

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

```
3  int main ()
4  {
5  int M[10], j;
6  /* store seven numbers in array M */
7  M[0] = 2;
8  M[1] = 4;
9  M[2] = 6;
10 M[3] = 8;
11 M[4] = 10;
12 M[5] = 12;
13 M[6] = 14;
14 /* print numbers in M */
15 cout<<"Print all the Numbers : \n";
16 for (j = 0; j < 7; ++j)
17     cout<<"M ["<<j <<"] = "<<M[j]<<endl;
18 /* print numbers in M backwards */
19 cout<<"\nFrom End to Beginning : \n";
20 for (j = 6; j >= 0; --j)
21     cout<<"M ["<<j <<"] = "<<M[j] <<endl;
22 return 0;
23 }
```

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:

```
1  #include <iostream>
2  #include <string>
3  using namespace std;
4
5  int main() {
6      string cars[4] = {"Volvo", "BMW", "Ford", "Mazda"};
7      cout << cars[0];
8      return 0;
9  }
```

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

Volvo

Task -02:

```
1 #include <iostream>
2 #include <string>
3 using namespace std;
4
5 int main() {
6     string cars[4] = {"Volvo", "BMW", "Ford", "Mazda"};
7     for(int i = 0; i < 4; i++) {
8         cout << cars[i] << "\n";
9     }
10    return 0;
11 }
12
```

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

```
Volvo
BMW
Ford
Mazda
```

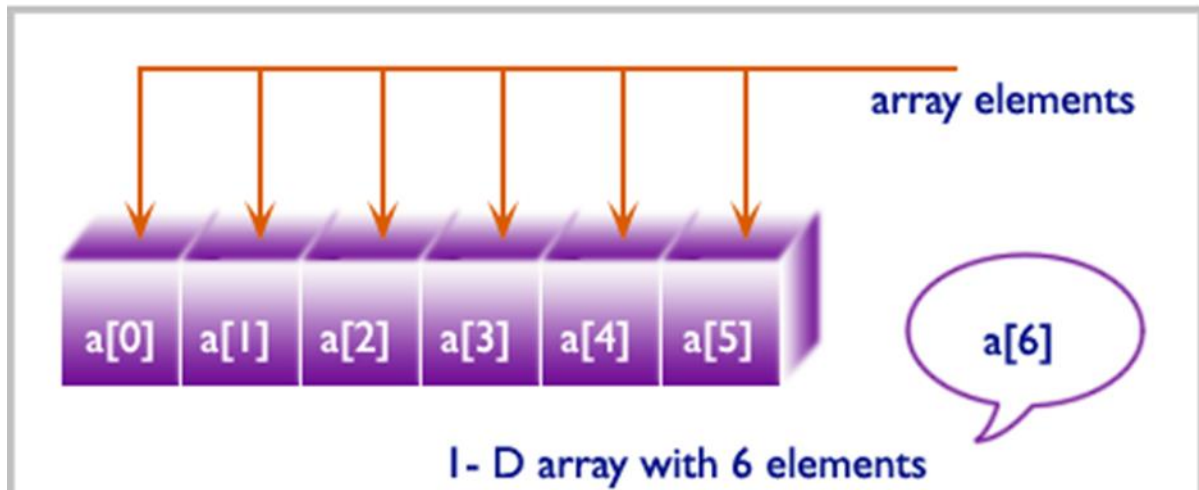
Task -03:

```
1 #include <iostream>
2 #include <string>
3 using namespace std;
4
5 int main() {
6     string cars[5];
7     cars[0] = "Volvo";
8     cars[1] = "BMW";
9     cars[2] = "Ford";
10    cars[3] = "Mazda";
11    cars[4] = "Tesla";
12    for(int i = 0; i < 5; i++) {
13        cout << cars[i] << "\n";
14    }
15    return 0;
16 }
17
```

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.



```
[*] Array 1.cpp
1  #include<iostream>
2  using namespace std;
3  int main ()
4  {
5
6      int arr[10];
7      int i; |
8          cout<<"\nRead and Print elements of an array:\n";
9          cout<<"-----\n";
10
11         cout<<"Input 10 elements in the array :\n";
12         for(i=0; i<10; i++)
13         {
14             cout<<"Enter element : "<<i+1 <<" ";
15             cin>> arr[i];
16         }
17
18         cout<<"\nElements in array are: ";
19         for(i=0; i<10; i++)
20         {
21             cout<<" "<< arr[i];
22         }
23
24     }
```

