String Functions In C++

Read | Courses | Practice

A string is referred to as an array of characters. In C++, a stream/sequence of characters is stored in a char array. C++ includes the std::string class that is used to represent strings. It is one of the most fundamental datatypes in C++ and it comes with a huge set of inbuilt functions. In this article, will look at the functions of string computations.

What is std::string?

> *The std::string is a class in C++ since C++98. This class is the standard representation for a text string. It includes some typical string operations like find, replace, concatenate, compare etc. It is present in* ***<string>*** *header file.*

Declaration and Initialization of String

```
std::string company = "GeeksforGeeks!";
```

Commonly Used String Functions in C++

The std::string class contains functions to provide some common string operations. The below table contains some of the most commonly used functions in C++:

S. No.	Category	Functions and Operators	Functionality
1.	String Length	length() or size()	It will return the length of the string.
2.	Accessing Characters	Indexing (using array[index])	To access individual characters using array indexing.
		at()	Used to access a character at a specified index.
3.	Appending and Concatenating Strings	*+ Operator*	+ operator is used to concatenate two strings.
		append()	The append() function adds one string to the end of another.
4.	String Comparison	*== Operator*	You can compare strings using the == operator.
		compare()	The compare() function returns an integer value indicating the comparison result.
5.	Substrings	substr()	Use the substr() function to extract a substring from a string.
6.	Searching	find()	The find() function returns the position of the first occurrence of a substring.
7.	Modifying Strings	replace()	Use the replace() function to modify a part of the string.

No.		Operators	
		insert()	The insert() function adds a substring at a specified position.
		erase()	Use the erase() function to remove a part of the string.
8.	Conversion	c_str()	To obtain a C-style string from a std::string, you can use the c_str() function.

Note: *The above functions only works for C++ Style strings (std::string objects) not for C Style strings (array of characters).*

1. String Length – length() or size()

We can find the length of the string (number of characters) using either **length() or size()** function of the std::string class.

Syntax

```
string_object.size()
or
string_object.length()
```

Parameters

- This function does not take any parameter.

Return Value

- This function returns the number of characters in the string object.

Example

```
std::string text = "geeksforGeeks";
int length = text.length();
//or
int length = text.size();
```

It will return the length of the string **text** which is 13.

2. Accessing Characters – at()

Generally, we can access the character of a string using the **[] array subscript operator** and indexing. But std::string also has a function named **at()** which can be used to access the characters of the string.

Syntax

```
string_object.at(index);
```

Parameters

- **index:** It represents the position of the character in the string.

Return Value

- This function returns the character present at the **index.**

Example

```
std::string str = "GEEKSFORGEEKS";
std::cout << str.at(3);
```

The std::cout will print K on the console as it is the character present at index 3.

3. Concatenating Strings – append() or + Operator

We can concatenate string in C++ using two methods:

1. + Operator

The + operator is overloaded in the std::string class to perform string concatenation.

Syntax

```
string_object1 + string_object2
```

Example

```
std::string firstName = "Geeks";
std::string lastName = "forGeeks";
std::string fullName = firstName + " " + lastName;
```

+ operator is used to concatenate two strings. The string fullName will be “GeeksforGeeks”.

2. append()

The append() function is another member function to concatenate two strings.

Syntax

```
string_object1.append(string2)
```

Parameters

- **string2:** This function takes the string to be appended as a parameter. It can be both C or C++ Style string.

Return Value

- Reference to the final string.

```
std::string base = "Hey! Geeks";
base.append(" Welcome to GeeksforGeeks!"); // Append a string
```

The append() function adds one string to the end of another.

4. String Comparison – compare() or == Operator

Just like the concatenation, we can do the string comparison using two methods:

1. == Operator

The equality operator can be used to compare the two strings as it is overloaded for this operation in the std::string class.

Syntax

```
string_object1 == string_object2
```

This will return **true** if both the strings are equal, otherwise returns **false.**

Example

```
std::string str1 = "apple";
std::string str2 = "banana";
if (str1 == str2) {
    std::cout << "Strings are equal";
}
else {
    std::cout << "Strings are not equal";
}
```

Here, “Strings are not equal” will be printed as the == operator will return **false.**

2. compare()

The compare() function is a member function of std::string class which can be used to compare two strings.

Syntax

```
str1.compare(str2);
```

Parameters

- **str2:** It is the string to be compared. It can be both C or C++ style string.

Return Value

- If the strings are equal, return **zero.**
- If str1 is greater than str2, return value **>0**
- If str2 is greater than str1, return value **<0**

Example

```
string str1 = "Geeks";
string str2: = "Geeksfor";
int result = str1.compare(str2);
```

The result will contain a value less than zero as str2 is greater than str1.

We can also compare the substring of str2 using the compare function():

```
str1.compare(position, length, str2);
```

where,

- **position**: position of the first character substring.
- **length:** length of the substring.
- **str2:** String object to be compared.

5. Searching – find()

We can use the **find()** function of the std::string class to check whether a given character or a substring is present in the string or a part of string.

Syntax of find()

```
str1.find(var);
```

Parameters

- **var:** It can be a C style string, C++ style string, or a character that is to be searched in the string.

