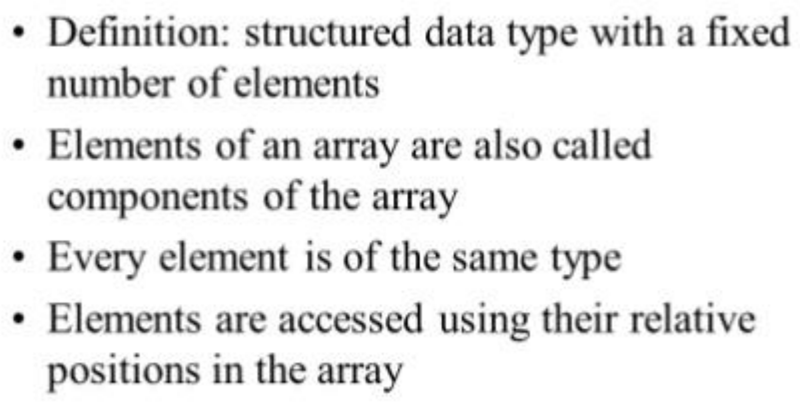
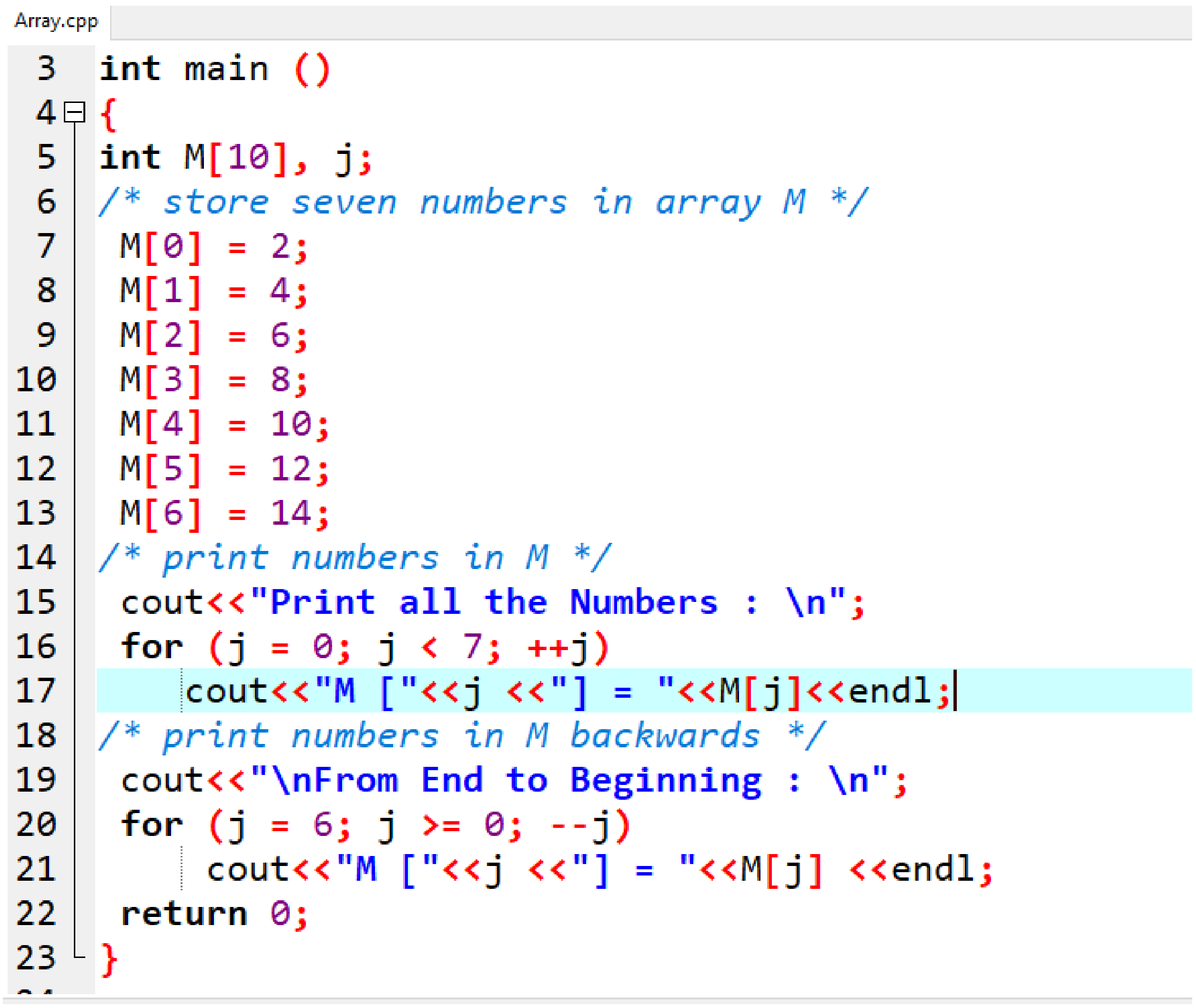
Arrays:



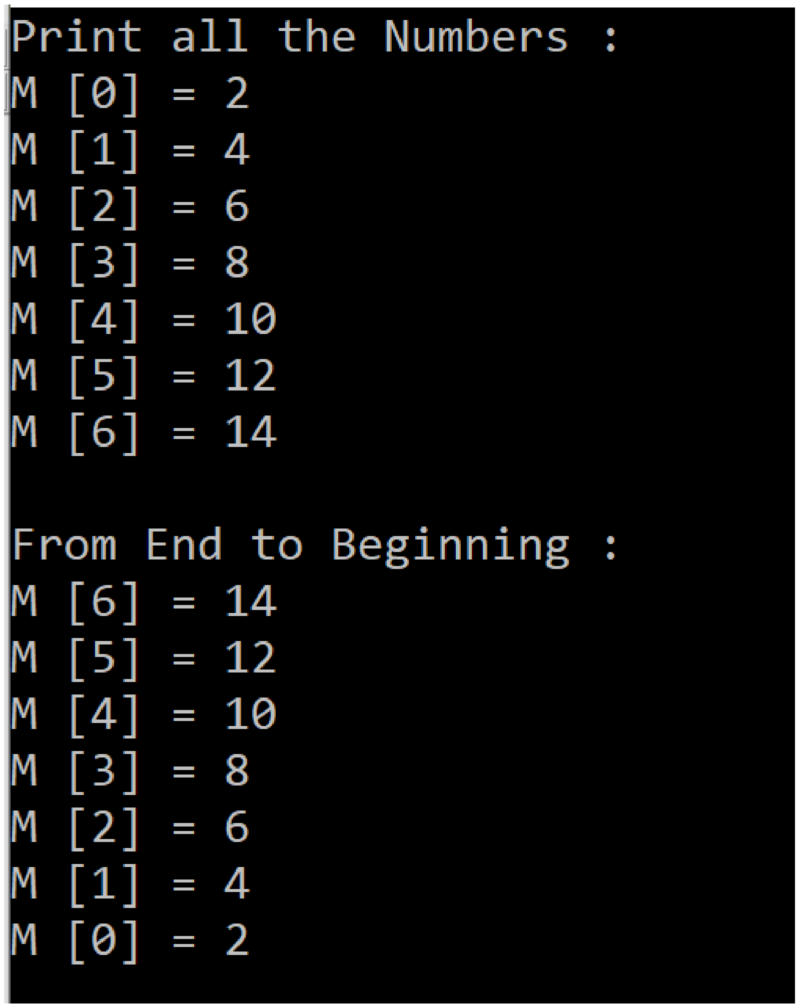
- Definition: structured data type with a fixed number of elements
- Elements of an array are also called components of the array
- Every element is of the same type
- Elements are accessed using their relative positions in the array

```
int A[10];            // An array of ten integers; A[0], A[1], …, A[9]
char name[20];            // An array of 20 characters
float nums[50]; // An array of fifty floating numbers; nums[0],
nums[1], …,nums[49]
int C[]; // An array of an unknown number of integers; C[0], C[1], …,
C[size-1]
int table[5][10];        // A two dimensional array of integers
```

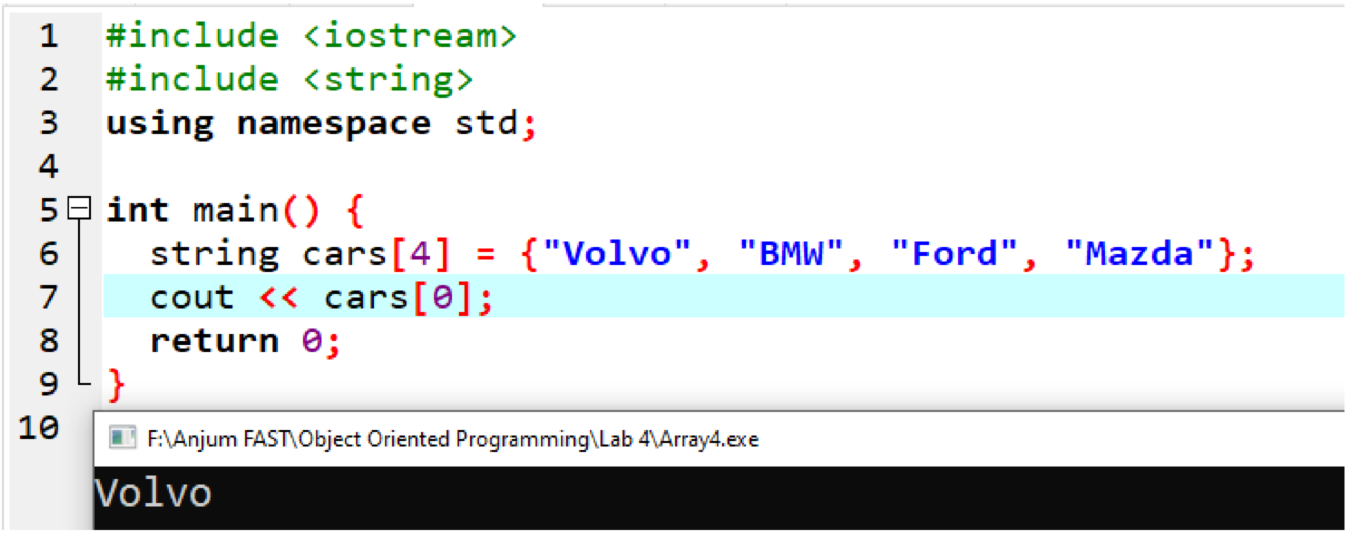
Task -01 a:


