std::string class in C++

Difficulty Level : Easy • Last Updated : 17 Feb, 2023

C++ has in its definition a way to represent a **sequence of characters as an object of the class**. This class is called std:: string. The string class stores the characters as a sequence of bytes with the functionality of allowing **access to the single-byte character**.

String vs Character Array

String	Char Array
A string is a **class that defines objects** that be represented as a stream of characters.	A character array is simply an **array of characters** that can be terminated by a null character.
In the case of strings, memory is **allocated dynamically**. More memory can be allocated at run time on demand. As no memory is preallocated, **no memory is wasted**.	The size of the character array has to be **allocated statically**, more memory cannot be allocated at run time if required. Unused allocated **memory is also wasted**
As strings are represented as objects, **no array decay** occurs.	There is a **threat of array decay** in the case of the character array.
Strings are slower when compared to implementation than character array.	Implementation of **character array is faster** than std:: string.
String class defines **a number of functionalities** that allow manifold	Character arrays **do not offer** many **inbuilt functions** to manipulate strings.

String	Char Array
operations on strings.	

Operations on Strings

1) Input Functions

Function	Definition
getline()	This function is used to store a stream of characters as entered by the user in the object memory.
push_back()	This function is used to input a character at the end of the string.
pop_back()	Introduced from C++11 (for strings), this function is used to delete the last character from the string.

Example:

CPP

```
// C++ Program to demonstrate the working of
// getline(), push_back() and pop_back()
#include <iostream>
#include <string> // for string class
using namespace std;

// Driver Code
int main()
{
    // Declaring string
    string str;

    // Taking string input using getline()
```

```
    getline(cin, str);

    // Displaying string
    cout << "The initial string is : ";
    cout << str << endl;

    // Inserting a character
    str.push_back('s');

    // Displaying string
    cout << "The string after push_back operation is : ";
    cout << str << endl;

    // Deleting a character
    str.pop_back();

    // Displaying string
    cout << "The string after pop_back operation is : ";
    cout << str << endl;

    return 0;
}
```

Output

```
The initial string is :
The string after push_back operation is : s
The string after pop_back operation is :
```

Time Complexity: O(1)

Space Complexity: O(n) where n is the size of the string

2) Capacity Functions

Function	Definition
capacity()	This function returns the capacity allocated to the string, which can be equal to or more than the size of the string. Additional space is allocated so that when the new characters are added to the string, the operations can be done efficiently.
resize()	This function changes the size of the string, the size can be increased or decreased.
length()	This function finds the length of the string.

Function	Definition
shrink_to_fit()	This function decreases the capacity of the string and makes it equal to the minimum capacity of the string. This operation is useful to save additional memory if we are sure that no further addition of characters has to be made.

Example:

CPP

```
// C++ Program to demonstrate the working of
// capacity(), resize() and shrink_to_fit()
#include <iostream>
#include <string> // for string class
using namespace std;

// Driver Code
int main()
{
    // Initializing string
    string str = "geeksforgeeks is for geeks";

    // Displaying string
    cout << "The initial string is : ";
    cout << str << endl;

    // Resizing string using resize()
    str.resize(13);

    // Displaying string
    cout << "The string after resize operation is : ";
    cout << str << endl;

    // Displaying capacity of string
    cout << "The capacity of string is : ";
    cout << str.capacity() << endl;

    // Displaying length of the string
    cout << "The length of the string is :" << str.length()
         << endl;

    // Decreasing the capacity of string
    // using shrink_to_fit()
    str.shrink_to_fit();

    // Displaying string
    cout << "The new capacity after shrinking is : ";
    cout << str.capacity() << endl;

    return 0;
}
```

Output

```
The initial string is : geeksforgeeks is for geeks
The string after resize operation is : geeksforgeeks
The capacity of string is : 26
The length of the string is :13
The new capacity after shrinking is : 13
```

Time Complexity: O(1)

Space Complexity: O(n) where n is the size of the string

3) Iterator Functions

Function	Definition
begin()	This function returns an iterator to the beginning of the string.
end()	This function returns an iterator to the next to the end of the string.
rbegin()	This function returns a reverse iterator pointing at the end of the string.
rend()	This function returns a reverse iterator pointing to the previous of beginning of the string.
cbegin()	This function returns a constant iterator pointing to the beginning of the string, it cannot be used to modify the contents it points-to.
cend()	This function returns a constant iterator pointing to the next of end of the string, it cannot be used to modify the contents it points-to.
crbegin()	This function returns a constant reverse iterator pointing to the end of the string, it cannot be used to modify the contents it points-to.
crend()	This function returns a constant reverse iterator pointing to the previous of beginning of the string, it cannot be used to modify the contents it points-to.

