Strings in C++

Last Updated : 20 Sep, 2025

Strings in C++ are objects of the std::string class. They are used to represent and manipulate sequences of characters.

- Unlike C-style character arrays (char[]), std::string handles memory management automatically and provides a wide range of built-in functions for ease of use.
- Can automatically grow and shrink as you add or remove characters, unlike fixed-size C-style strings.
- You can easily access characters, join strings, compare them, extract substrings, and search within strings using built-in functions.

C++

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

int main() {

    // Creating a string
    string str = "Hello Geeks";

    // Traversing using index
    cout << "Using index: ";
    for (int i = 0; i < str.size(); i++) {
        cout << str[i] ;
    }
    cout << endl;

    // Traversing using range-based for loop
    cout << "Using range-based for loop: ";
    for (char ch : str) {
        cout << ch ;
    }
    cout << endl;

    // Traversing using iterator
    cout << "Using iterator: ";
    for (auto it = str.begin(); it != str.end(); it++) {
        cout << *it ;
    }
    cout << endl;

    return 0;
}
```

Output

```
Using index: Hello Geeks

Using range-based for loop: Hello Geeks

Using iterator: Hello Geeks
```

Syntax

The string container is defined as std::string class inside the <string> header file.

string str;
where,

- **string:** Class provided by STL to handle sequences of characters.
- **str:** Name assigned to the string object.

Basic Operations in String

Basic operations on Strings are shown below:

Initializing a String

- Initialization of a string assigns characters to the string at the time of creation.
- A string can be initialized directly using = or constructor syntax with text inside quotes.

C++

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

int main() {

    // Initializing a string directly
    string str = "Hello Geeks";
    // Printing the string
    cout << str << endl;

    return 0;
}
```

Output

```
Hello Geeks
```

Accessing Characters

- Characters of a string can be accessed using the [] operator or the .at() function.
- Time complexity for accessing characters is O(1).

C++

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

int main() {
    string str = "Hello Geeks";

    // Access using index operator []
    cout << "First character: " << str[0] << endl;
    cout << "Fifth character: " << str[4] << endl;

    // Access using at()
    cout << "Character at index 6: " << str.at(6) << endl;

    return 0;
}
```

Output

```
First character: H

Fifth character: o

Character at index 6: G
```

String Length

- The number of characters in a string can be found using size() or length().
- Time complexity to find string length is O(1).

C++

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

int main() {
    string str = "Hello Geeks";

    // Using size()
    cout << "Length using size(): " << str.size() << endl;

    // Using length()
    cout << "Length using length(): " << str.length() << endl;

    return 0;
}
```

Concatenation of Strings

- Strings can be joined using the + operator or the append() function.
- The + operator creates a new string, while append() modifies the existing string in place.
- Time complexity for concatenation is O(n+m), where n is the size of string and m is the size of the string to be concatenated.

C++

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

int main() {
    string str1 = "Hello";
    string str2 = " Geeks";

    // Using + operator
    string result1 = str1 + str2;
    cout << "Concatenation using + : " << result1 << endl;

    // Using append() function
    string result2 = str1;
    result2.append(str2);
    cout << "Concatenation using append(): " << result2 << endl;

    return 0;
}
```

Output

```
Concatenation using + : Hello Geeks

Concatenation using append(): Hello Geeks
```

Modifying a String

- Characters of a string can be added with .push_back(), removed with .pop_back(), or altered using .insert() and .erase().
- Time complexity for push/pop is O(1) and O(n) for insert/erase.

C++

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

int main() {
    string str = "Hello Geeks";

    // Adding a character at the end
    str.push_back('!');
    cout << "After push_back: " << str << endl;

    // Removing the last character
    str.pop_back();
    cout << "After pop_back: " << str << endl;

    // Inserting a substring
    str.insert(5, " C++");
    cout << "After insert: " << str << endl;

    // Erasing part of the string
    str.erase(5, 4);
    cout << "After erase: " << str << endl;

    return 0;
}
```

Output

```
After push_back: Hello Geeks!

After pop_back: Hello Geeks

After insert: Hello C++ Geeks

After erase: Hello Geeks
```

Substring Extraction

- The .substr(pos,len) is used to extract a part of a string, where pos means the starting position and len means how many characters you want to copy.
- This function creates a new string containing the selected portion, starting at pos and copying len characters.
- Time complexity of extraction is O(len).

C++

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

int main() {
    string str = "Hello Geeks";

    // Extract "Hello"
    string sub1 = str.substr(0, 5);
    cout << "Substring 1: " << sub1 << endl;

    // Extract "Geeks"
    string sub2 = str.substr(6, 5);
    cout << "Substring 2: " << sub2 << endl;

    return 0;
}
```

Output

```
Substring 1: Hello

Substring 2: Geeks
```

Know more about Substring extraction in C++.

Searching in a String

- The find() function is used to search for a substring inside a string. If found, it returns the index (position) where the substring starts; if not, it returns a special value (npos).
- Time complexity to search is O(n*m), where n is the length of string and m is the substring length.

C++

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

int main() {
    string str = "Hello Geeks";

    size_t pos = str.find("Geeks");

    if (pos < str.size()) {
        cout << "\"Geeks\" found at index: " << pos << endl;
    }

    return 0;
}
```

Output

```
"Geeks" found at index: 6
```

Learn more about find() in C++