Array.cpp

```
int main ()
{
int M[10], j;
/* store seven numbers in array M */
 M[0] = 2;
 M[1] = 4;
 M[2] = 6;
 M[3] = 8;
 M[4] = 10;
 M[5] = 12;
 M[6] = 14;
/* print numbers in M */
 cout<<"Print all the Numbers : \n";
 for (j = 0; j < 7; ++j)
    cout<<"M ["<<j <<"] = "<<M[j]<<endl;
/* print numbers in M backwards */
 cout<<"\nFrom End to Beginning : \n";
 for (j = 6; j >= 0; --j)
     cout<<"M ["<<j <<"] = "<<M[j] <<endl;
 return 0;
}
```


```
Print all the Numbers :
M [0] = 2
M [1] = 4
M [2] = 6
M [3] = 8
M [4] = 10
M [5] = 12
M [6] = 14

From End to Beginning :
M [6] = 14
M [5] = 12
M [4] = 10
M [3] = 8
M [2] = 6
M [1] = 4
M [0] = 2
```

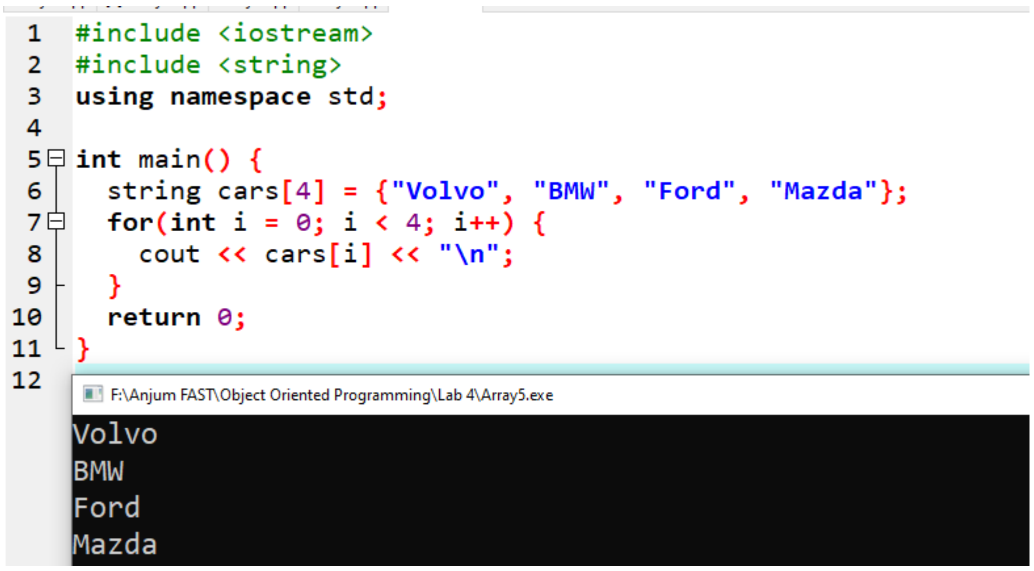
Task -01 b:


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

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

F:\Anjum FAST\Object Oriented Programming\Lab 4\Array4.exe

```
Volvo
```

Task -02:

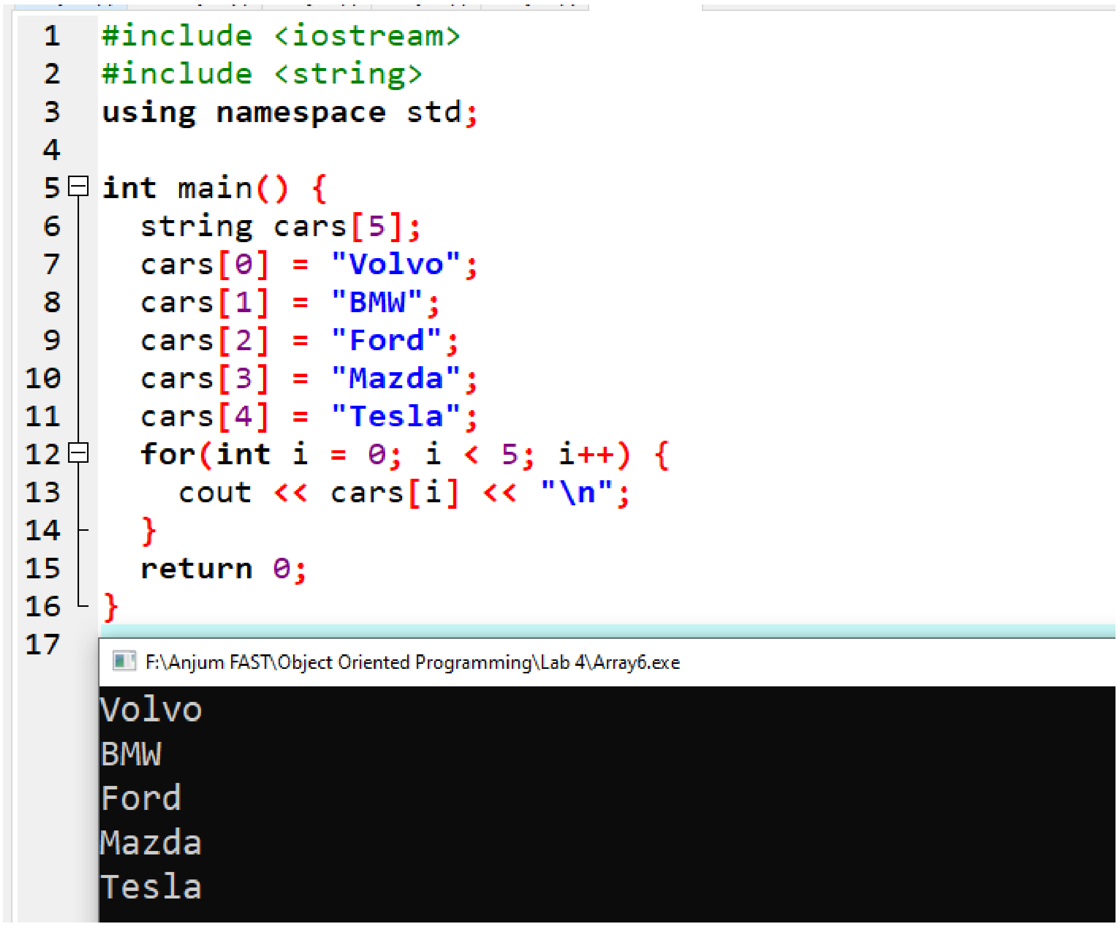


```
#include <iostream>
#include <string>
using namespace std;

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

F:\Anjum FAST\Object Oriented Programming\Lab 4\Array5.exe

```
Volvo
BMW
Ford
Mazda
```

Task -03:

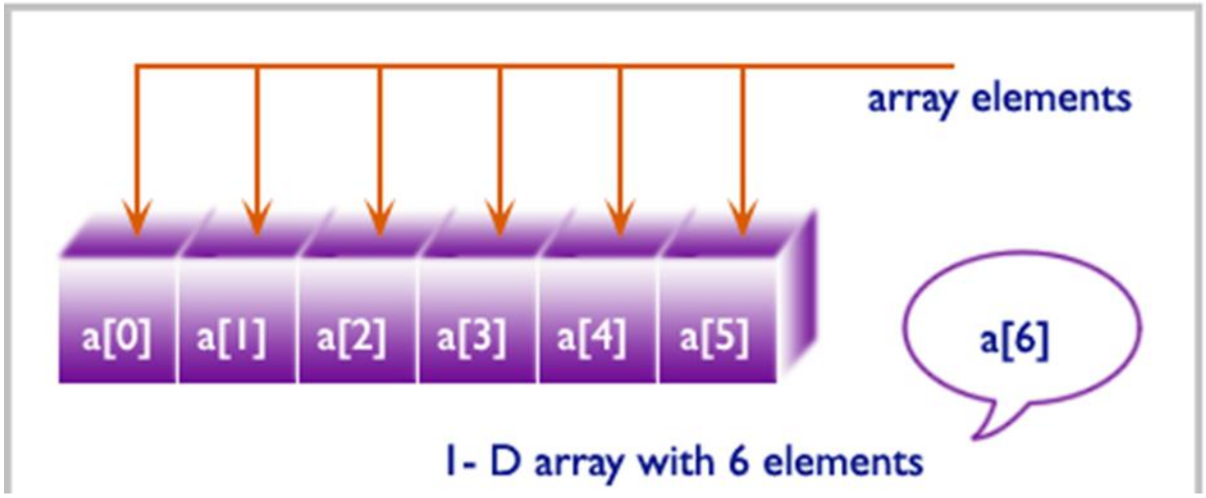


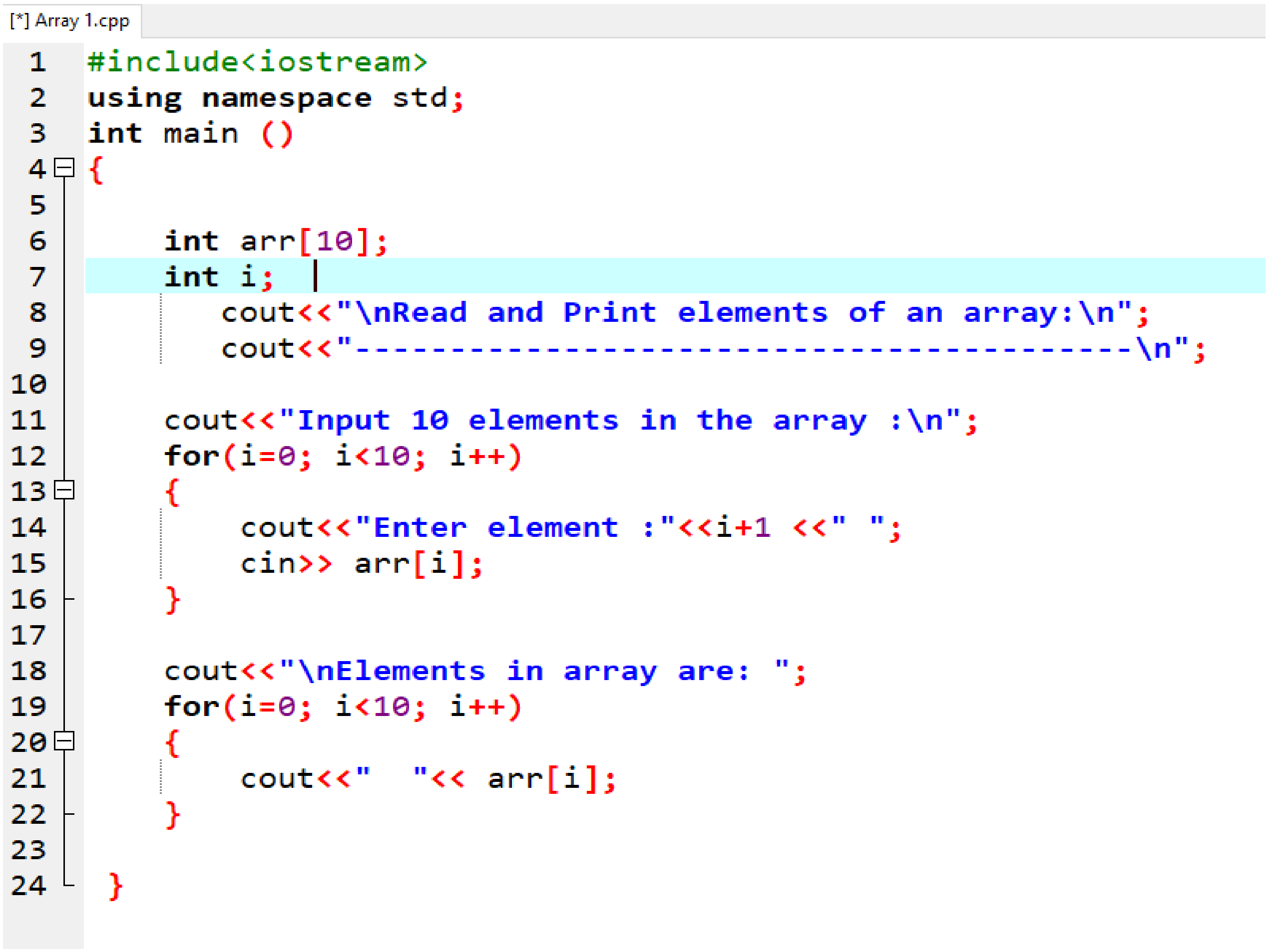
```
#include <iostream>
#include <string>
using namespace std;