Return Value

- It returns the pointer to the first occurrence of the character or a substring in the string.

Example

```
std::string text = "C++ Programming";
int position = text.find("Programming"); // Find the position of a
substring
```

The position variable will contain 4 which is the start of the first occurrence of the string “Programming” in string text.

6. Generate Substring – substr()

We can use the **substr() function** to generate a part of the string as a new string object. It is a member function of the std::string class.

Syntax of substr() in C

```
str1.substr(start, end);
```

Parameters

- **start:** Starting position of the substring to be generated.
- **end:** Ending of the substring to be generated.

Return Type

- It returns the newly created string object.

Example

```
std::string text = "Hello, World!";
std::string sub = text.substr(7, 5); // Extract "World"
```

In the above example. the **sub** string will contain the “World”.

Modifying Strings

The following function allows us to modify the current string.

1. insert()

The insert() function not only allows us to add a string but also allows us to add it at the specified position. It is also a member function of the std::string class.

Syntax

```
str1.insert(index, str2);
```

Parameters

- **str2:** string to be inserted.
- **index:** position of where to insert the new string

Return Type

- Reference to the str1.

Example

```
std::string text = "I have a cat.";
text.insert(9, " black"); // Insert " black" at position 9
```

2. replace()

The replace() function replaces the part of the string with the given other string. Unlike insert, the characters in the part where the new string is to be inserted are removed.

Syntax

```
str1.replace(index, size, str2);
```

Parameters

- **index:** Index of where to start replacing the new string.
- **size:** length of the part of the string that is to be replaced.
- **str2:** new string that is to be inserted.

Return Type

- Reference to the str1.

Example

```
std::string text = "I like dogs.";
text.replace(7, 4, "cats"); // Replace "dogs" with "cats"
```

3. erase()

The erase() function is a member function of std::string class that is used to remove a character or a part of the string.

Syntax

```
str1.erase(start, end);
```

Parameters

- **start:** Starting position.
- **end:** Ending position.

Return Type

- Reference to the str1.

Example

```
std::string text = "This is an example.";
text.erase(5, 3); // Erase "is "
```

Convert std::string to C String – c_str)_

The c_str() function is a member function that is used to convert the C++ style string i.e. std::string objects to C style string i.e. array of characters.

Syntax

```
str1.c_str()
```

Parameters

- This function does not take any parameter.

Return Value

- Pointer to the equivalent array of characters.

Example

```
std::string str = "C++";
const char* cstr = str.c_str()
```

Example of String Functions in C++

The below code demonstrate the use of the above specified string functions:

C++

```
// C++ Code to demostrate various functions available in
// String class

#include <iostream>
#include <string>

using namespace std;

int main()
{
    // Creating and initializing strings
    string greeting = "Hello, World!";
    cout << greeting << endl;
    string name;

    // Input from the user
    cout << "Enter your name: ";
    cin >> name;
    cout << name << endl;

    // String length
    int length = greeting.length();
    cout << length << endl;

    // Accessing characters
    char firstChar = greeting[0];
    char secondChar = greeting.at(1);
```

```
cout << firstChar << " " << secondChar << endl;

// Appending and concatenating strings
string firstName = "Geek";
string lastName = "Geeks";
string fullName = firstName + " " + lastName;
cout << fullName << endl;
string base = "Hello";
cout << base << endl;
base.append(" World!");
cout << base << endl;

// String comparison
string str1 = "apple";
string str2 = "banana";
if (str1 == str2) {
    cout << "Strings are equal" << endl;
}
else {
    cout << "Strings are not equal" << endl;
}

int result = str1.compare(str2);
if (result == 0) {
    cout << "Strings are equal" << endl;
}
else if (result < 0) {
    cout << "str1 comes before str2" << endl;
}
else {
    cout << "str1 comes after str2" << endl;
}

// Substrings
string text = "Hello, World!";
cout << text << endl;
string sub = text.substr(7, 5);
cout << sub << endl;

// Searching
string searchIn = "C++ Programming";
size_t position = searchIn.find("Programming");
if (position != string::npos) {
    cout << "Found at position " << position << endl;
}
else {
    cout << "Not found" << endl;
}
```

```
    // Modifying strings
    string modify = "I like dogs.";
    modify.replace(7, 4, "cats");
    cout << modify << endl;
    modify.insert(6, " black");
    cout << modify << endl;
    modify.erase(6, 6);
    cout << modify << endl;

    // Conversion
    string str = "C++";
    const char* cstr = str.c_str();
    cout << cstr << endl;

    return 0;
}
```

Output

```
Hello, World!
Enter your name: Geeks
Geeks
13
H e
Geek Geeks
Hello
Hello World!
Strings are not equal
str1 comes before str2
Hello, World!
World
Found at position 4
I like cats.
I like black cats.
I like cats.
C++
```

Whether you're preparing for your first job interview or aiming to upskill in this ever-evolving tech landscape, GeeksforGeeks Courses are your key to success. We provide top-quality content at affordable prices, all geared towards accelerating your growth in a time-bound manner. Join the millions we've