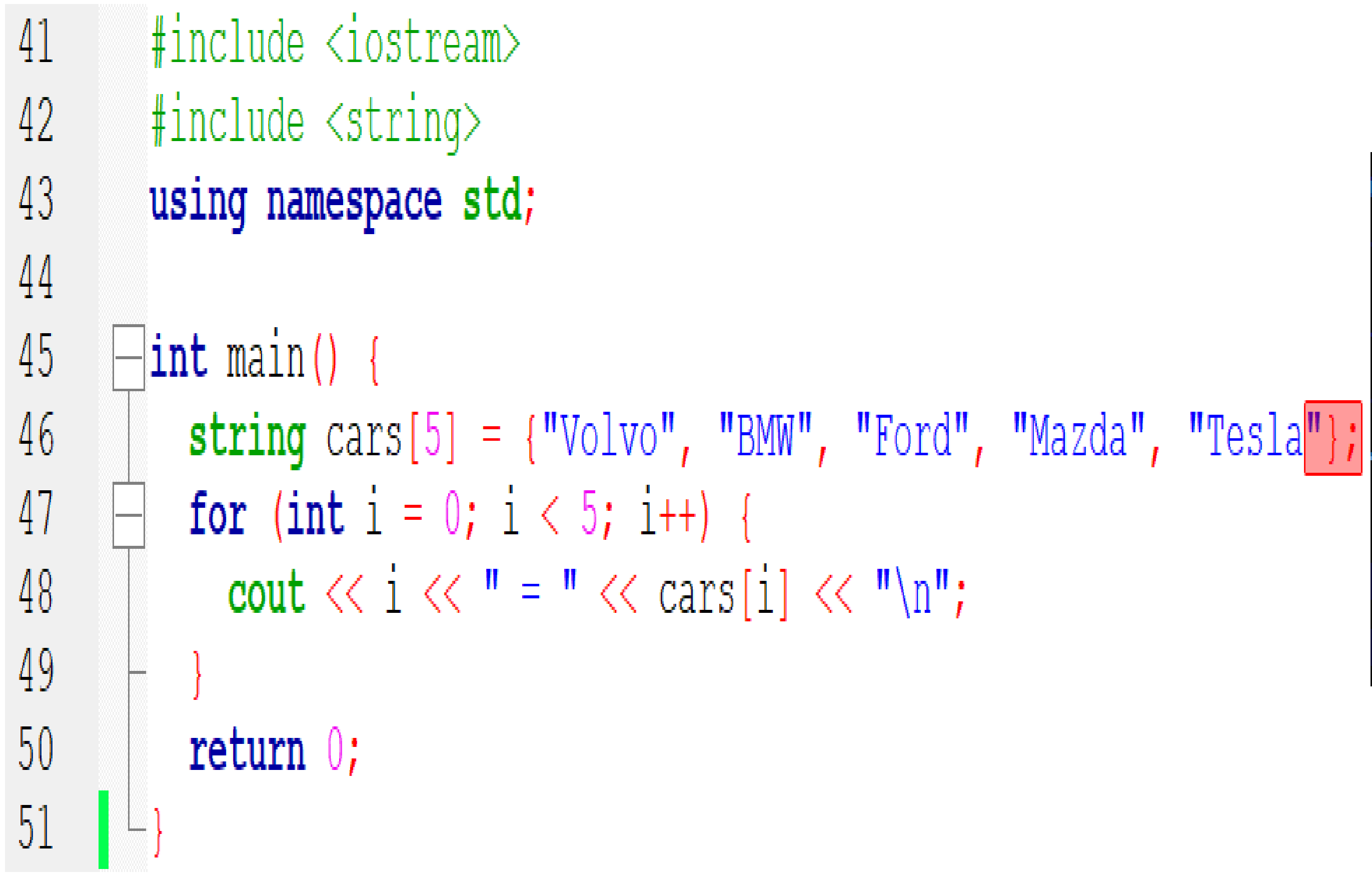


C:\Users\hossam\Desktop\Array\bin\Debug\Array.exe

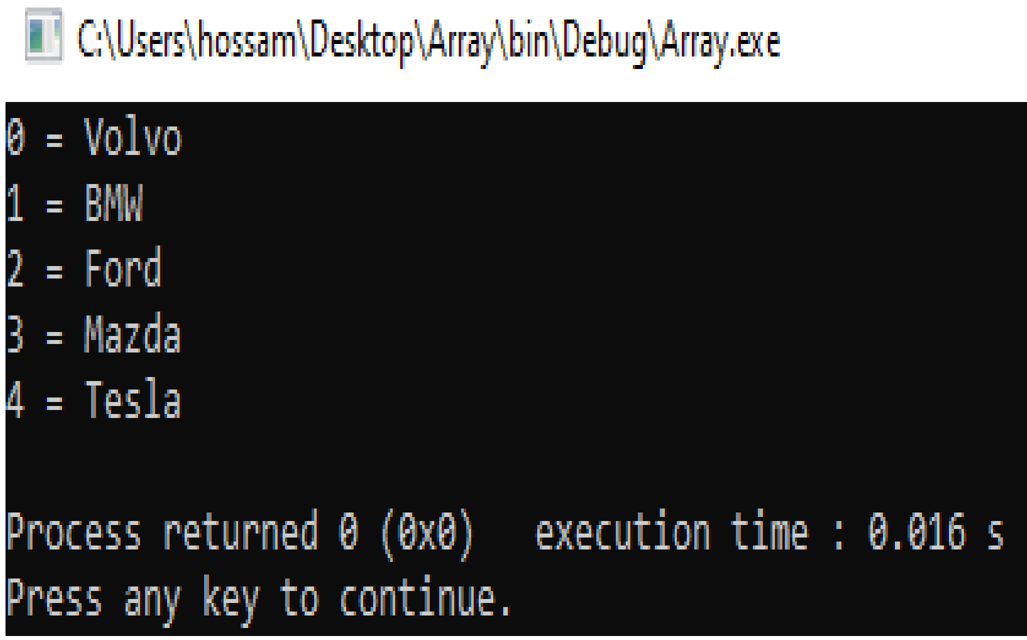
```
Volvo
BMW
Ford
Mazda
Tesla

Process returned 0 (0x0)   execution time : 0.053 s
Press any key to continue.
```

C++ Arrays and Loops

- You can loop through the array elements with the for loop.
- Example : outputs the index of each element together with its value:



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

int main() {
  string cars[5] = {"Volvo", "BMW", "Ford", "Mazda", "Tesla"};
  for (int i = 0; i < 5; i++) {
    cout << i << " = " << cars[i] << "\n";
  }
  return 0;
}
```

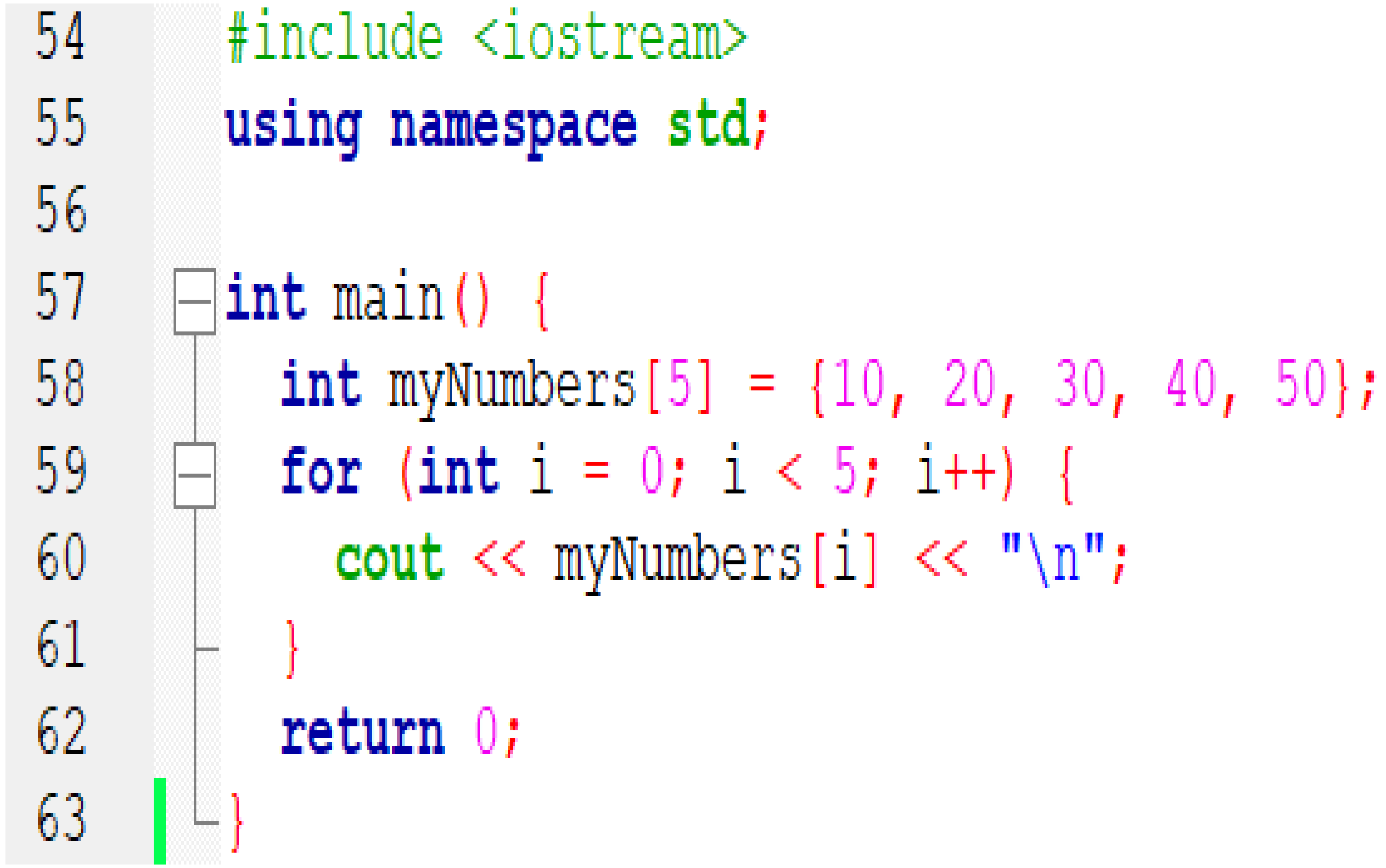


C:\Users\hossam\Desktop\Array\bin\Debug\Array.exe

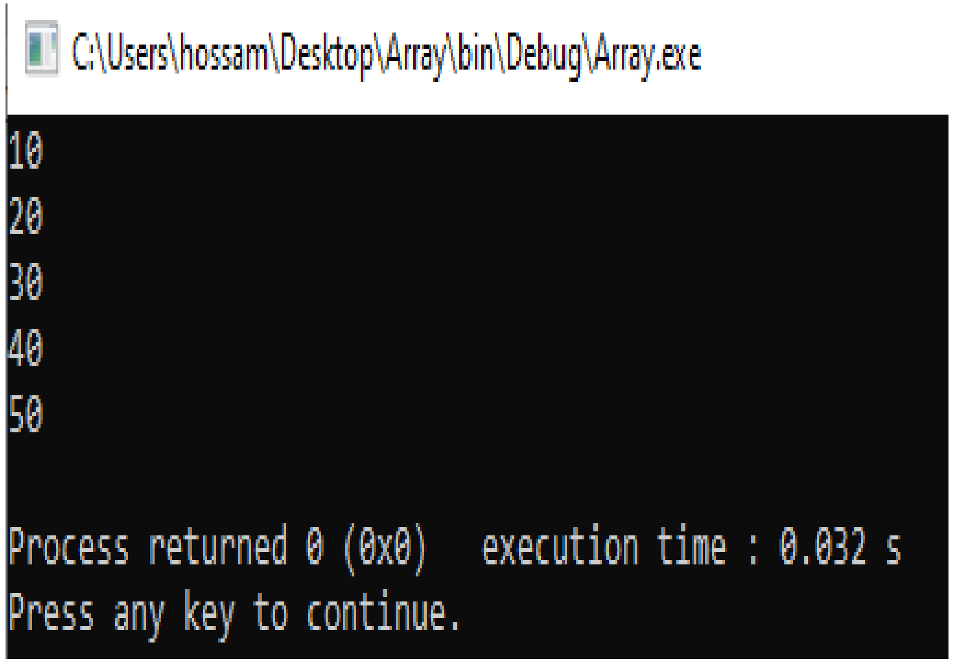
```
0 = Volvo
1 = BMW
2 = Ford
3 = Mazda
4 = Tesla