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;
}
```

F:\Anjum FAST\Object Oriented Programming\Lab 4\Array6.exe

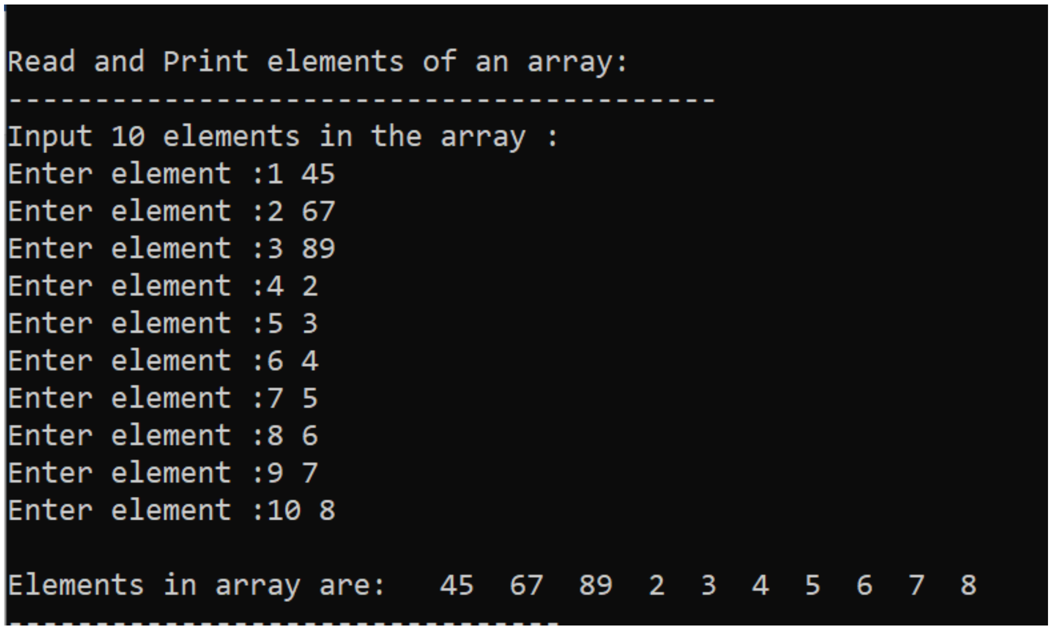
```
Volvo
BMW
Ford
Mazda
Tesla
```

Task: 04- Write a program in C++ to store elements in an array and print it.



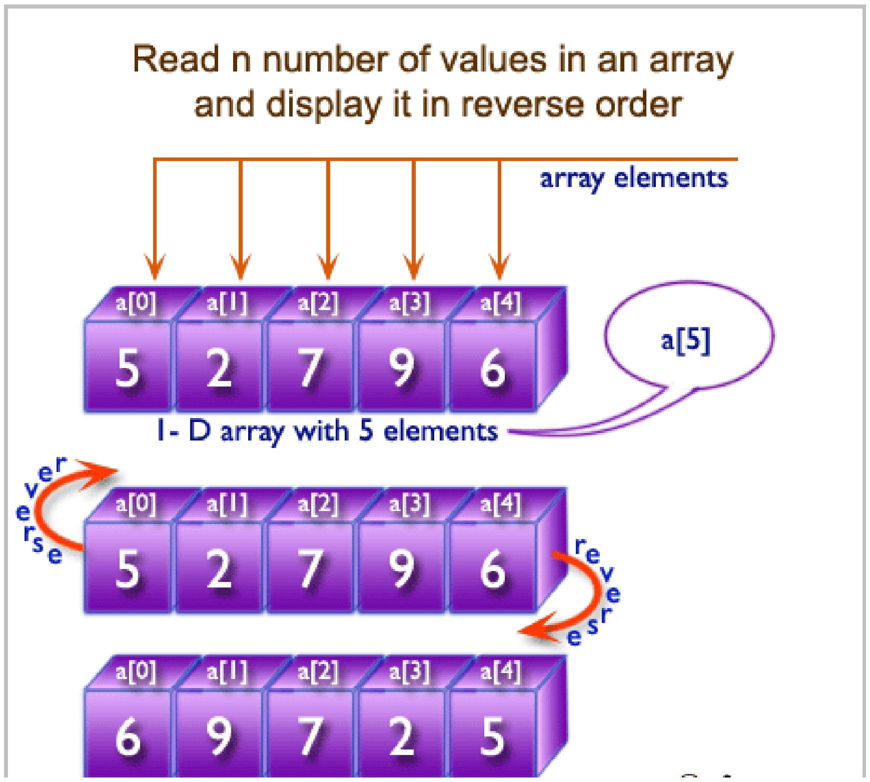


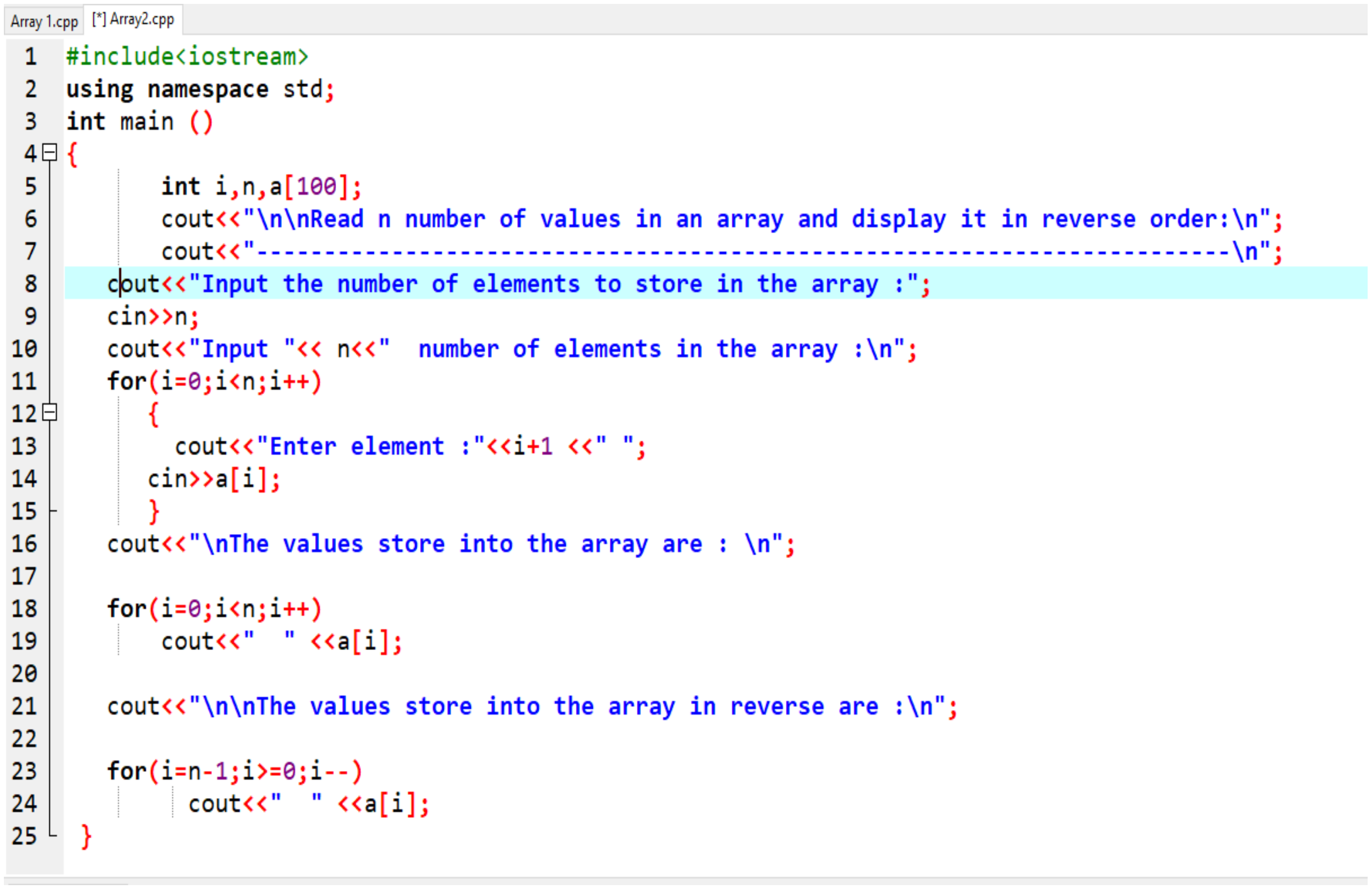
```
#include<iostream>
using namespace std;
int main ()
{

    int arr[10];
    int i;
        cout<<"\nRead and Print elements of an array:\n";
        cout<<"-----------------------------------------\n";

    cout<<"Input 10 elements in the array :\n";
    for(i=0; i<10; i++)
    {
        cout<<"Enter element :"<<i+1 <<" ";
        cin>> arr[i];
    }

    cout<<"\nElements in array are: ";
    for(i=0; i<10; i++)
    {
        cout<<"  "<< arr[i];
    }