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

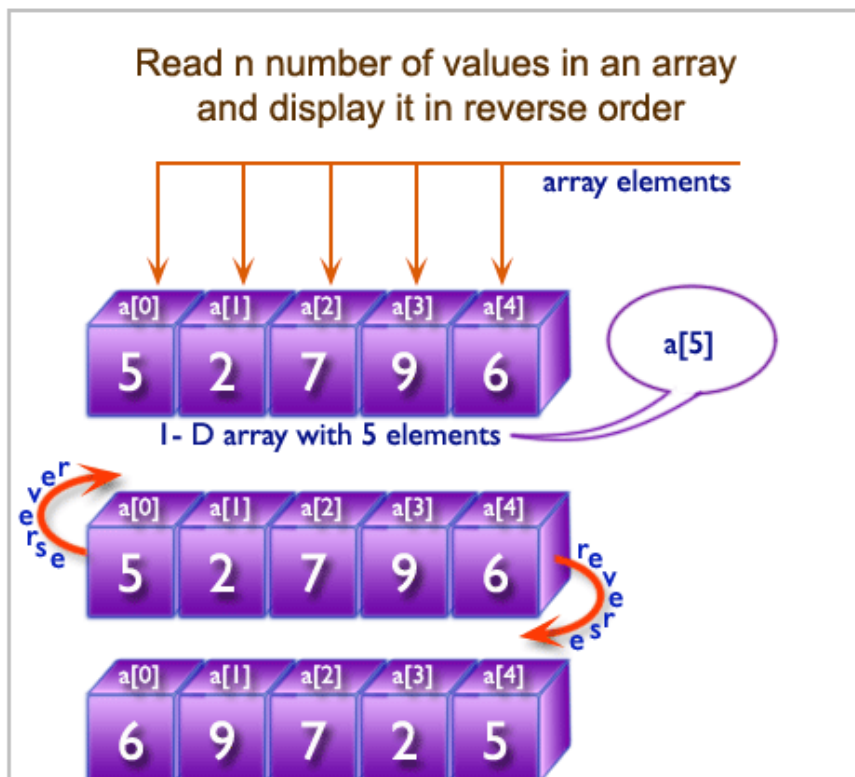
Enter element :8 6

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
1  #include<iostream>
2  using namespace std;
3  int main ()
4  {
5      int i,n,a[100];
6      cout<<"\n\nRead n number of values in an array and display it in reverse order:\n";
7      cout<<"-----\n";
8      cout<<"Input the number of elements to store in the array :";
9      cin>>n;
10     cout<<"Input "<< n<<"  number of elements in the array :\n";
11     for(i=0;i<n;i++)
12     {
13         cout<<"Enter element :"<<i+1 <<" ";
14         cin>>a[i];
15     }
16     cout<<"\nThe values store into the array are : \n";
17
18     for(i=0;i<n;i++)
19         cout<<" " <<a[i];
20
21     cout<<"\n\nThe values store into the array in reverse are :\n";
22
23     for(i=n-1;i>=0;i--)
24         cout<<" " <<a[i];
25 }

```

```

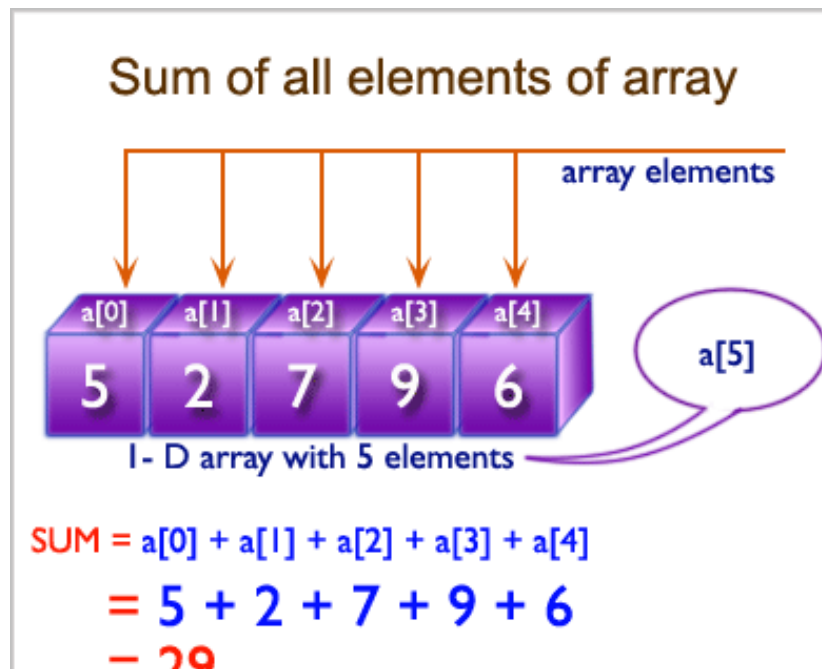
Read n number of values in an array and display it in reverse order:
-----
Input the number of elements to store in the array :6
Input 6  number of elements in the array :
Enter element :1 1
Enter element :2 2
Enter element :3 3
Enter element :4 5
Enter element :5 6
Enter element :6 5