Process returned 0 (0x0)   execution time : 0.016 s
Press any key to continue.
```

C++ Arrays and Loops

- You can loop through the array elements with the for loop.
- Example : shows how to loop through an array of integers:



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

int main() {
  int myNumbers[5] = {10, 20, 30, 40, 50};
  for (int i = 0; i < 5; i++) {
    cout << myNumbers[i] << "\n";
  }
  return 0;
}
```



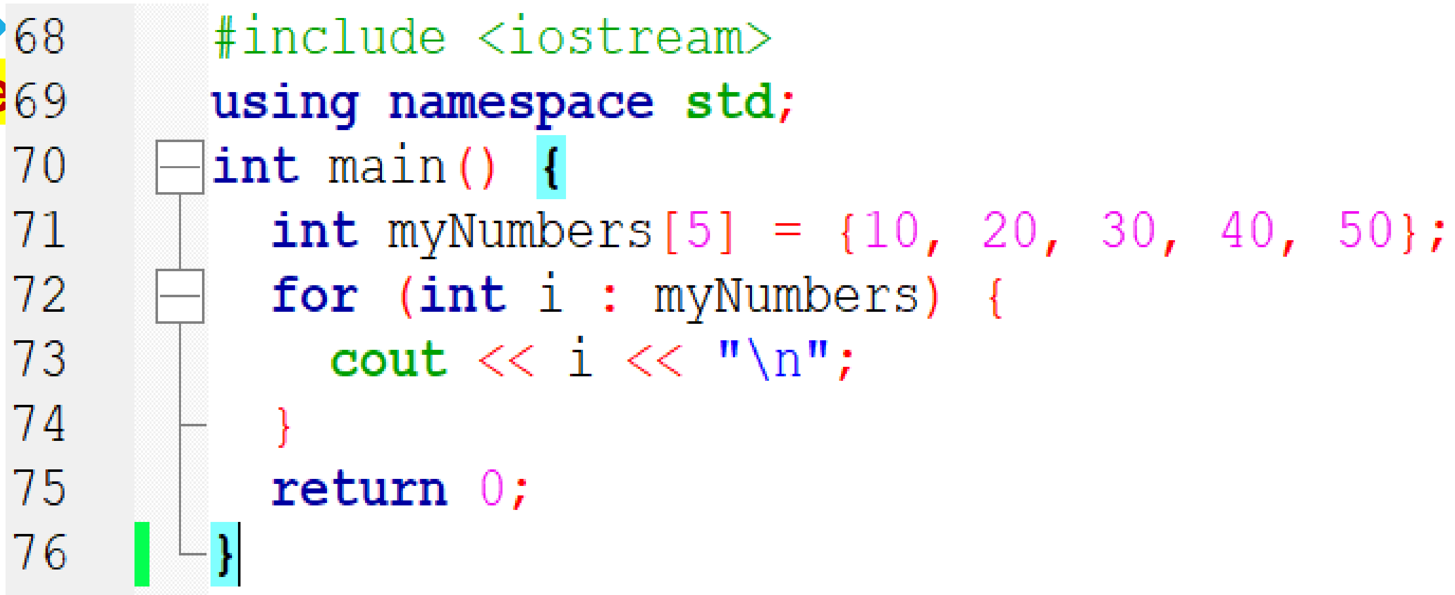
```
C:\Users\hossam\Desktop\Array\bin\Debug\Array.exe
10
20
30
40
50

Process returned 0 (0x0)   execution time : 0.032 s
Press any key to continue.
```

C++ FOREACH LOOP

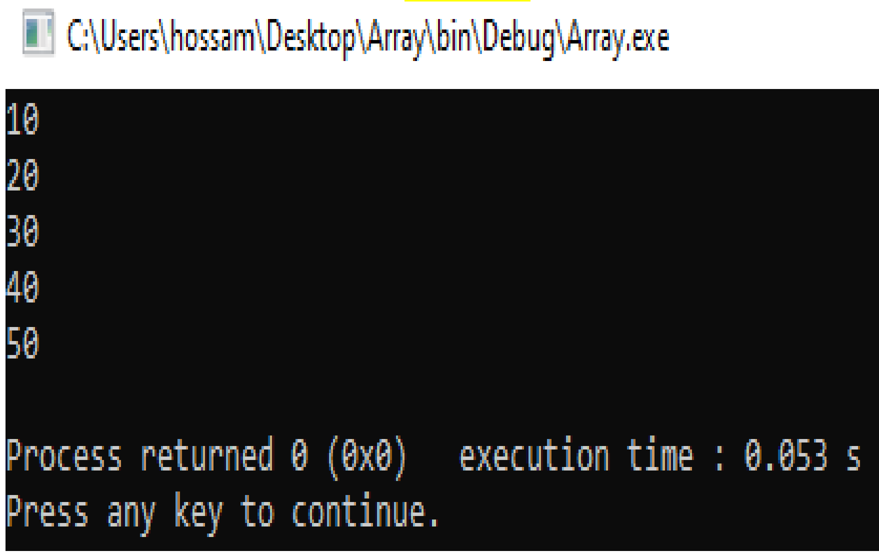
- a "for-each loop" , which is used exclusively to loop through elements in an array:
- Syntax:

```
for (type variableName : arrayName) {
  // code block to be executed
}
```

- e



```cpp
#include <iostream>
using namespace std;
int main() {
   int myNumbers[5] = {10, 20, 30, 40, 50};
   for (int i : myNumbers) {
     cout << i << "\n";
   }
   return 0;
}
```



C:\Users\hossam\Desktop\Array\bin\Debug\Array.exe

```
10
20
30
40
50

Process returned 0 (0x0)   execution time : 0.053 s
Press any key to continue.
```

C++ Omit Array Size

❖In C++, you don't have to specify the size of the array. The compiler is smart enough to determine the size of the array based on the number of inserted values:

➢string cars[] = {"Volvo", "BMW", "Ford"}; // Three arrays

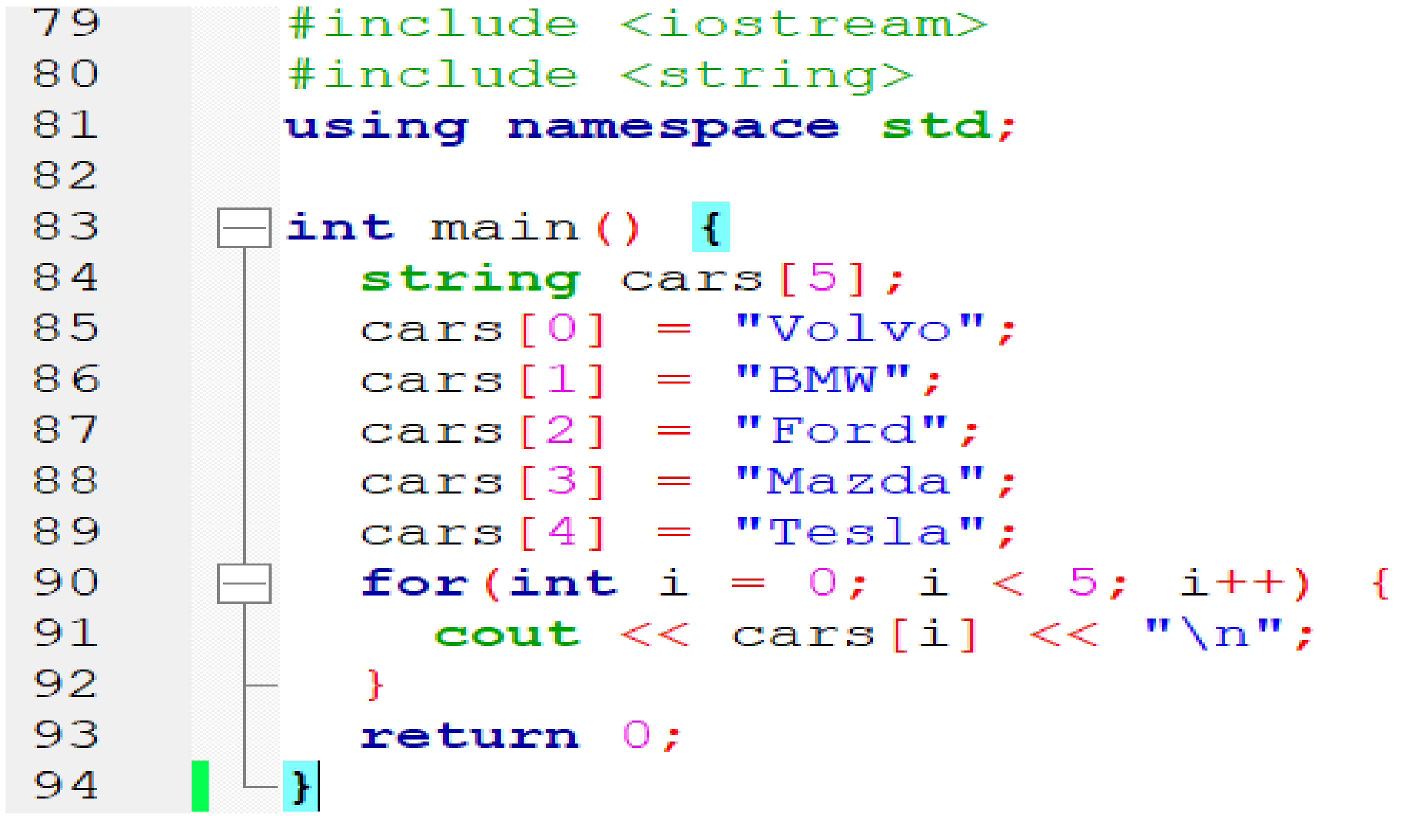
is equal to

➢string cars[3] = {"Volvo", "BMW", "Ford"}; // Also three arrays

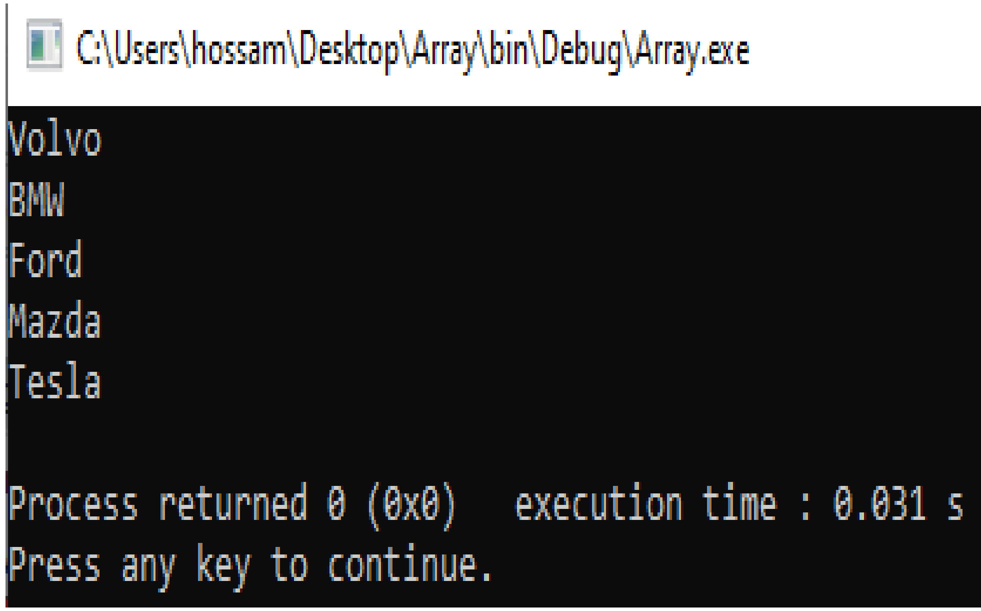
C++ OMIT ARRAY SIZE “CONT”

❖ **Omit Elements on Declaration :**

➢ **It is also possible to declare an array without specifying the elements on declaration, and add them later:**



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

int main() {
   string cars[5];
   cars[0] = "Volvo";
   cars[1] = "BMW";
   cars[2] = "Ford";
   cars[3] = "Mazda";
   cars[4] = "Tesla";
   for(int i = 0; i < 5; i++) {
     cout << cars[i] << "\n";
   }
   return 0;
}
```



C:\Users\hossam\Desktop\Array\bin\Debug\Array.exe

```
Volvo
BMW
Ford
Mazda
Tesla

Process returned 0 (0x0)   execution time : 0.031 s
Press any key to continue.
```

SIZEOF() OPERATOR IN C++

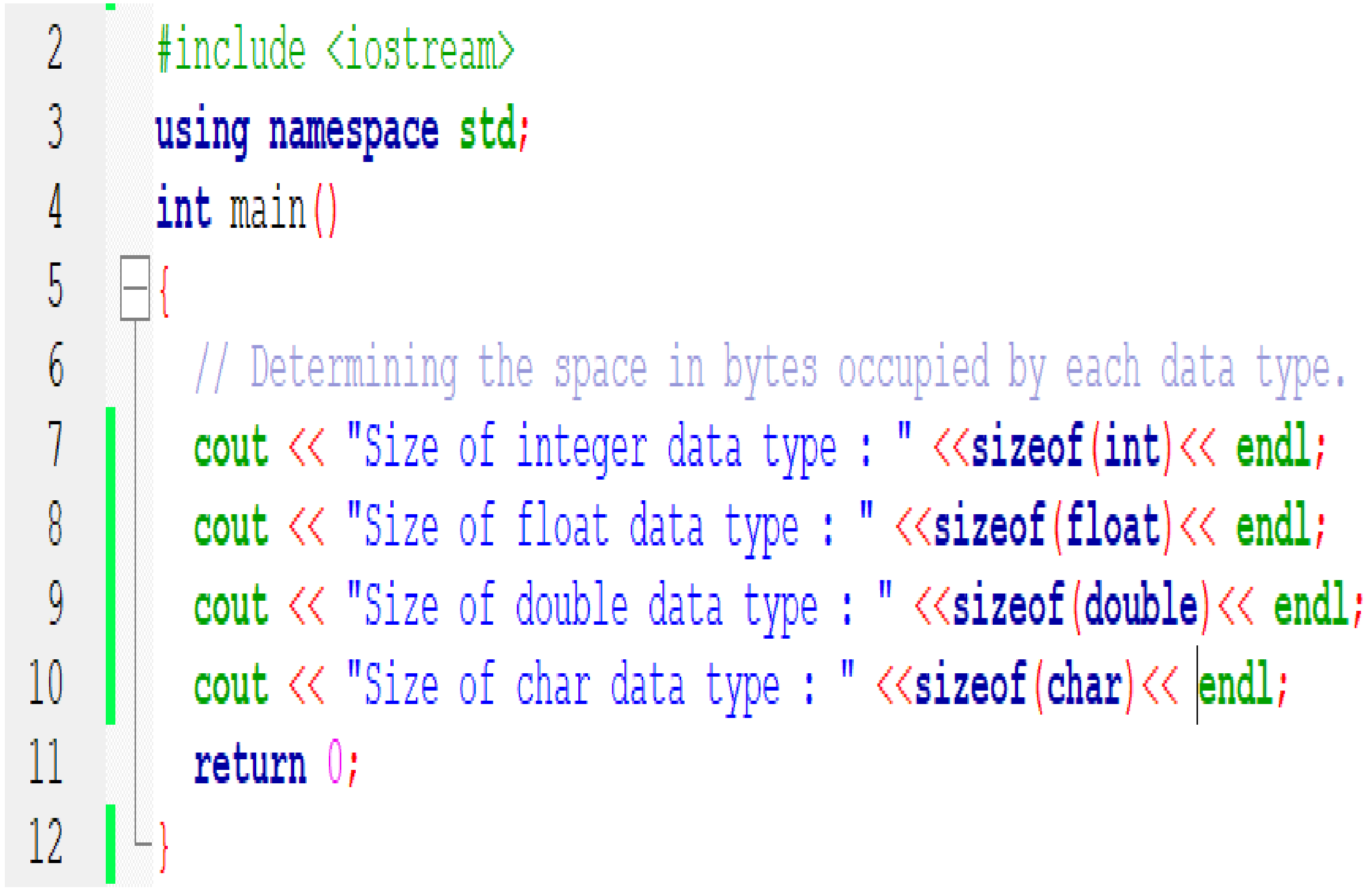
- The sizeof() is an operator that evaluates the size of data type, constants, variable.
- It is a compile-time operator as it returns the size of any variable or a constant at the compilation time.
- the sizeof() operator which is calculated the amount of RAM occupied in the computer.

➢ **Syntax of the sizeof() operator :**

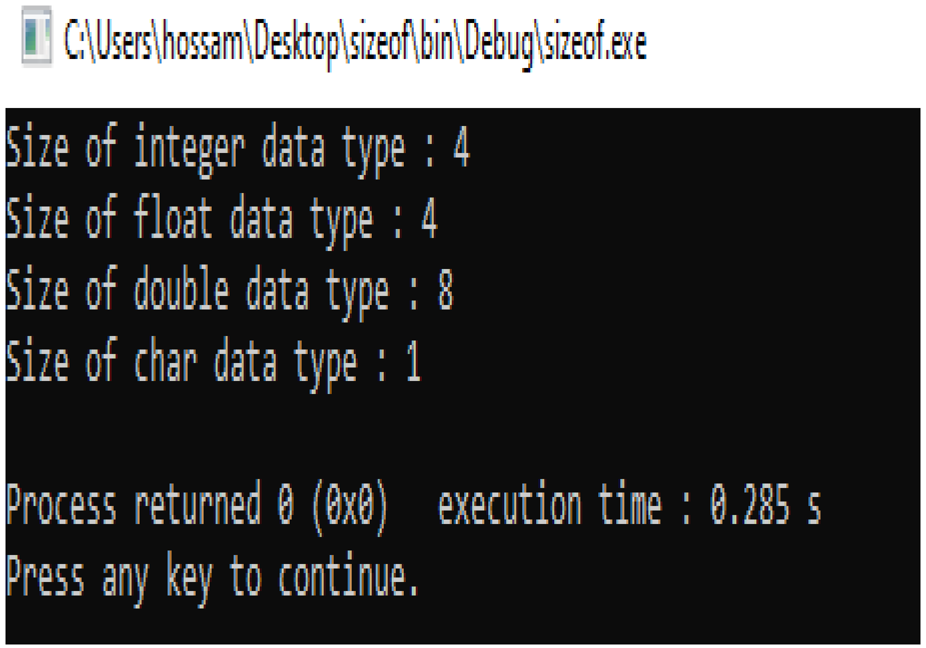
```
sizeof(data_type);
```

SIZEOF() OPERATOR IN C++ "CONT"

- **Example 1_:**