}
```



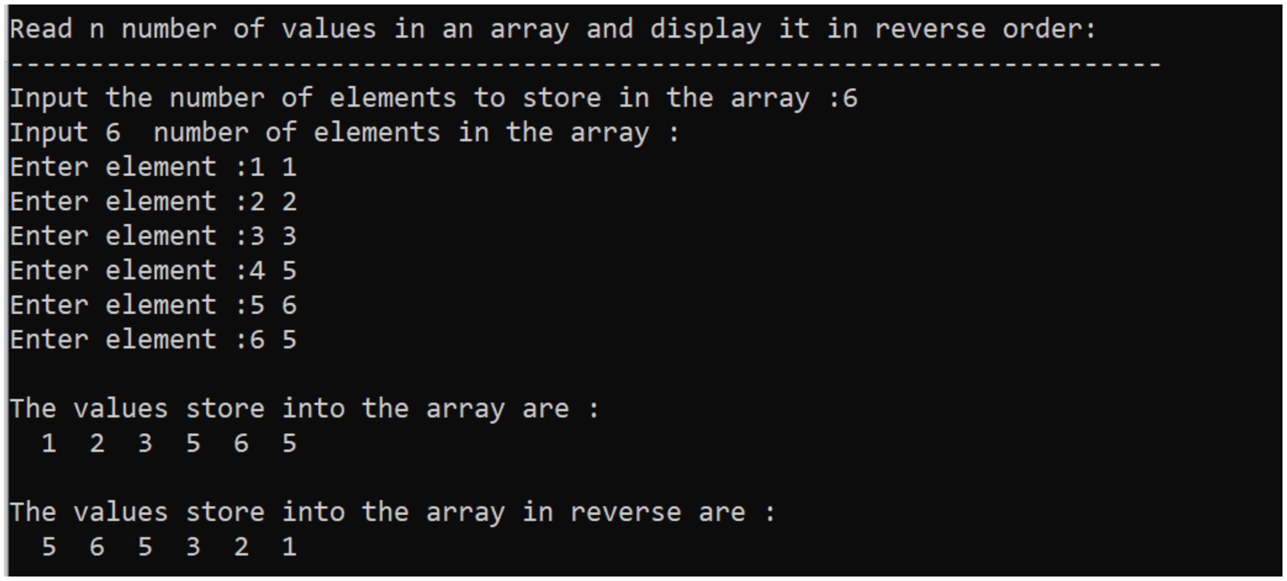
```
Read and Print elements of an array:
-----------------------------------------
Input 10 elements in the array :
Enter element :1 45
Enter element :2 67
Enter element :3 89
Enter element :4 2
Enter element :5 3
Enter element :6 4
Enter element :7 5
Enter element :8 6
Enter element :9 7
Enter element :10 8

Elements in array are:   45  67  89  2  3  4  5  6  7  8
--------------------------------
```

Task-05: Write a program in C++ to read n number of values in an array and display it in reverse order.



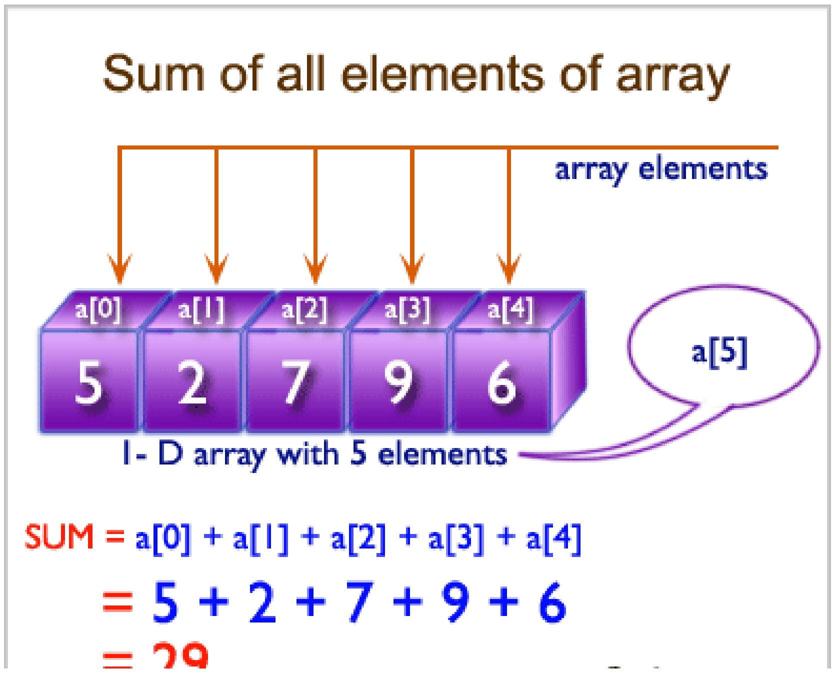

Array 1.cpp [*] Array2.cpp

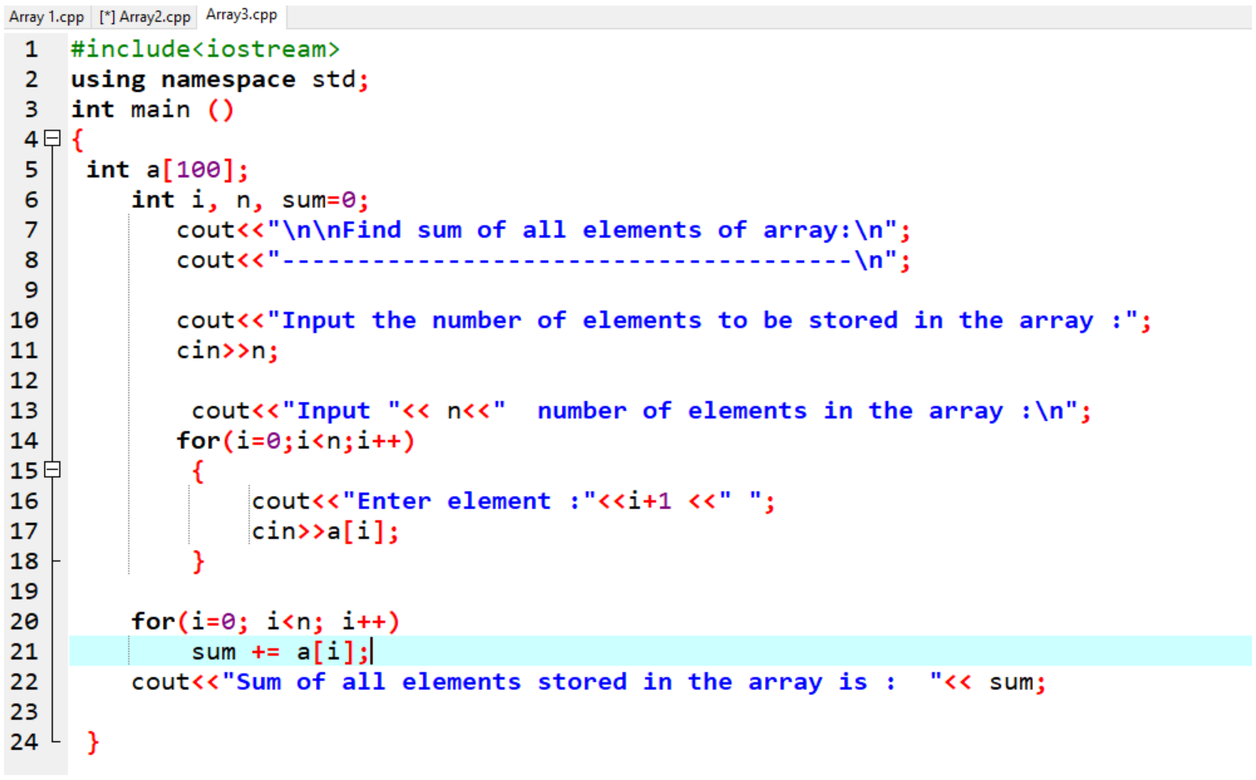
```cpp
#include<iostream>
using namespace std;
int main ()
{
        int i,n,a[100];
        cout<<"\n\nRead n number of values in an array and display it in reverse order:\n";
        cout<<"---------------------------------------------------------------------\n";
    cout<<"Input the number of elements to store in the array :";
    cin>>n;
    cout<<"Input "<< n<<"  number of elements in the array :\n";
    for(i=0;i<n;i++)
       {
         cout<<"Enter element :"<<i+1 <<" ";
       cin>>a[i];
       }
    cout<<"\nThe values store into the array are : \n";

    for(i=0;i<n;i++)
        cout<<"  " <<a[i];

    cout<<"\n\nThe values store into the array in reverse are :\n";

    for(i=n-1;i>=0;i--)
          cout<<"  " <<a[i];
 }
```



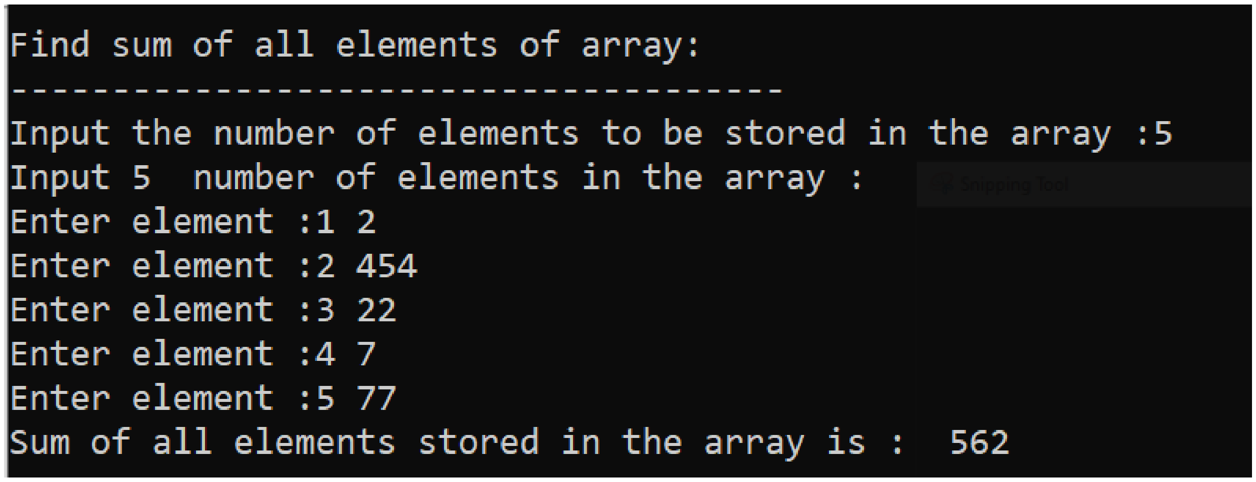
```
Read n number of values in an array and display it in reverse order:
------------------------------------------------------------------------
Input the number of elements to store in the array :6
Input 6  number of elements in the array :
Enter element :1 1
Enter element :2 2
Enter element :3 3
Enter element :4 5
Enter element :5 6
Enter element :6 5

