# C++ Strings

programiz.com/cpp-programming/strings

In this tutorial, you'll learn to handle strings in C++. You'll learn to declare them, initialize them and use them for various input/output operations.

String is a collection of characters. There are two types of strings commonly used in C++ programming language:

- Strings that are objects of string class (The Standard C++ Library string class)
- C-strings (C-style Strings)

## C-strings

In C programming, the collection of characters is stored in the form of arrays. This is also supported in C++ programming. Hence it's called C-strings.

C-strings are arrays of type `char` terminated with null character, that is, `\0` (ASCII value of null character is 0).

### How to define a C-string?

```
char str[] = "C++";
```

In the above code, `str` is a string and it holds 4 characters.

Although, " `C++` " has 3 character, the null character `\0` is added to the end of the string automatically.

### Alternative ways of defining a string

```
char str[4] = "C++";

char str[] = {'C','+','+','\0'};

char str[4] = {'C','+','+','\0'};
```

Like arrays, it is not necessary to use all the space allocated for the string. For example:

```
char str[100] = "C++";
```

### Example 1: C++ String to read a word

**C++ program to display a string entered by user.**

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

int main()
{
    char str[100];

    cout << "Enter a string: ";
    cin >> str;
    cout << "You entered: " << str << endl;

    cout << "\nEnter another string: ";
    cin >> str;
    cout << "You entered: "<<str<<endl;

    return 0;
}
```

### Output

```
Enter a string: C++
You entered: C++

Enter another string: Programming is fun.
You entered: Programming
```

Notice that, in the second example only "Programming" is displayed instead of "Programming is fun".

This is because the extraction operator >> works as `scanf()` in C and considers a space " " has a terminating character.

---

## Example 2: C++ String to read a line of text

### C++ program to read and display an entire line entered by user.

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

int main()
{
    char str[100];
    cout << "Enter a string: ";
    cin.get(str, 100);

    cout << "You entered: " << str << endl;
    return 0;
}
```

### Output

```
Enter a string: Programming is fun.
You entered: Programming is fun.
```

To read the text containing blank space, `cin.get` function can be used. This function takes two arguments.

First argument is the name of the string (address of first element of string) and second argument is the maximum size of the array.

In the above program, *str* is the name of the string and `100` is the maximum size of the array.

---

## string Object

In C++, you can also create a string object for holding strings.

Unlike using char arrays, string objects has no fixed length, and can be extended as per your requirement.

---

### Example 3: C++ string using string data type

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

int main()
{
    // Declaring a string object
    string str;
    cout << "Enter a string: ";
    getline(cin, str);

    cout << "You entered: " << str << endl;
    return 0;
}
```

**Output**

```
Enter a string: Programming is fun.
You entered: Programming is fun.
```

In this program, a string *str* is declared. Then the string is asked from the user.

Instead of using `cin>>` or `cin.get()` function, you can get the entered line of text using `getline()`.

`getline()` function takes the input stream as the first parameter which is `cin` and `str` as the location of the line to be stored.

---

## Passing String to a Function

Strings are passed to a function in a similar way arrays are passed to a function.

```
#include <iostream>

using namespace std;

void display(char *);
void display(string);

int main()
{
    string str1;
    char str[100];

    cout << "Enter a string: ";
    getline(cin, str1);

    cout << "Enter another string: ";
    cin.get(str, 100, '\n');

    display(str1);
    display(str);
    return 0;
}

void display(char s[])
{
    cout << "Entered char array is: " << s << endl;
}

void display(string s)
{
    cout << "Entered string is: " << s << endl;
}
```

## Output

```
Enter a string:  Programming is fun.
Enter another string:  Really?
Entered string is: Programming is fun.
Entered char array is: Really?
```

In the above program, two strings are asked to enter. These are stored in *str* and *str1* respectively, where str is a `char` array and str1 is a `string` object.

Then, we have two functions `display()` that outputs the string onto the string.

The only difference between the two functions is the parameter. The first `display()` function takes char array as a parameter, while the second takes string as a parameter.

This process is known as function overloading. Learn more about Function Overloading.