```
#include <iostream>
using namespace std;
int main()
{
   // Determining the space in bytes occupied by each data type.
   cout << "Size of integer data type : " <<sizeof(int)<< endl;
   cout << "Size of float data type : " <<sizeof(float)<< endl;
   cout << "Size of double data type : " <<sizeof(double)<< endl;
   cout << "Size of char data type : " <<sizeof(char)<< endl;
   return 0;
}
```

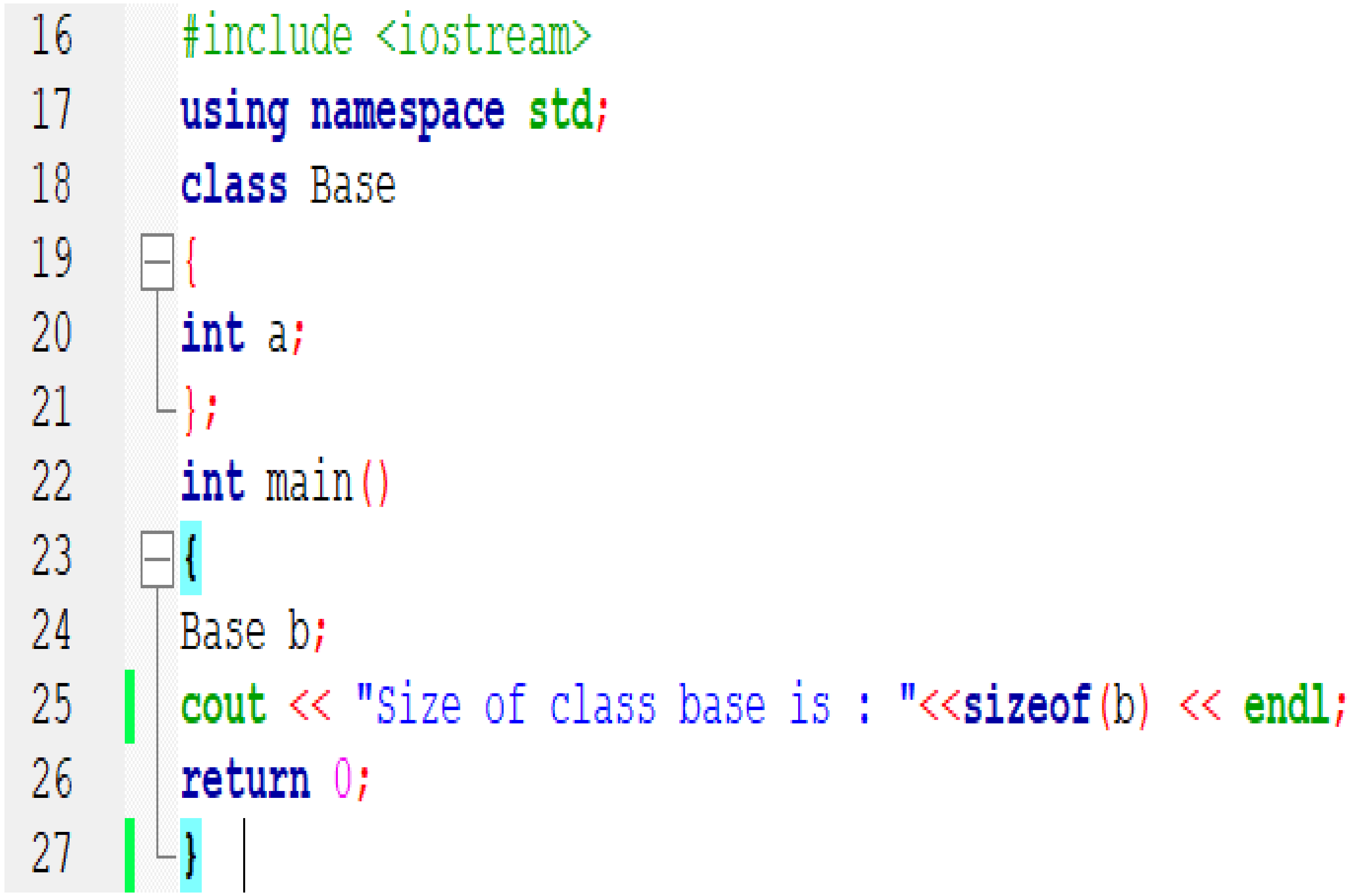


C:\Users\hossam\Desktop\sizeof\bin\Debug\sizeof.exe

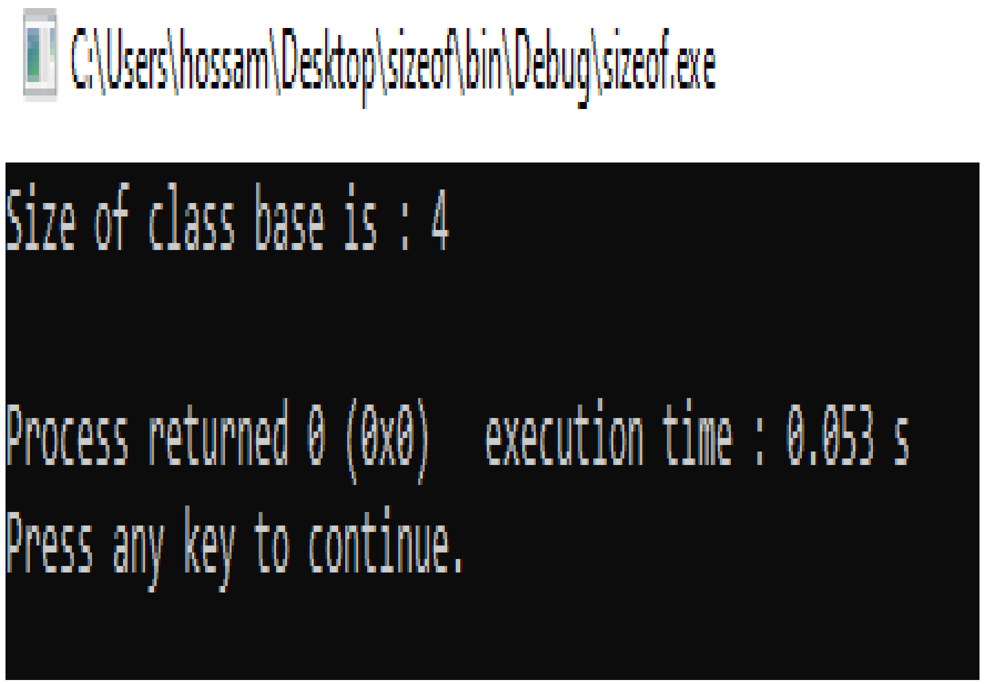
```
Size of integer data type : 4
Size of float data type : 4
Size of double data type : 8
Size of char data type : 1


Process returned 0 (0x0)   execution time : 0.285 s
Press any key to continue.
```

SIZEOF() OPERATOR IN C++ “CONT”

➢ **Example 2_:**



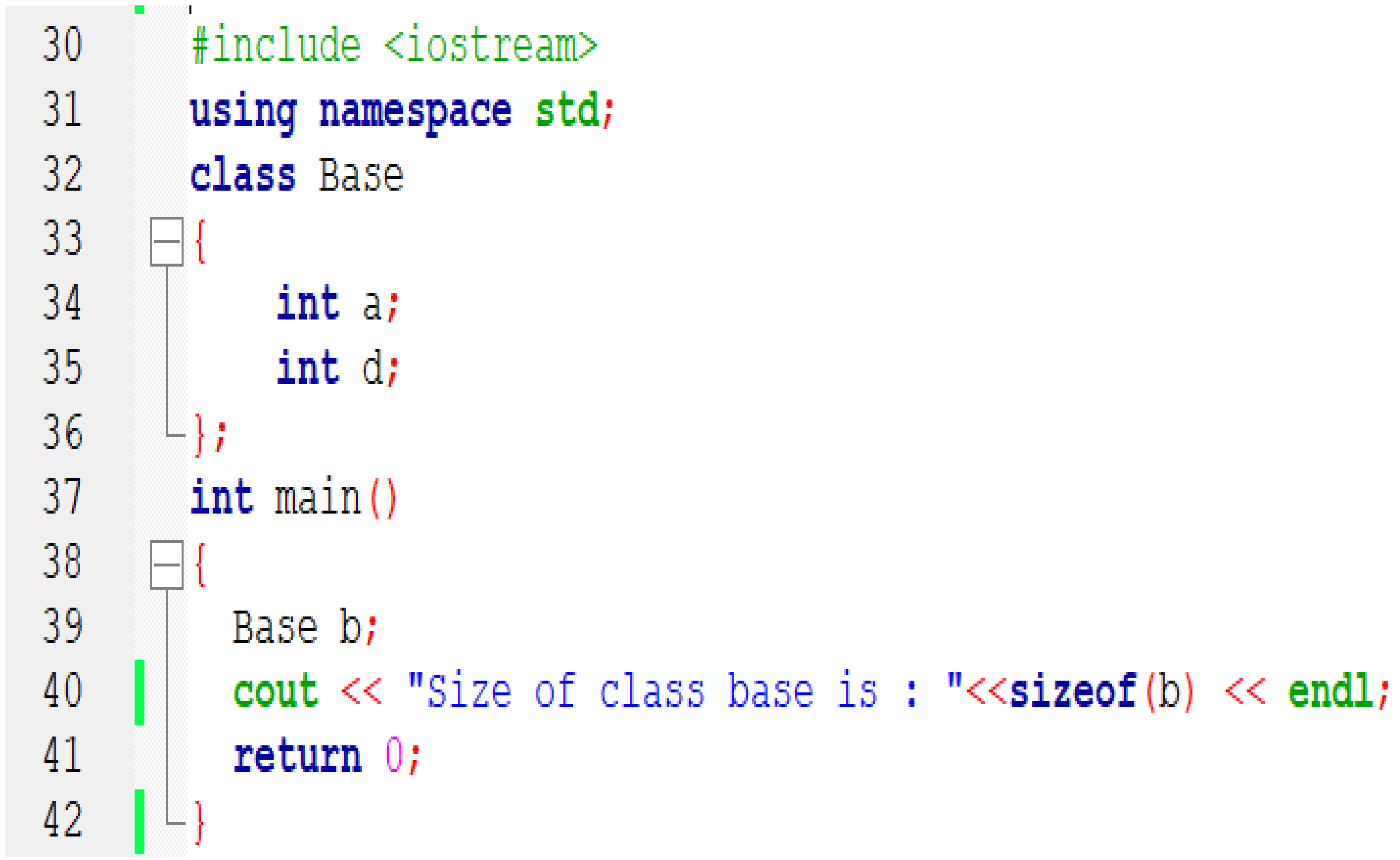
```cpp
#include <iostream>
using namespace std;
class Base
{
int a;
};
int main()
{
Base b;
cout << "Size of class base is : "<<sizeof(b) << endl;
return 0;
}
```



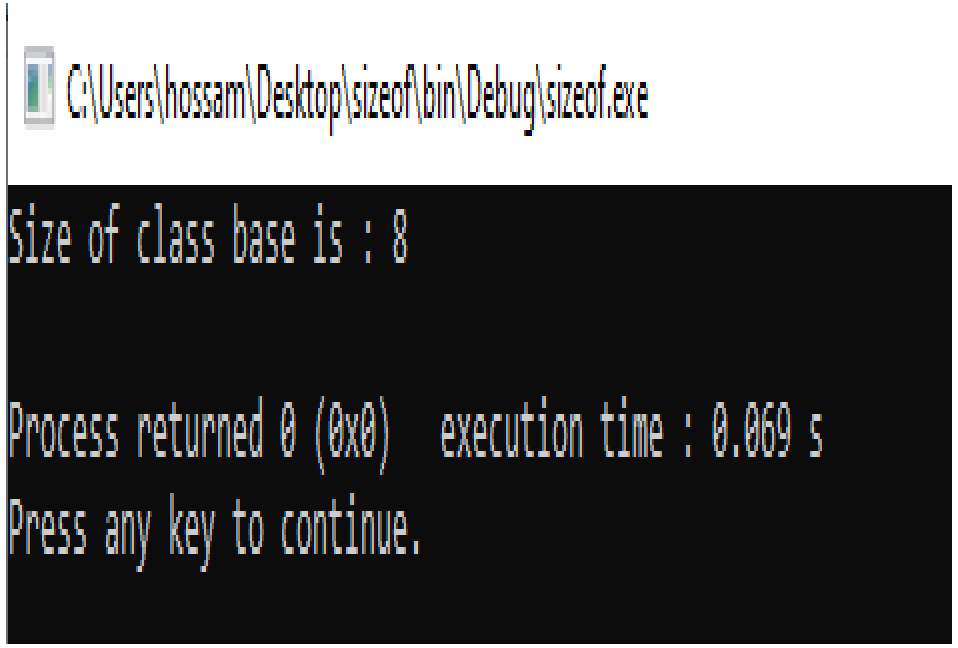
```
C:\Users\hossam\Desktop\sizeof\bin\Debug\sizeof.exe