The values store into the array are :
  1  2  3  5  6  5

The values store into the array in reverse are :
  5  6  5  3  2  1
```

Task-06: Write a program in C++ to find the sum of all elements of an array.

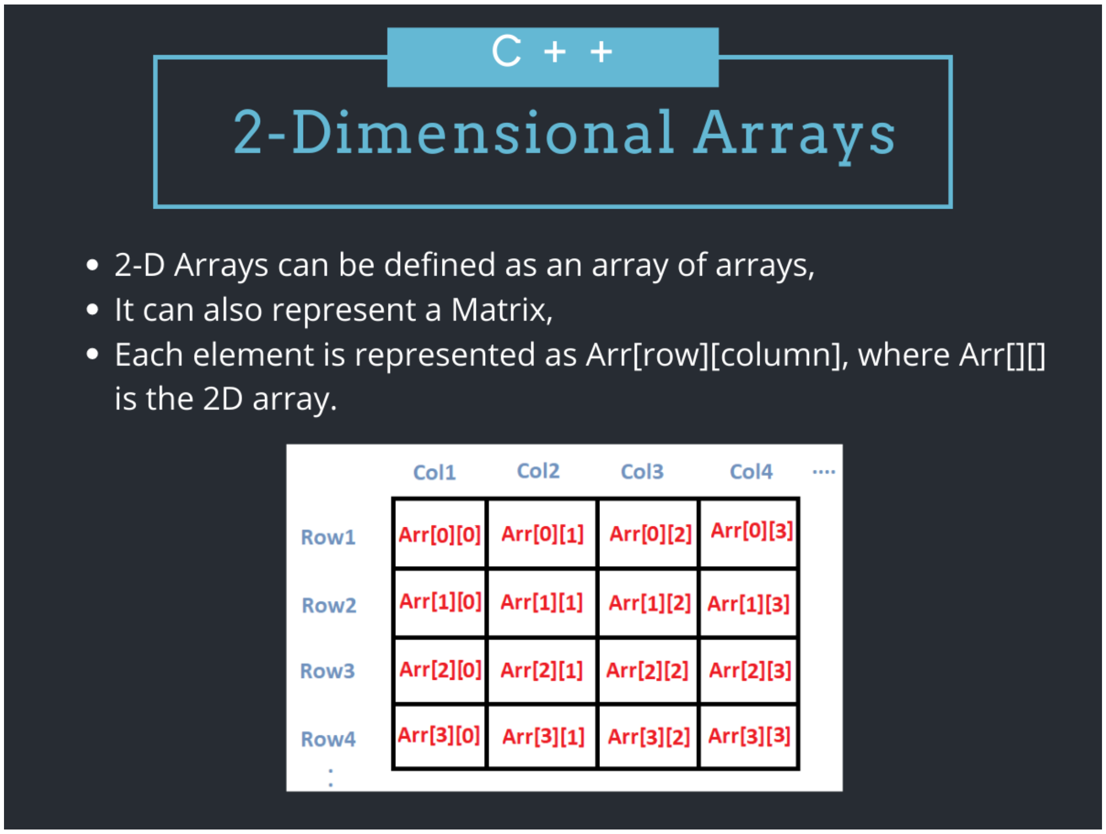

Sum of all elements of array
array elements
a[0] a[1] a[2] a[3] a[4]
5 2 7 9 6
a[5]
1- D array with 5 elements
SUM = a[0] + a[1] + a[2] + a[3] + a[4]
= 5 + 2 + 7 + 9 + 6
= 29



```cpp
#include<iostream>
using namespace std;
int main ()
{
 int a[100];
    int i, n, sum=0;
       cout<<"\n\nFind sum of all elements of array:\n";
       cout<<"---------------------------------------\n";

       cout<<"Input the number of elements to be stored in the array :";
       cin>>n;

        cout<<"Input "<< n<<"  number of elements in the array :\n";
       for(i=0;i<n;i++)
        {
            cout<<"Enter element :"<<i+1 <<" ";
            cin>>a[i];
        }

    for(i=0; i<n; i++)
        sum += a[i];
    cout<<"Sum of all elements stored in the array is :  "<< sum;

 }
```



```
Find sum of all elements of array:
--------------------------------------
Input the number of elements to be stored in the array :5
Input 5  number of elements in the array :
Enter element :1 2
Enter element :2 454
Enter element :3 22
Enter element :4 7
Enter element :5 77
Sum of all elements stored in the array is :  562
```

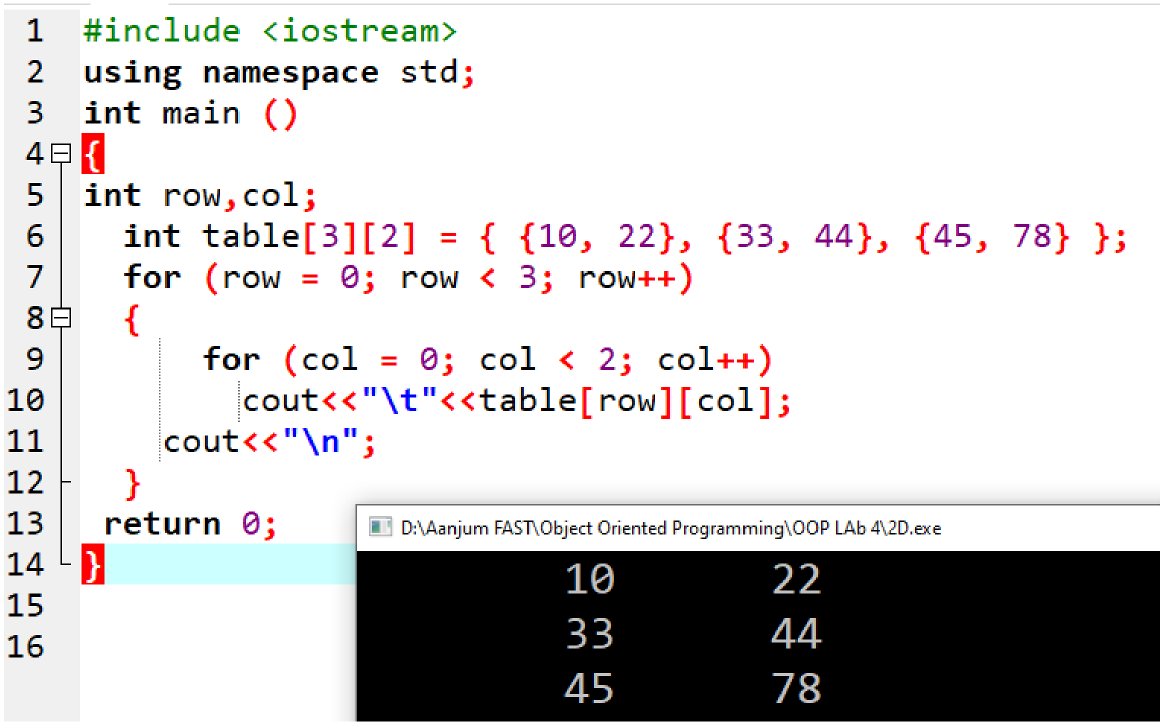


C + +

2-Dimensional Arrays

- 2-D Arrays can be defined as an array of arrays,
- It can also represent a Matrix,
- Each element is represented as Arr[row][column], where Arr[][] is the 2D array.

	Col1	Col2	Col3	Col4	....
Row1	Arr[0][0]	Arr[0][1]	Arr[0][2]	Arr[0][3]	
Row2	Arr[1][0]	Arr[1][1]	Arr[1][2]	Arr[1][3]	
Row3	Arr[2][0]	Arr[2][1]	Arr[2][2]	Arr[2][3]	
Row4	Arr[3][0]	Arr[3][1]	Arr[3][2]	Arr[3][3]	
:					



```
#include <iostream>
using namespace std;
int main ()
{
int row,col;
  int table[3][2] = { {10, 22}, {33, 44}, {45, 78} };
  for (row = 0; row < 3; row++)
  {
      for (col = 0; col < 2; col++)
        cout<<"\t"<<table[row][col];
    cout<<"\n";
  }
 return 0;
}
```

D:\Aanjum FAST\Object Oriented Programming\OOP LAb 4\2D.exe

```
        10      22
        33      44
        45      78
```

C++ References:

Creating References

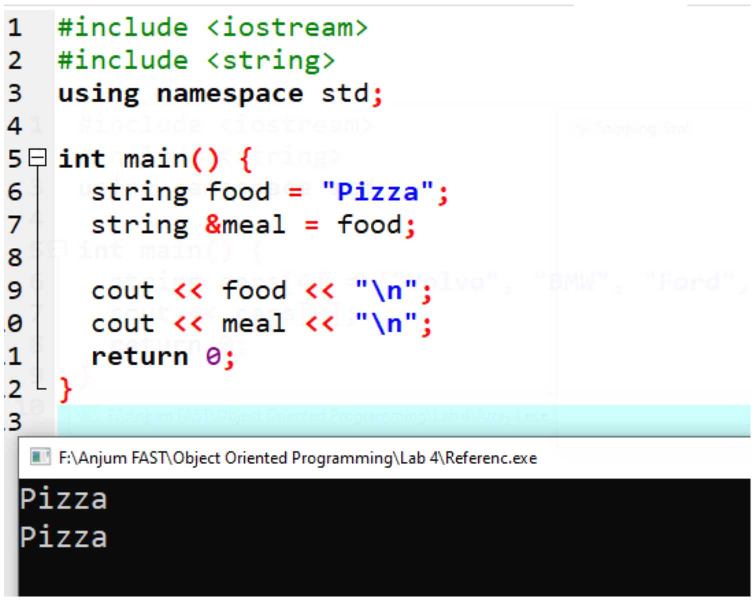
A reference variable is a "reference" to an existing variable, and it is created with the & operator:

```
string food = "Pizza";  // food variable
string &meal = food;    // reference to food
```

Now, we can use either the variable name food or the reference name meal to refer to the food variable:

Example

```
string food = "Pizza";
string &meal = food;

cout << food << "\n";  // Outputs Pizza
cout << meal << "\n";  // Outputs Pizza
```



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

int main() {
  string food = "Pizza";
  string &meal = food;

  cout << food << "\n";
  cout << meal << "\n";
  return 0;
}
```

F:\Anjum FAST\Object Oriented Programming\Lab 4\Referenc.exe

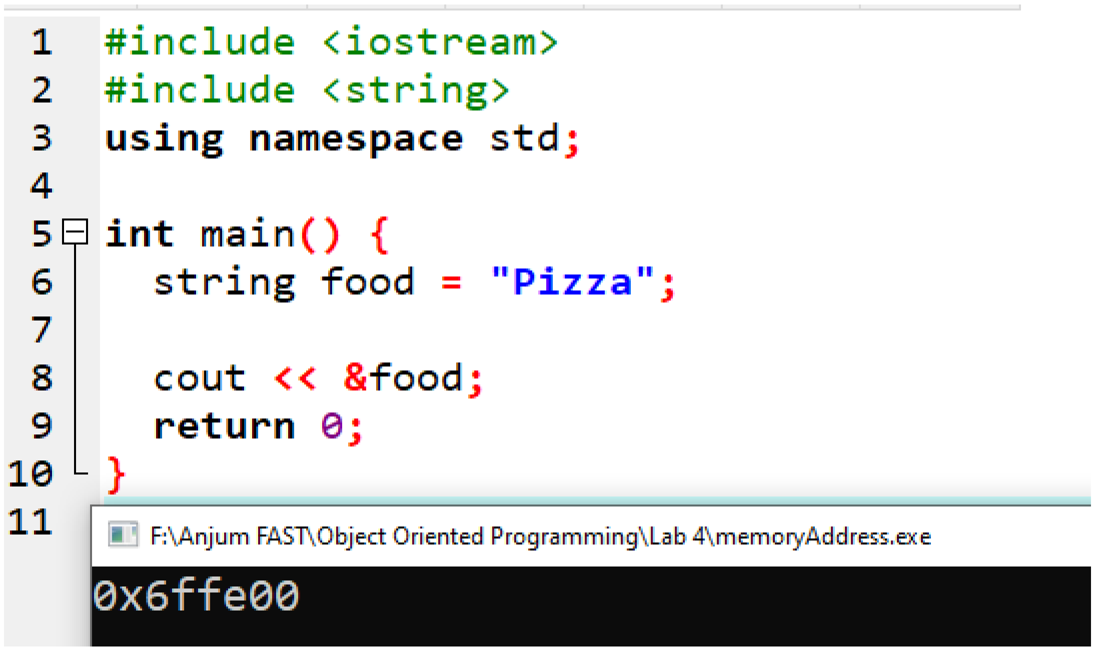
```
Pizza
Pizza
```

Memory Address:

In the example from the previous page, the & operator was used to create a reference variable. But it can also be used to get the memory address of a variable; which is the location of where the variable is stored on the computer.