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.



```
Array 1.cpp  [*] Array2.cpp  Array3.cpp
1  #include<iostream>
2  using namespace std;
3  int main ()
4  {
5      int a[100];
6      int i, n, sum=0;
7      cout<<"\n\nFind sum of all elements of array:\n";
8      cout<<"-----\n";
9
10     cout<<"Input the number of elements to be stored in the array :";
11     cin>>n;
12
13     cout<<"Input "<< n<<"  number of elements in the array :\n";
14     for(i=0;i<n;i++)
15     {
16         cout<<"Enter element :"<<i+1 <<" ";
17         cin>>a[i];
18     }
19
20     for(i=0; i<n; i++)
21         sum += a[i];
22     cout<<"Sum of all elements stored in the array is : "<< sum;
23
24 }
```


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]	
:					

```
1  #include <iostream>
2  using namespace std;
3  int main ()
4  {
5      int row,col;
6      int table[3][2] = { {10, 22}, {33, 44}, {45, 78} };
7      for (row = 0; row < 3; row++)
8      {
9          for (col = 0; col < 2; col++)
10             cout<<"\t"<<table[row][col];
11         cout<<"\n";
12     }
13     return 0;
14 }
```

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
```

```
1 #include <iostream>
2 #include <string>
3 using namespace std;
4
5 int main() {
6     string food = "Pizza";
7     string &meal = food;
8
9     cout << food << "\n";
10    cout << meal << "\n";
11    return 0;
12 }
```

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:

```
1  #include <iostream>
2  #include <string>
3  using namespace std;
4
5  int main() {
6      string food = "Pizza";
7
8      cout << &food;
9      return 0;
10 }
```

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

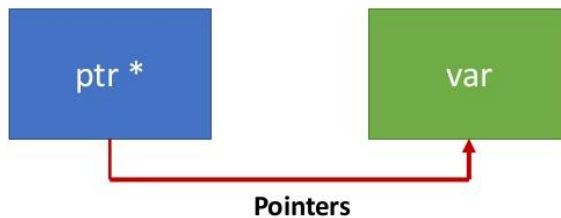
```
0x6ffe00
```

C++ Pointers

Introduction to C++

C++ Pointers

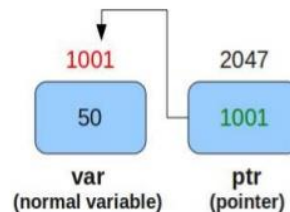
- Pointers are powerful features of C++ that differentiates it from other programming languages like Java and Python.
- Pointers are used in C++ program to access the memory and manipulate the address.



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.



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)
```

```
1  #include <iostream>
2  #include <string>
3  using namespace std;
4
5  int main() {
6      string food = "Pizza";
7
8      cout << food << "\n";
9      cout << &food << "\n";
10     return 0;
11 }
12
```