Size of class base is : 4


Process returned 0 (0x0)   execution time : 0.053 s
Press any key to continue.
```

SIZEOF() OPERATOR IN C++ “CONT”

- **Example 3_:**



```cpp
#include <iostream>
using namespace std;
class Base
{
    int a;
    int d;
};
int main()
{
  Base b;
  cout << "Size of class base is : "<<sizeof(b) << endl;
  return 0;
}
```



```
C:\Users\hossam\Desktop\sizeof\bin\Debug\sizeof.exe

Size of class base is : 8


Process returned 0 (0x0)   execution time : 0.069 s
Press any key to continue.
```

C++ ARRAY SIZE

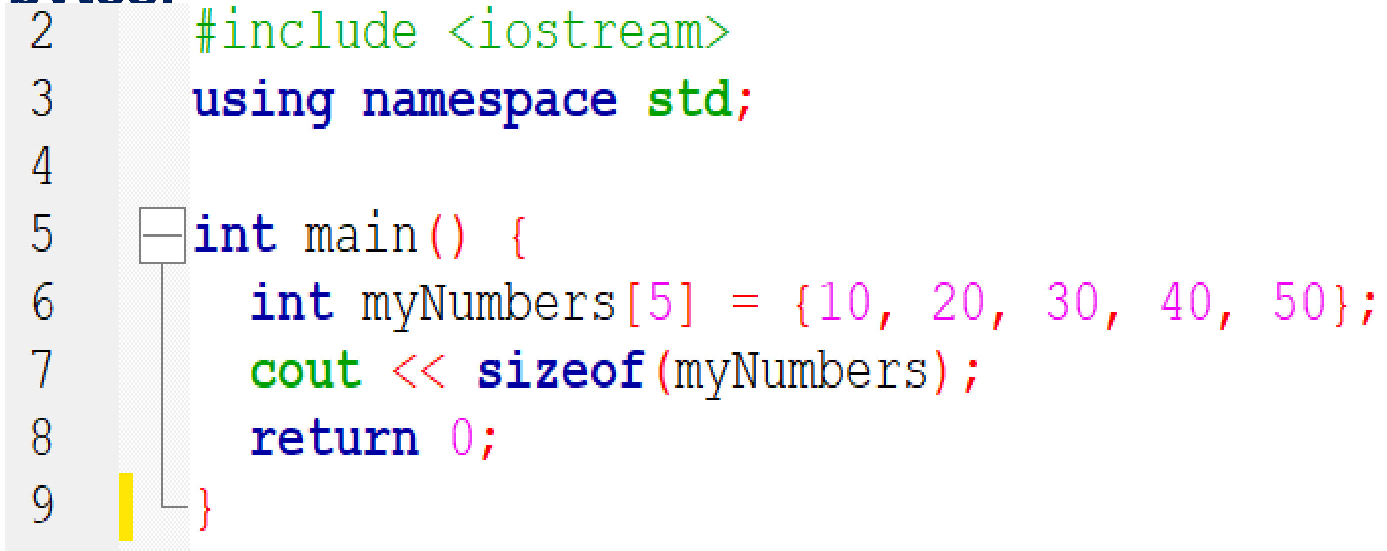
- In C++, To get the size of an array, you can use the sizeof() operator:

- **Example :**

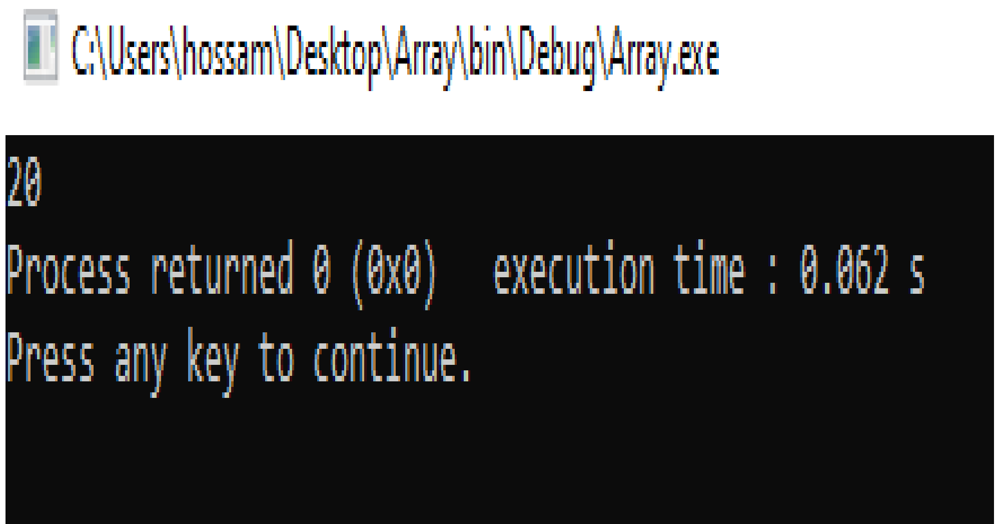
  Why did the result show 20 instead of 5, when the array contains 5 elements?

- It is because the sizeof() operator returns the size of a type in bytes.

- an int type is usually 4 bytes, so from the example above, (4 bytes x 5 elements) = 20 bytes.



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

int main() {
   int myNumbers[5] = {10, 20, 30, 40, 50};
   cout << sizeof(myNumbers);
   return 0;
}
```

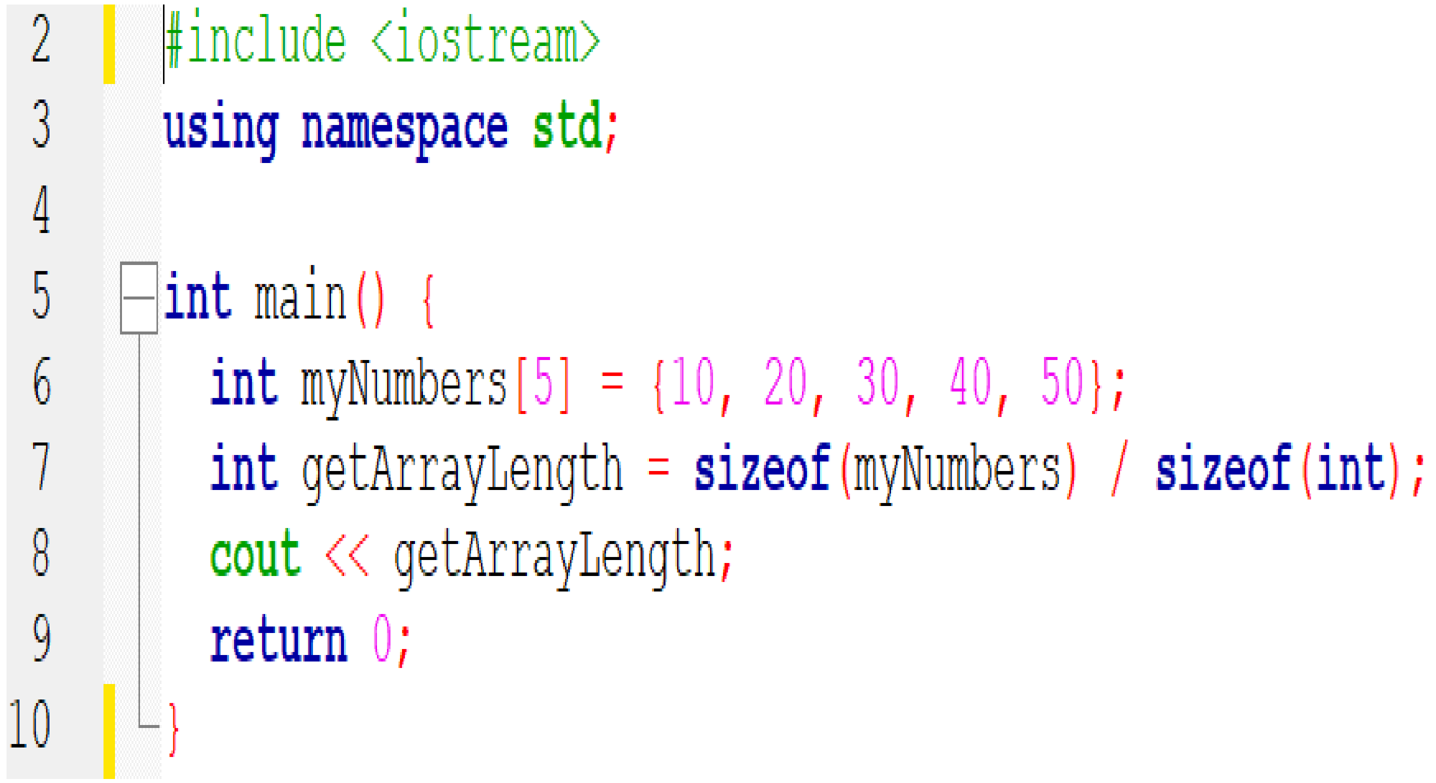


```
C:\Users\hossam\Desktop\Array\bin\Debug\Array.exe

20
Process returned 0 (0x0)   execution time : 0.062 s
Press any key to continue.
```

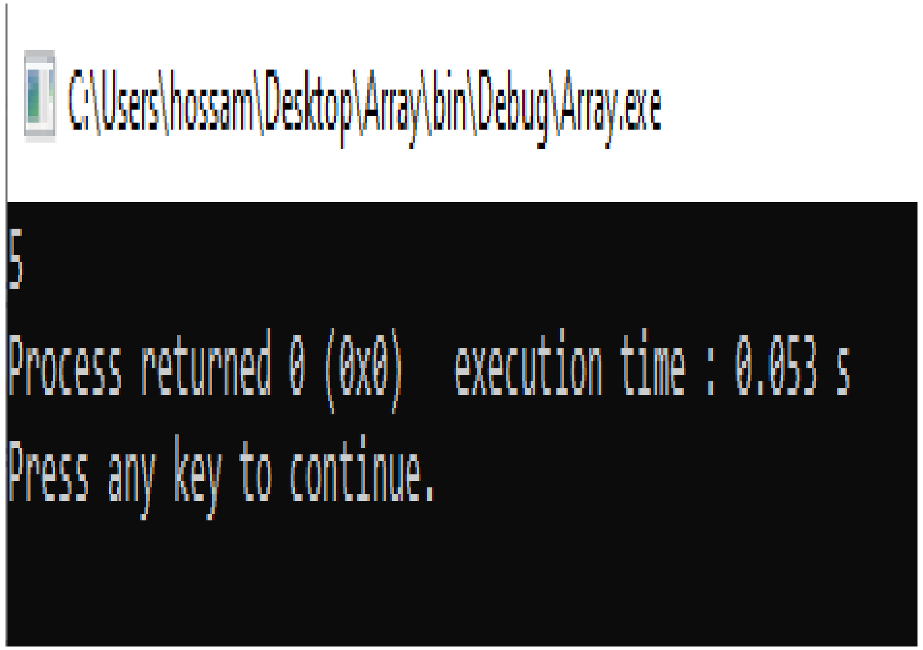
C++ Array Size "Cont"