When a variable is created in C++, a memory address is assigned to the variable. And when we assign a value to the variable, it is stored in this memory address.

To access it, use the & operator, and the result will represent where the variable is stored:

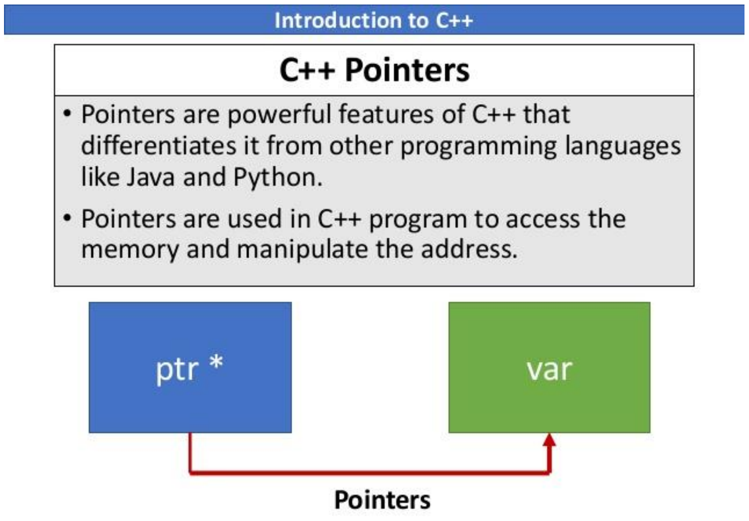


```cpp
#include <iostream>
#include <string>
using namespace std;

int main() {
  string food = "Pizza";

  cout << &food;
  return 0;
}
```

F:\Anjum FAST\Object Oriented Programming\Lab 4\memoryAddress.exe

```
0x6ffe00
```

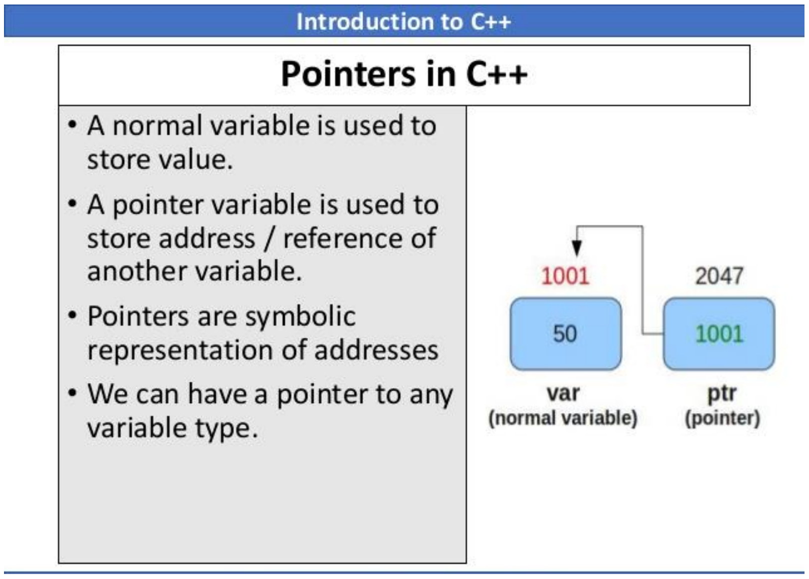
C++ Pointers


Introduction to C++

C++ Pointers

- Pointers are powerful features of C++ that differentiates it from other programming languages like Java and Python.
- Pointers are used in C++ program to access the memory and manipulate the address.

ptr *
var
Pointers


Introduction to C++

Pointers in C++

- A normal variable is used to store value.
- A pointer variable is used to store address / reference of another variable.
- Pointers are symbolic representation of addresses
- We can have a pointer to any variable type.

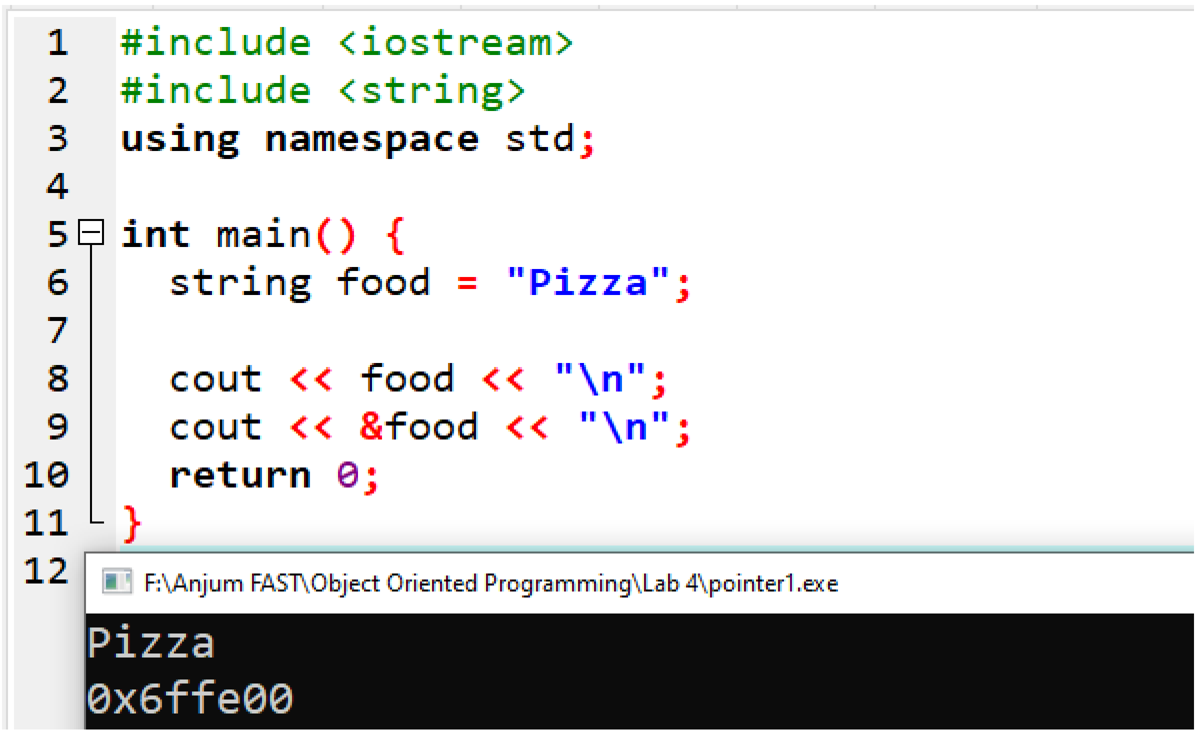
1001
2047
50
1001
var
(normal variable)
ptr
(pointer)

Creating Pointers

You learned from the previous topic, that we can get the **memory address** of a variable by using the `&` operator:

Example

```
string food = "Pizza"; // A food variable of type string
cout << food;  // Outputs the value of food (Pizza)

cout << &food; // Outputs the memory address of food (0x6dfed4)
```



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

int main() {
  string food = "Pizza";

  cout << food << "\n";
  cout << &food << "\n";
  return 0;
}
```

F:\Anjum FAST\Object Oriented Programming\Lab 4\pointer1.exe

```
Pizza
0x6ffe00
```

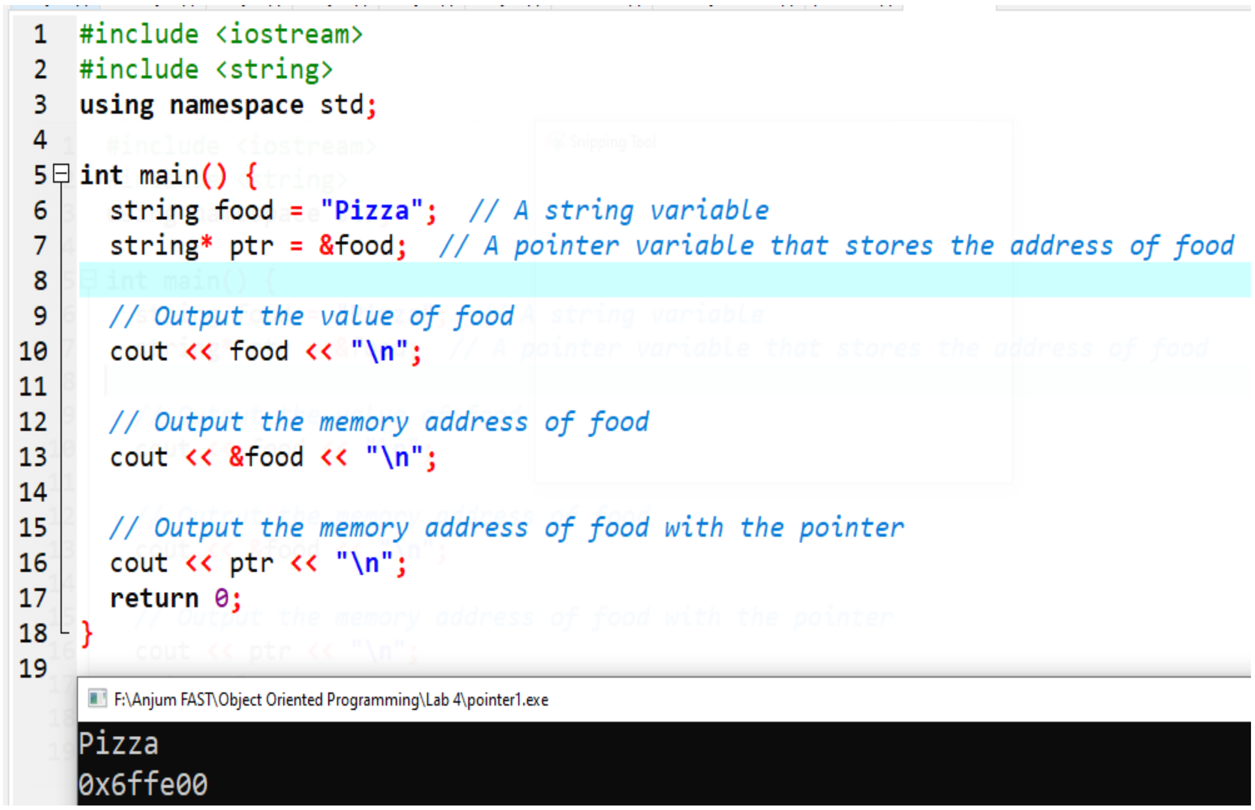
A **pointer** however, is a variable that **stores the memory address as its value**.

A pointer variable points to a data type (like `int` or `string`) of the same type, and is created with the `*` operator. The address of the variable you're working with is assigned to the pointer:

Example