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
```

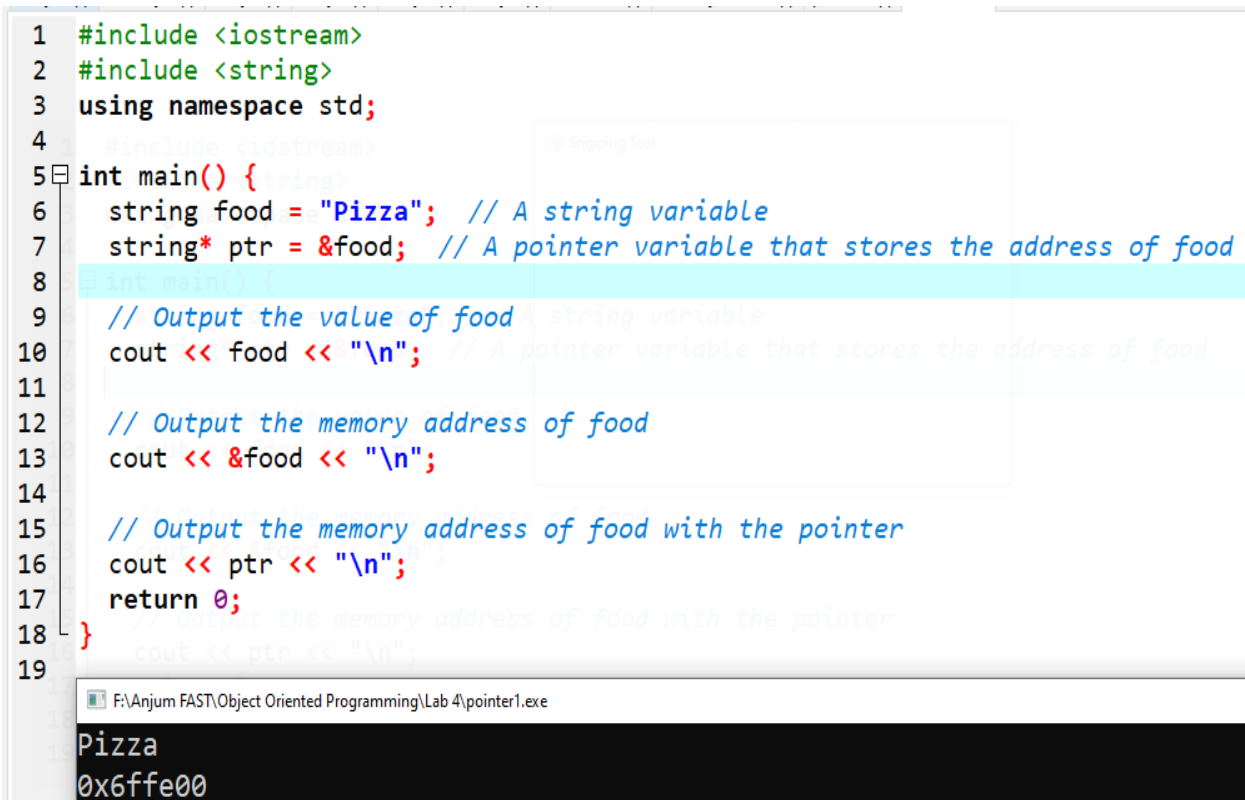
```
// Output the value of food (Pizza)
cout << food << "\n";
```

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

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

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

```
cout << ptr << "\n";
```



```
1 #include <iostream>
2 #include <string>
3 using namespace std;
4
5 int main() {
6     string food = "Pizza"; // A string variable
7     string* ptr = &food; // A pointer variable that stores the address of food
8
9     // Output the value of food
10    cout << food << "\n";
11
12    // Output the memory address of food
13    cout << &food << "\n";
14
15    // Output the memory address of food with the pointer
16    cout << ptr << "\n";
17    return 0;
18 }
19
```

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**:

```
1 #include <iostream>
2 #include <string>
3 using namespace std;
4
5 int main () {
6     string firstName = "John ";
7     string lastName = "Doe";
8     string fullName = firstName + lastName;
9     cout << fullName;
10    return 0;
11 }
12
```

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 `' '`):

```
1 #include <iostream>
2 #include <string>
3 using namespace std;
4
5 int main () {
6     string firstName = "John";
7     string lastName = "Doe";
8     string fullName = firstName + " " + lastName;
9     cout << fullName;
10    return 0;
11 }
12
```

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:

```
1 #include <iostream>
2 #include <string>
3 using namespace std;
4
5 int main () {
6     string firstName = "John ";
7     string lastName = "Doe";
8     string fullName = firstName.append(lastName);
9     cout << fullName<< endl;
10
11    return 0;
12 }
13
```

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)
```

```
1 #include <iostream>
2 #include <string>
3 using namespace std;
4
5 int main () {
6     string x = "10";
7     string y = "20";
8     string z = x + y;
9     cout << z;
10    return 0;
11 }
```

12 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()`:

```
1 #include <iostream>
2 #include <string>
3 using namespace std;
4
5 int main() {
6     string txt = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
7     cout << "The length of the txt string is: " << txt.length() << endl;
8     cout << "The length of the txt string is: " << txt.size();
9     return 0;
10 }
```

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

```
1 #include <iostream>
2 #include <string>
3 using namespace std;
4
5 int main() {
6     string myString = "Hello";
7     cout << myString[0];
8     return 0;
9 }
```

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

H

```
1 #include <iostream>
2 #include <string>
3 using namespace std;
4
5 int main() {
6     string myString = "Hello";
7     myString[0] = 'J';
8     cout << myString;
9     return 0;
10 }
```

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

Jello

C++ User Input Strings & Omitting Namespace

```
1 #include <iostream>
2 #include <string>
3 using namespace std;
4 int main() {
5     string fullName;
6     cout << "Type your full name: ";
7     getline (cin, fullName);
8     cout << "Your name is: " << fullName;
9     return 0;
10 }
```

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

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

```
1 #include <iostream>
2 #include <string>
3
4 int main() {
5     std::string greeting = "Hello";
6     std::cout << greeting;
7     return 0;
8 }
9
```

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

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