❖To find out how many elements an array has, you have to divide the size of the array by the size of the data type it contains:

❖Example :



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

int main() {
  int myNumbers[5] = {10, 20, 30, 40, 50};
  int getArrayLength = sizeof(myNumbers) / sizeof(int);
  cout << getArrayLength;
  return 0;
}
```



```
C:\Users\hossam\Desktop\Array\bin\Debug\Array.exe
5
Process returned 0 (0x0)   execution time : 0.053 s
Press any key to continue.
```

C++ Array Size "Cont"

❖ **Loop Through an Array with sizeof()**

➢ **Example :**

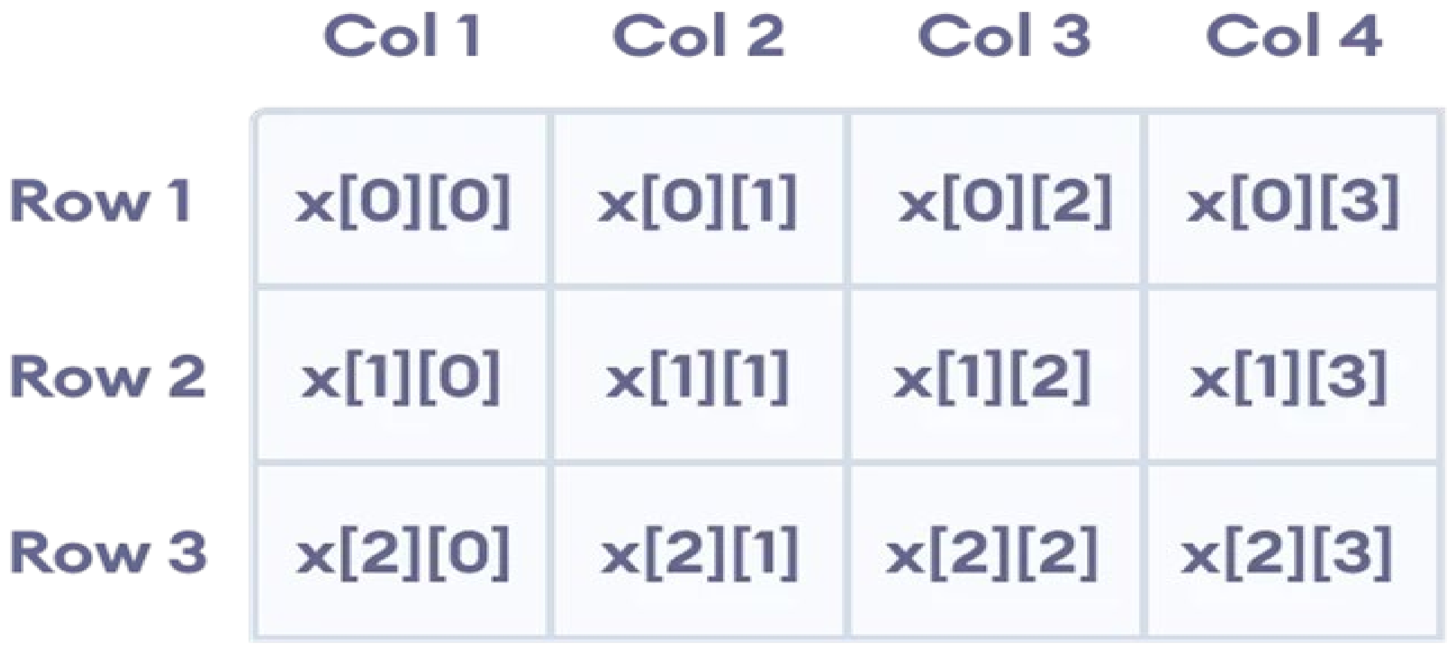
```
int myNumbers[5] = {10, 20, 30, 40, 50};
for (int i = 0; i < 5; i++) {
  cout << myNumbers[i] << "\n";
}
```

It is better

```
int myNumbers[5] = {10, 20, 30, 40, 50};
for (int i = 0; i < sizeof(myNumbers) / sizeof(int); i++)
{
  cout << myNumbers[i] << "\n";
}
```

C++ MULTI-DIMENSIONAL ARRAYS

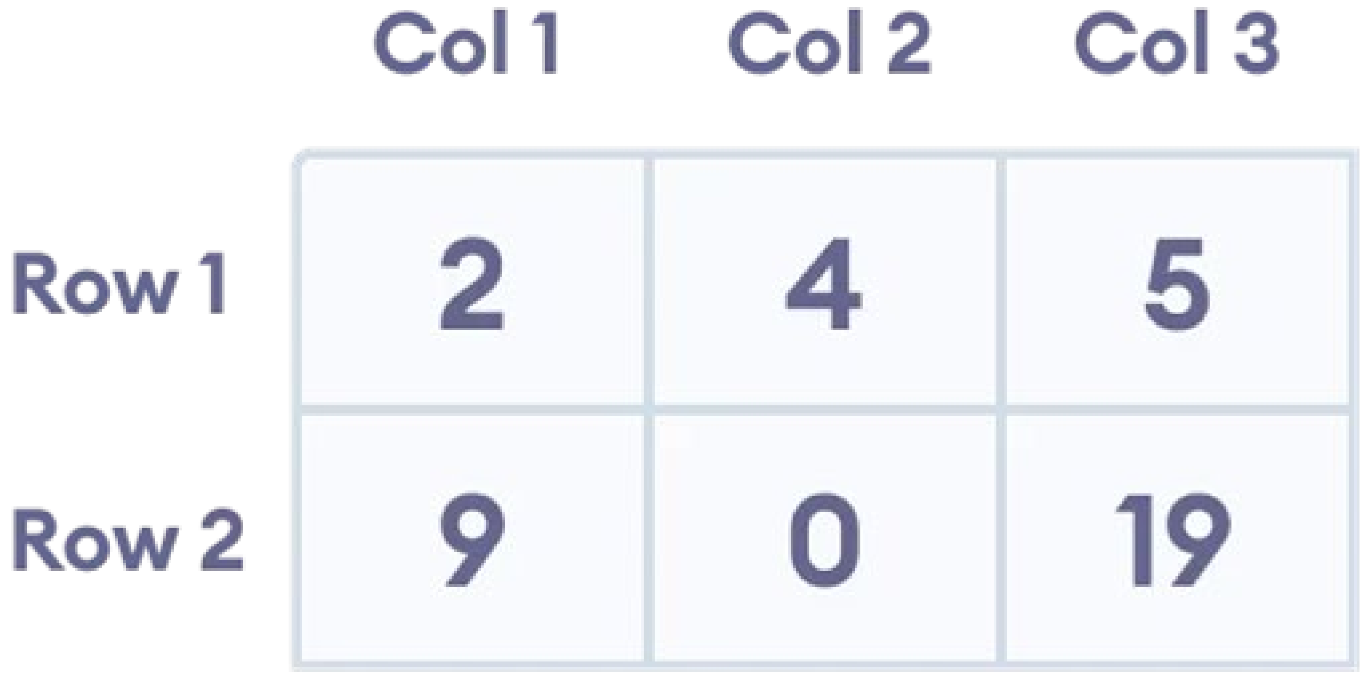
- A multi-dimensional array is an array of arrays.
- example: int x[3][4];
- x is a two-dimensional array. It can hold a maximum of 12 elements.



	Col 1	Col 2	Col 3	Col 4
Row 1	x[0][0]	x[0][1]	x[0][2]	x[0][3]
Row 2	x[1][0]	x[1][1]	x[1][2]	x[1][3]
Row 3	x[2][0]	x[2][1]	x[2][2]	x[2][3]

C++ Multi-Dimensional Arrays “Cont”

- A multi-dimensional array is an array of arrays.
- example: int test[2][3] = { {2, 4, 5}, {9, 0, 19}};
- x is a two-dimensional array. This array has 2 rows and 3 columns.



	Col 1	Col 2	Col 3
Row 1	2	4	5
Row 2	9	0	19

C++ Multi-Dimensional Arrays "Cont"

- Arrays can have any number of dimensions.
- array has three dimensions:

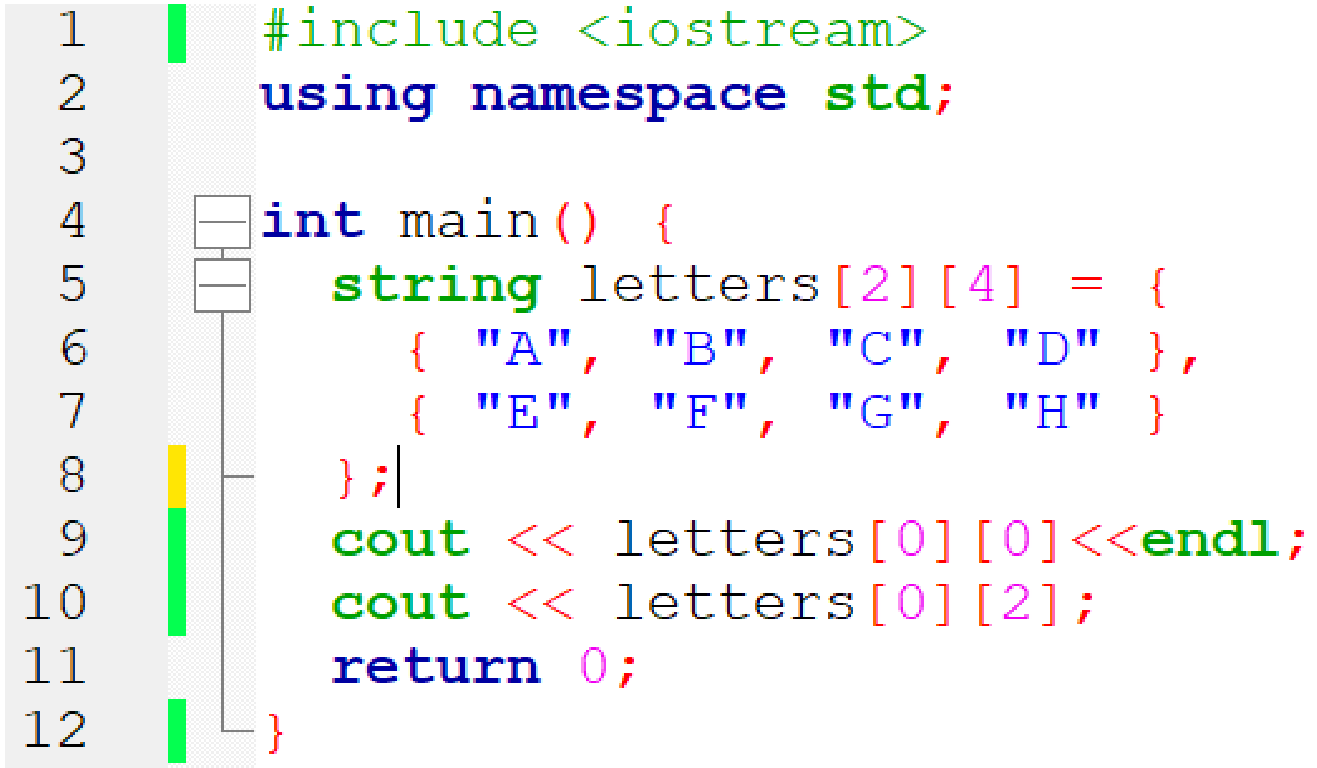
```
string letters[2][2][2] = {
  { { "A", "B" },{ "C", "D" } },
  { { "E", "F" }, { "G", "H" } }
};
```

C++ Multi-Dimensional Arrays “Cont”

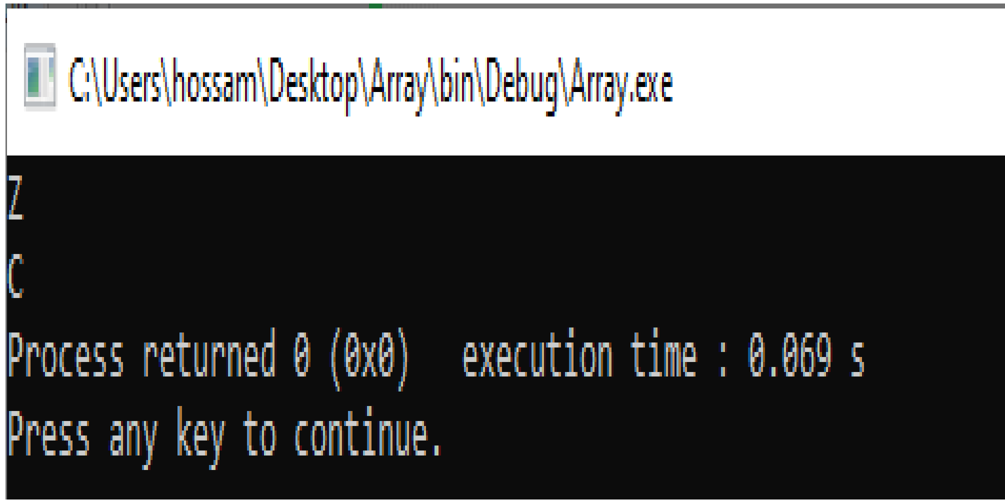
❖**Access the Elements of a Multi-Dimensional Array.**

➢**To access an element of a multi-dimensional array, specify an index number in each of the array's dimensions.**

➢**Example :**



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

int main()  {
   string letters[2][4] = {
     { "A", "B", "C", "D" },
     { "E", "F", "G", "H" }
   };
   cout << letters[0][0]<<endl;
   cout << letters[0][2];
   return 0;
}
```