```
string food = "Pizza";  // A food variable of type string
string* ptr = &food;    // A pointer variable, with the name ptr, that
stores the address of food

// Output the value of food (Pizza)
cout << food << "\n";

// Output the memory address of food (0x6dfed4)
```

```
cout << &food << "\n";

// Output the memory address of food with the pointer (0x6dfed4)
cout << ptr << "\n";
```



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

int main() {
  string food = "Pizza";  // A string variable
  string* ptr = &food;  // A pointer variable that stores the address of food

  // Output the value of food
  cout << food << "\n";

  // Output the memory address of food
  cout << &food << "\n";

  // Output the memory address of food with the pointer
  cout << ptr << "\n";
  return 0;
}
```

F:\Anjum FAST\Object Oriented Programming\Lab 4\pointer1.exe

```
Pizza
0x6ffe00
```

Example explained

Create a pointer variable with the name `ptr`, that **points to** a `string` variable, by using the asterisk sign `*` (`string* ptr`). Note that the type of the pointer has to match the type of the variable you're working with.

Use the `&` operator to store the memory address of the variable called `food`, and assign it to the pointer.

Now, `ptr` holds the value of `food`'s memory address.

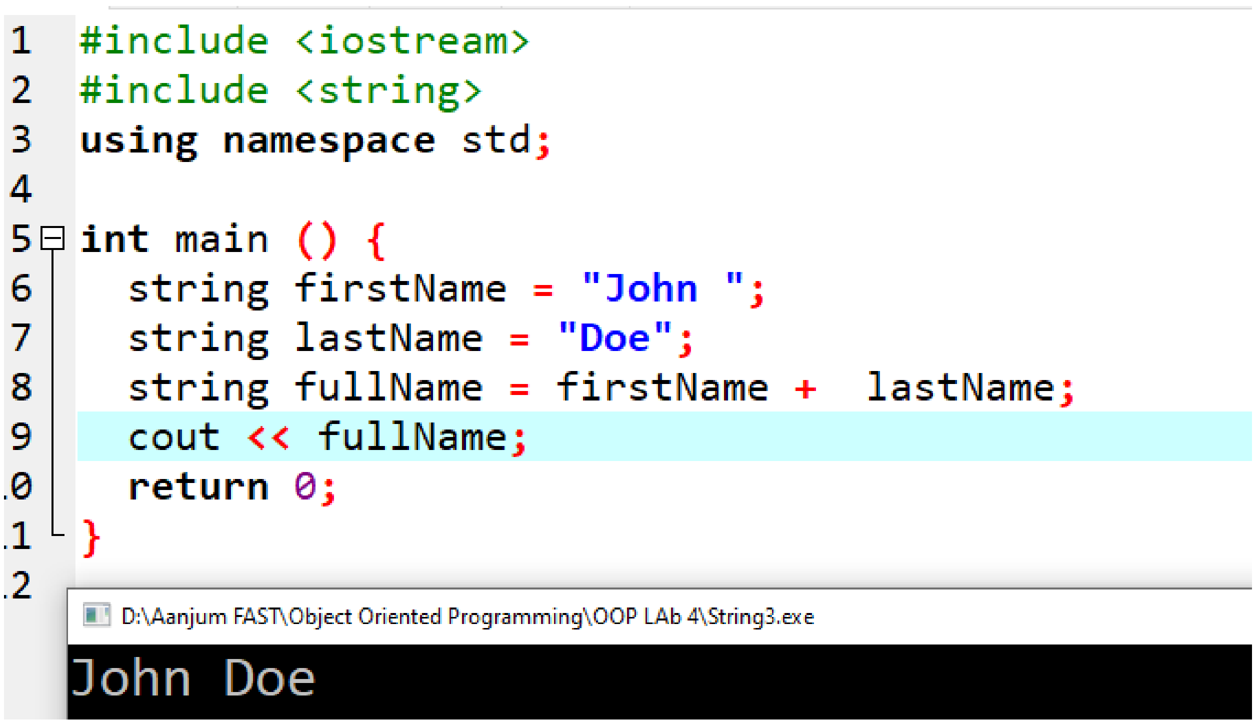
Tip: There are three ways to declare pointer variables, but the first way is preferred:

```
string* mystring; // Preferred
string *mystring;
string * mystring;
```

C++ String Concatenation

String Concatenation

The + operator can be used between strings to add them together to make a new string. This is called **concatenation**:

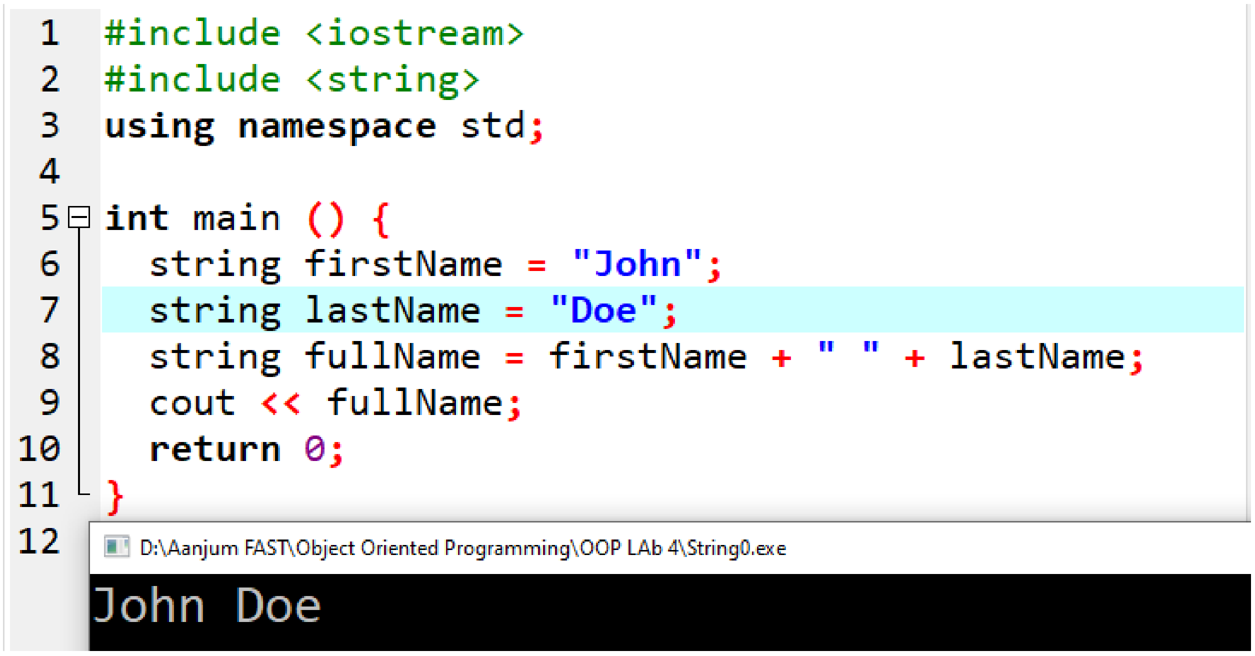


```
#include <iostream>
#include <string>
using namespace std;

int main () {
  string firstName = "John ";
  string lastName = "Doe";
  string fullName = firstName +  lastName;
  cout << fullName;
  return 0;
}
```

D:\Aanjum FAST\Object Oriented Programming\OOP LAb 4\String3.exe

```
John Doe
```

In the example above, we added a space after firstName to create a space between John and Doe on output. However, you could also add a space with quotes (" " or ' '):

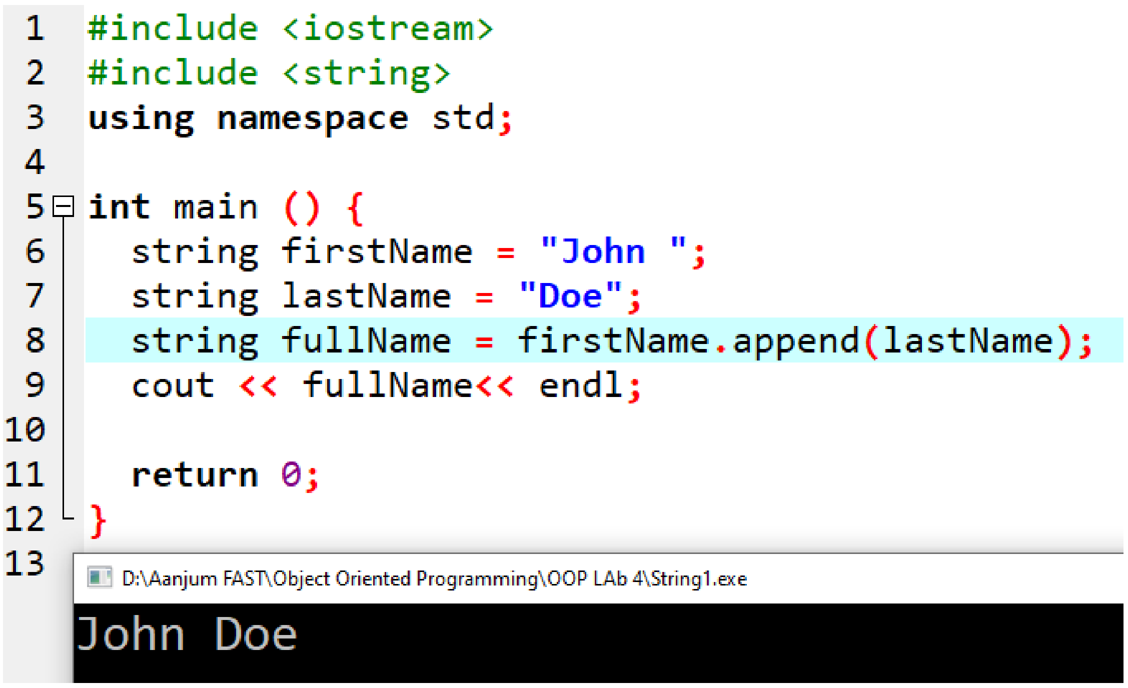


```cpp
#include <iostream>
#include <string>
using namespace std;

int main () {
  string firstName = "John";
  string lastName = "Doe";
  string fullName = firstName + " " + lastName;
  cout << fullName;
  return 0;
}
```

```
D:\Aanjum FAST\Object Oriented Programming\OOP LAb 4\String0.exe
John Doe
```

Append

A string in C++ is actually an object, which contain functions that can perform certain operations on strings. For example, you can also concatenate strings with the `append()` function:



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

int main () {
  string firstName = "John ";
  string lastName = "Doe";
  string fullName = firstName.append(lastName);
  cout << fullName<< endl;

  return 0;
}
```

```
D:\Aanjum FAST\Object Oriented Programming\OOP LAb 4\String1.exe
John Doe
```

It is up to you whether you want to use `+` or `append()`. The major difference between the two, is that the `append()` function is much faster. However, for testing and such, it might be easier to just use `+`.

C++ Numbers and Strings

Adding Numbers and Strings

WARNING!

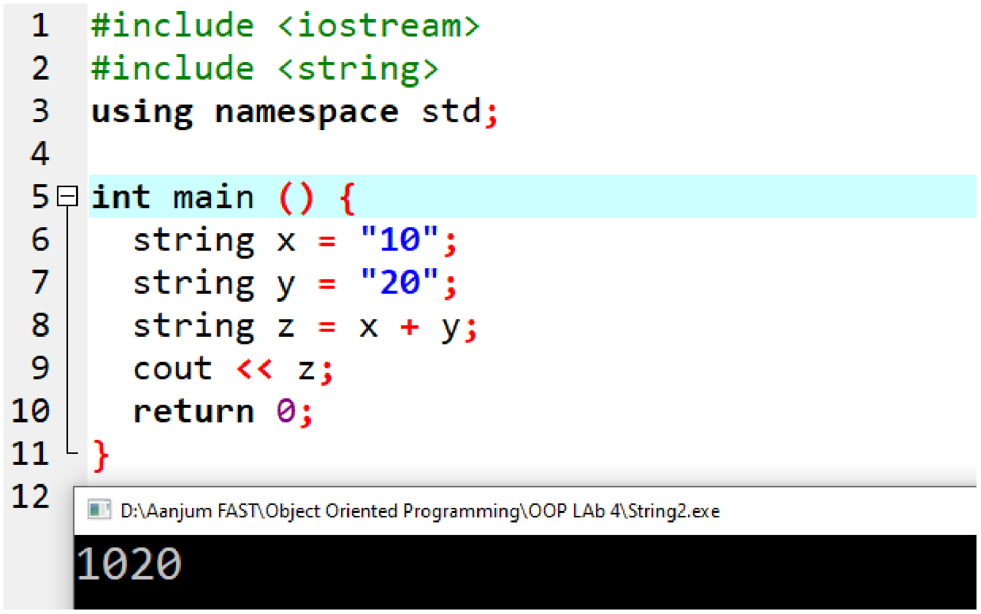
C++ uses the `+` operator for both **addition** and **concatenation**.

Numbers are added. Strings are concatenated.

If you add two numbers, the result will be a number:

Example

```
int x = 10;
int y = 20;
int z = x + y;      // z will be 30 (an integer)
```



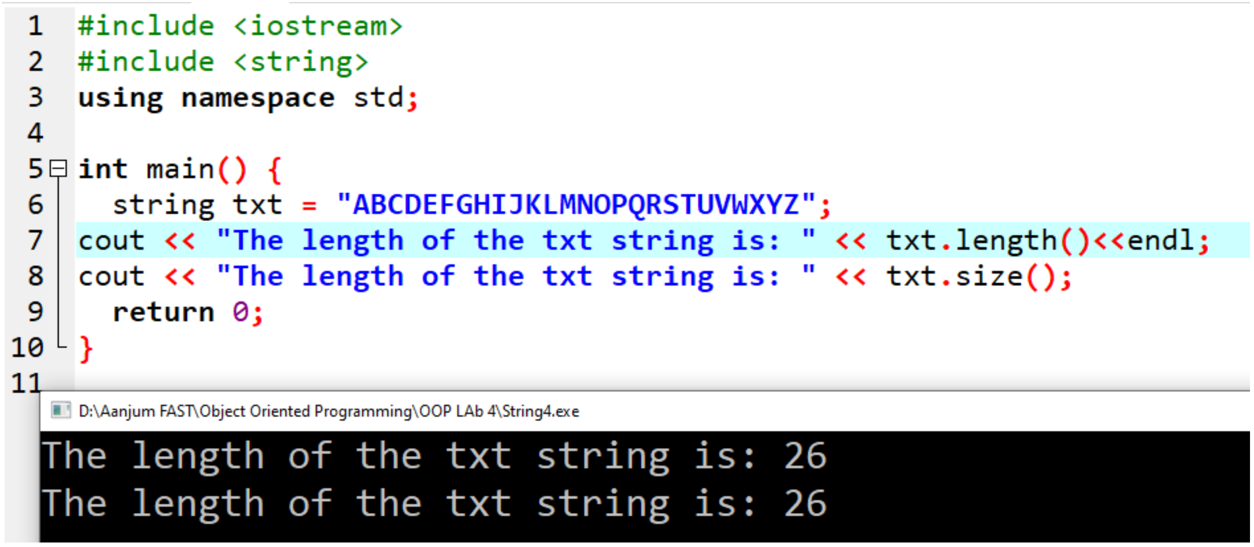
```
#include <iostream>
#include <string>
using namespace std;

int main () {
  string x = "10";
  string y = "20";
  string z = x + y;
  cout << z;
  return 0;
}
```

D:\Aanjum FAST\Object Oriented Programming\OOP LAb 4\String2.exe

```
1020
```

C++ String Length

Tip: You might see some C++ programs that use the `size()` function to get the length of a string. This is just an alias of `length()`. It is completely up to you if you want to use `length()` or `size()`:

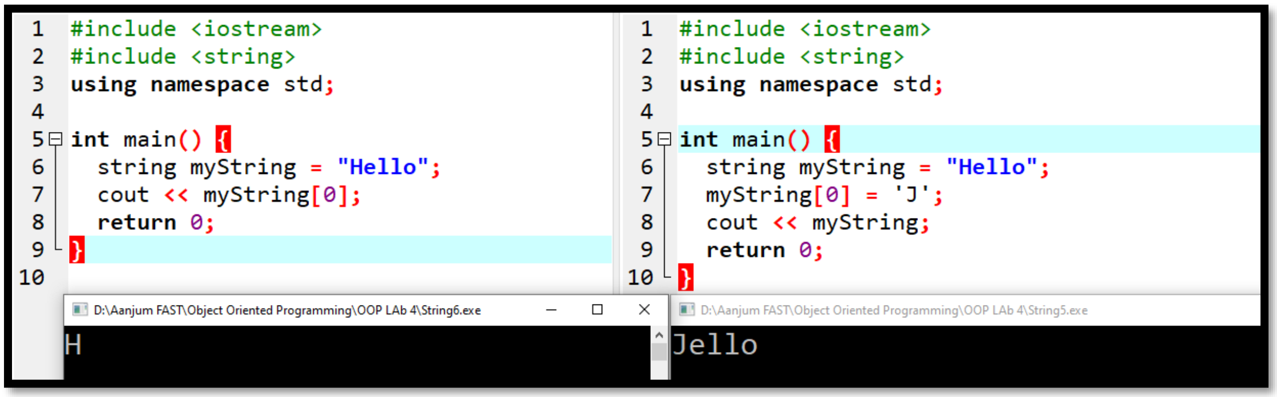


```cpp
#include <iostream>
#include <string>
using namespace std;

int main() {
  string txt = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
cout << "The length of the txt string is: " << txt.length()<<endl;
cout << "The length of the txt string is: " << txt.size();
  return 0;
}
```

D:\Aanjum FAST\Object Oriented Programming\OOP LAb 4\String4.exe

```
The length of the txt string is: 26
The length of the txt string is: 26
```

C++ Access Strings



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

int main() {
  string myString = "Hello";
  cout << myString[0];
  return 0;
}
```

D:\Aanjum FAST\Object Oriented Programming\OOP LAb 4\String6.exe

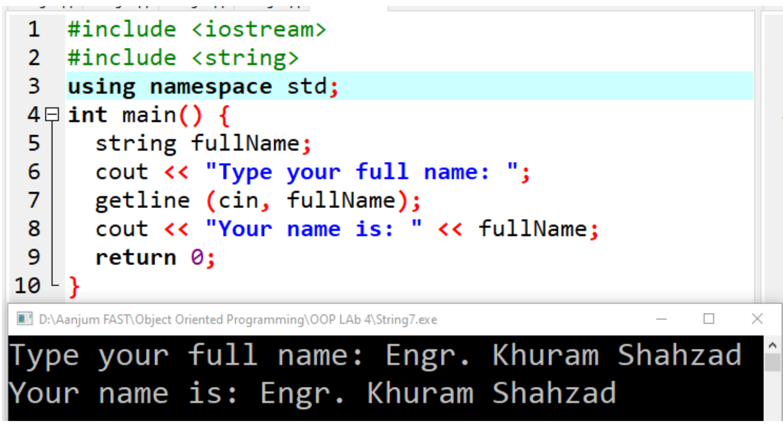
```
H
```

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

int main() {
  string myString = "Hello";
  myString[0] = 'J';
  cout << myString;
  return 0;
}
```

D:\Aanjum FAST\Object Oriented Programming\OOP LAb 4\String5.exe

```
Jello
```

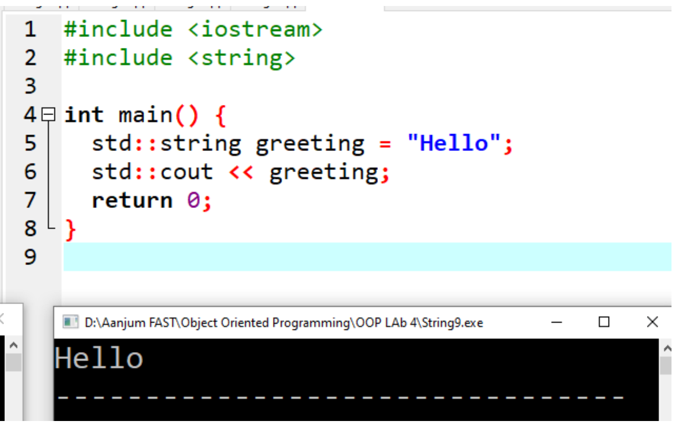
C++ User Input Strings & Omitting Namespace



```
#include <iostream>
#include <string>
using namespace std;
int main() {
  string fullName;
  cout << "Type your full name: ";
  getline (cin, fullName);
  cout << "Your name is: " << fullName;
  return 0;
}
```

D:\Aanjum FAST\Object Oriented Programming\OOP LAb 4\String7.exe

```
Type your full name: Engr. Khuram Shahzad
Your name is: Engr. Khuram Shahzad
```



```
#include <iostream>
#include <string>

int main() {
  std::string greeting = "Hello";
  std::cout << greeting;
  return 0;
}
```

D:\Aanjum FAST\Object Oriented Programming\OOP LAb 4\String9.exe

```
Hello
--------------------------------
```