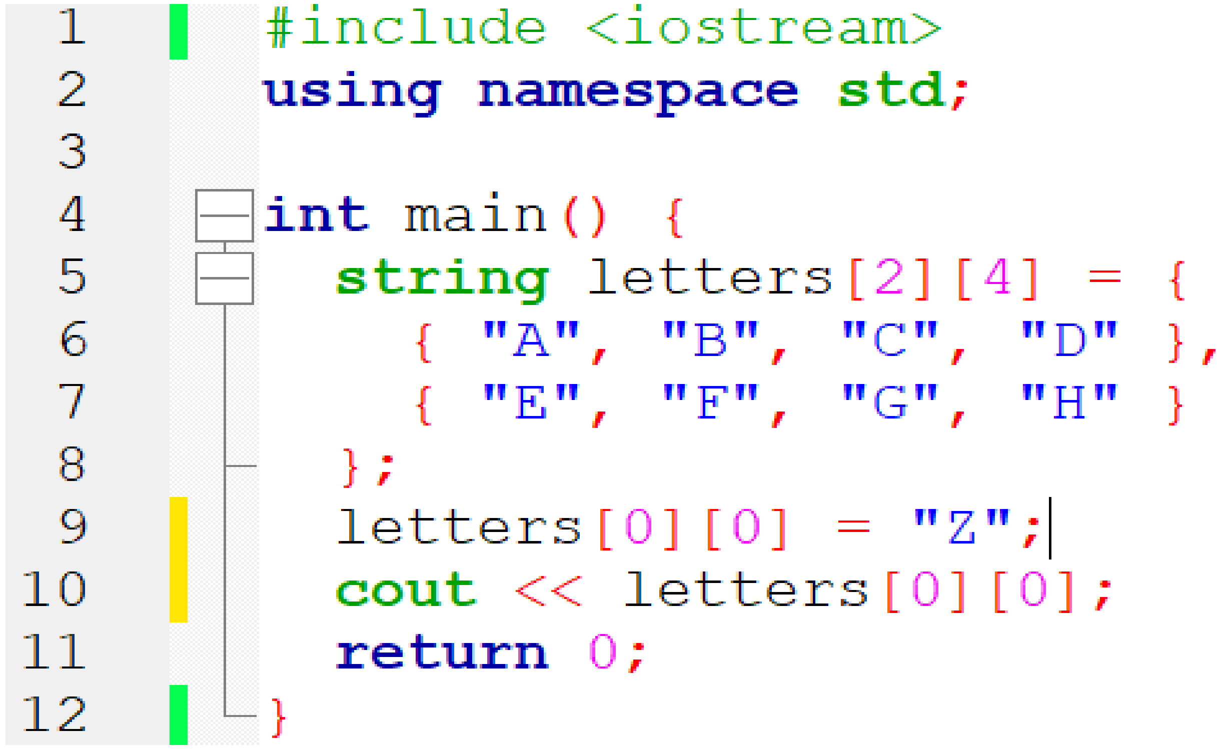
```
C:\Users\hossam\Desktop\Array\bin\Debug\Array.exe
A
C
Process returned 0 (0x0)   execution time : 0.069 s
Press any key to continue.
```

C++ MULTI-DIMENSIONAL ARRAYS “CONT”

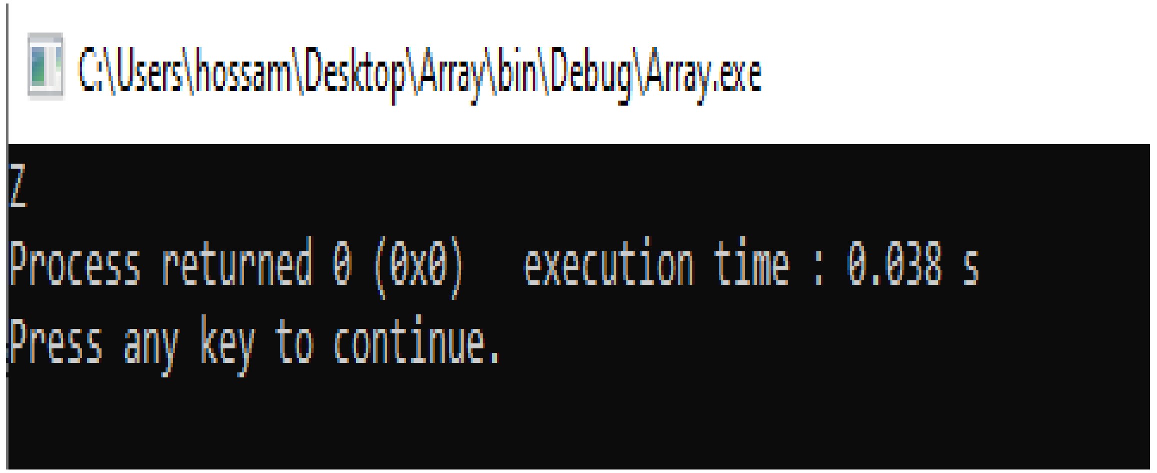
❖ Change Elements in a Multi-Dimensional Array .

➢ To change the value of an element, refer to the index number of the element in each of the dimensions:

➢ Example :



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

int main() {
  string letters[2][4] = {
    { "A", "B", "C", "D" },
    { "E", "F", "G", "H" }
  };
  letters[0][0] = "Z";
  cout << letters[0][0];
  return 0;
}
```



C:\Users\hossam\Desktop\Array\bin\Debug\Array.exe

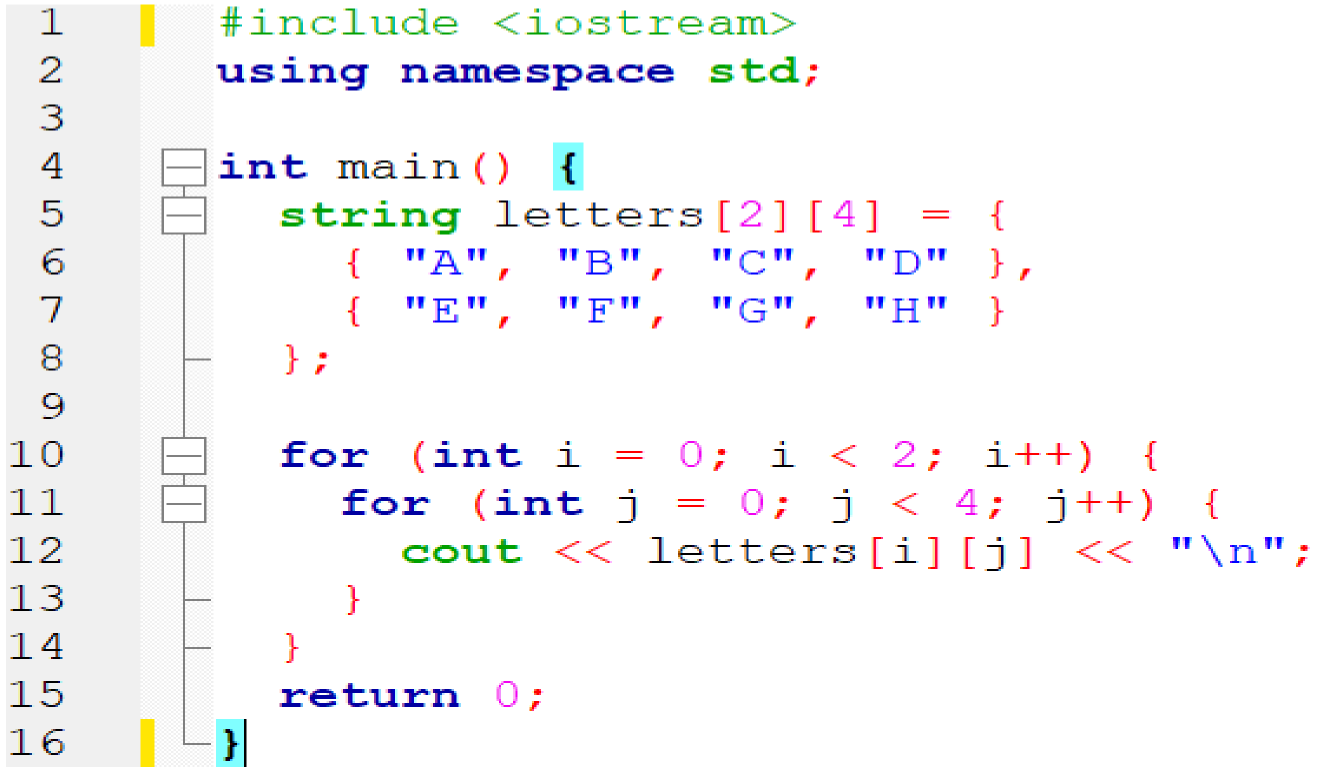
```
Z
Process returned 0 (0x0)   execution time : 0.038 s
Press any key to continue.
```

C++ MULTI-DIMENSIONAL ARRAYS "CONT"

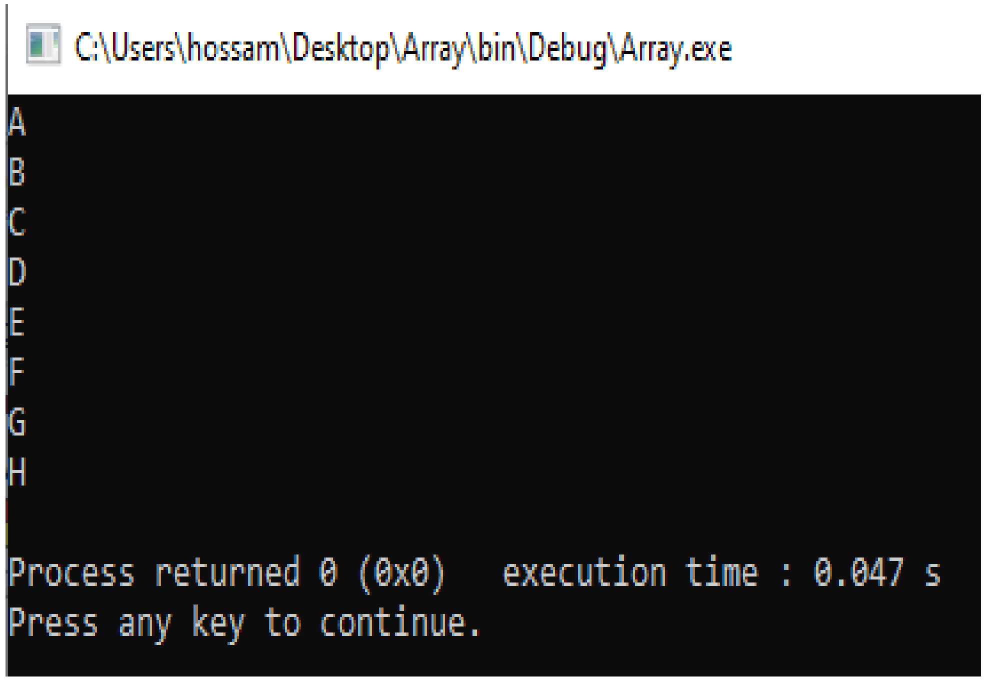
❖ **Loop Through a Multi-Dimensional Array .**

➢ **To loop through a multi-dimensional array, you need one loop for each of the array's dimensions.**

➢ **Example :**



```
#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;
}
```



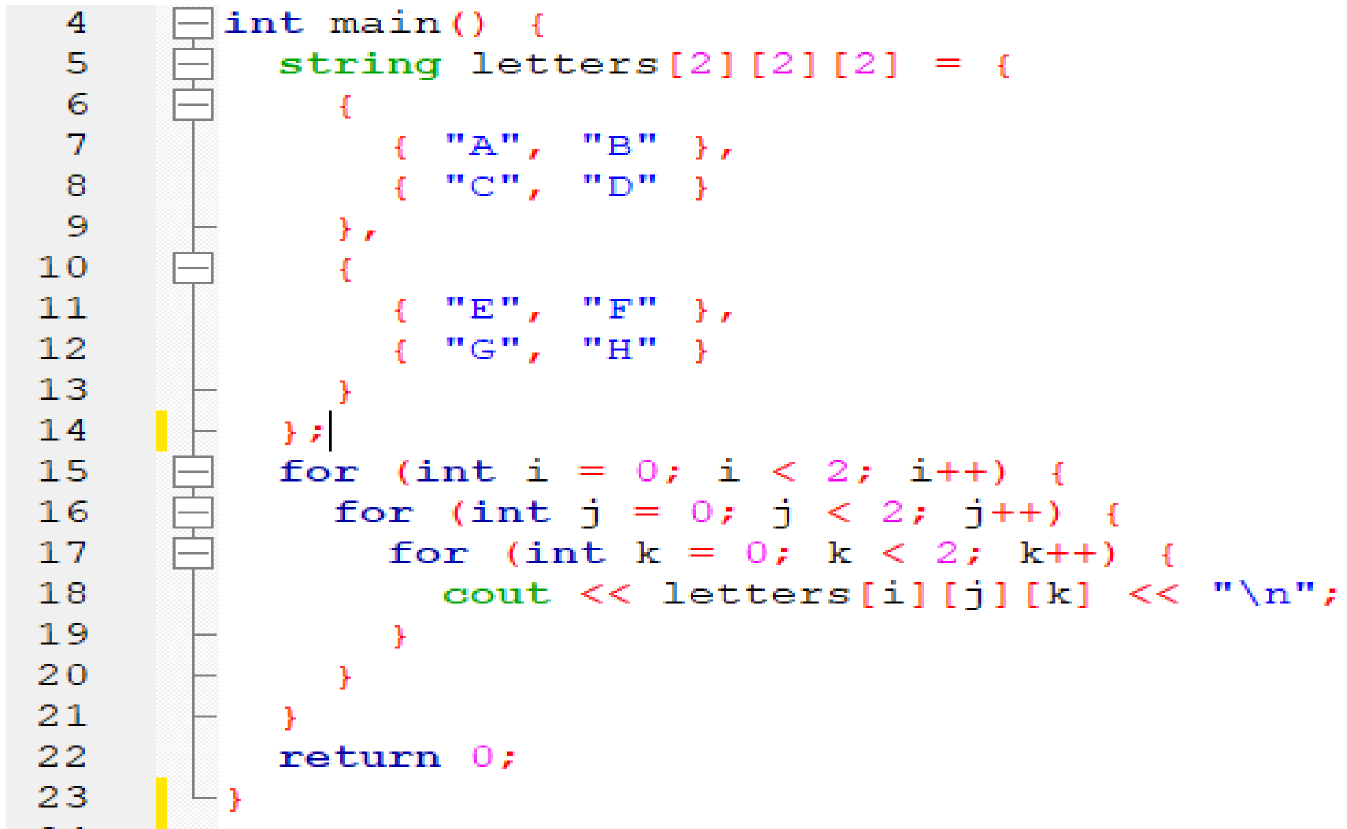
```
C:\Users\hossam\Desktop\Array\bin\Debug\Array.exe
A
B
C
D
E
F
G
H

Process returned 0 (0x0)   execution time : 0.047 s
Press any key to continue.
```

C++ Multi-Dimensional Arrays “cont”

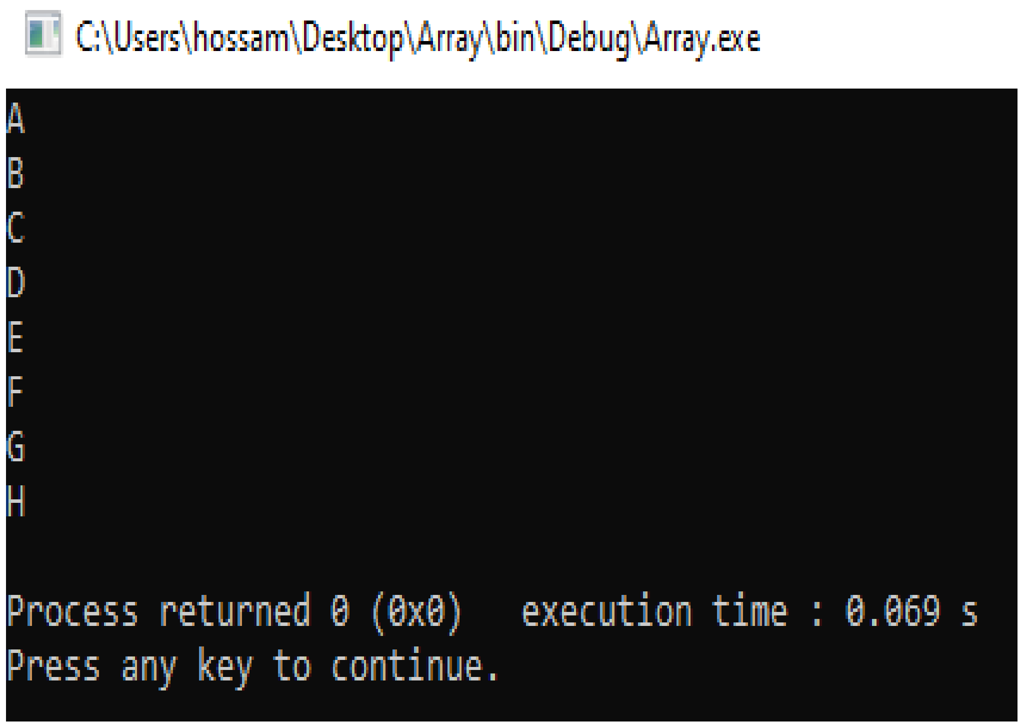
❖ **Loop Through a Multi-Dimensional Array .**

➢ **shows how to loop through a three-dimensional array .**

➢ **Example :**



```cpp
int main()  {
   string letters[2][2][2] = {
      {
         { "A",  "B" },
         { "C",  "D" }
      },
      {
         { "E",  "F" },
         { "G",  "H" }
      }
   };
   for (int i = 0; i < 2; i++) {
      for (int j = 0; j < 2; j++) {
         for (int k = 0; k < 2; k++) {
            cout << letters[i][j][k] << "\n";
         }
      }
   }
   return 0;
}
```



C:\Users\hossam\Desktop\Array\bin\Debug\Array.exe

```
A
B
C
D
E
F
G
H

Process returned 0 (0x0)   execution time : 0.069 s
Press any key to continue.
```

THANKS

Dr/Ghada Maher
Eng/ Hossam Medhat