Algorithm:

1. *Declare a string*
2. *Try to iterate the string using all types of iterators*
3. *Try modification of the element of the string.*
4. *Display all the iterations.*

Example:

CPP

```
// C++ Program to demonstrate the working of
// begin(), end(), rbegin(), rend(), cbegin(), cend(), crbegin(), crend()
#include <iostream>
#include <string> // for string class
using namespace std;

// Driver Code
int main()
{
    // Initializing string`
    string str = "geeksforgeeks";

    // Declaring iterator
    std::string::iterator it;

    // Declaring reverse iterator
    std::string::reverse_iterator it1;
    cout<<"Str:"<<str<<"\n";
    // Displaying string
    cout << "The string using forward iterators is : ";
    for (it = str.begin(); it != str.end(); it++){
        if(it == str.begin()) *it='G';
        cout << *it;
    }
    cout << endl;

      str = "geeksforgeeks";
    // Displaying reverse string
    cout << "The reverse string using reverse iterators is "
            ": ";
    for (it1 = str.rbegin(); it1 != str.rend(); it1++){
        if(it1 == str.rbegin()) *it1='S';
        cout << *it1;
    }
    cout << endl;

  str = "geeksforgeeks";
  //Displaying String
  cout<<"The string using constant forward iterator is :";
  for(auto it2 = str.cbegin(); it2!=str.cend(); it2++){
        //if(it2 == str.cbegin()) *it2='G';
        //here modification is NOT Possible
```

```
        //error: assignment of read-only location
        //As it is a pointer to the const content, but we can inc/dec-rement the itera
        cout<<*it2;
    }
    cout<<"\n";

    str = "geeksforgeeks";
    //Displaying String in reverse
    cout<<"The reverse string using constant reverse iterator is :";
    for(auto it3 = str.crbegin(); it3!=str.crend(); it3++){
        //if(it2 == str.cbegin()) *it2='S';
        //here modification is NOT Possible
        //error: assignment of read-only location
        //As it is a pointer to the const content, but we can inc/dec-rement the itera
        cout<<*it3;
    }
    cout<<"\n";

    return 0;
}

//Code modified by Balakrishnan R (rbkraj000)
```

Output

```
Str:geeksforgeeks
The string using forward iterators is : Geeksforgeeks
The reverse string using reverse iterators is : Skeegrofskeeg
The string using constant forward iterator is :geeksforgeeks
The reverse string using constant reverse iterator is :skeegrofskeeg
```

Time Complexity: O(1)

Space Complexity: O(n) where n is the size of the string

4) Manipulating Functions:

Function	Definition
copy("char array", len, pos)	This function copies the substring in the target character array mentioned in its arguments. It takes 3 arguments, target char array, length to be copied, and starting position in the string to start copying.
swap()	This function swaps one string with another

Example:

CPP

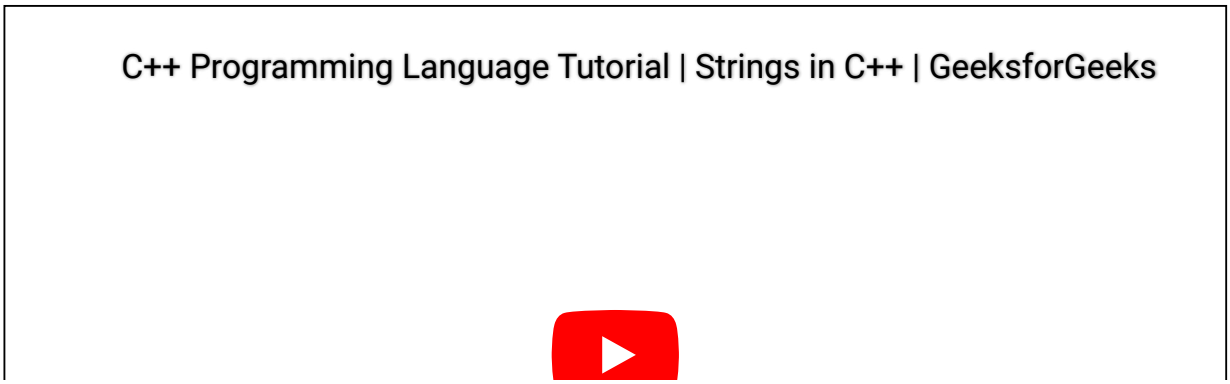
```cpp
// C++ Program to demonstrate the working of
// copy() and swap()
#include <iostream>
#include <string> // for string class
using namespace std;

// Driver Code
int main()
{
    // Initializing 1st string
    string str1 = "geeksforgeeks is for geeks";

    // Declaring 2nd string
    string str2 = "geeksforgeeks rocks";

    // Declaring character array
    char ch[80];

    // using copy() to copy elements into char array
    // copies "geeksforgeeks"
    str1.copy(ch, 13, 0);

    // Displaying char array
    cout << "The new copied character array is : ";
    cout << ch << endl;

    // Displaying strings before swapping
    cout << "The 1st string before swapping is : ";
    cout << str1 << endl;
    cout << "The 2nd string before swapping is : ";
    cout << str2 << endl;

    // using swap() to swap string content
    str1.swap(str2);

    // Displaying strings after swapping
    cout << "The 1st string after swapping is : ";
    cout << str1 << endl;
    cout << "The 2nd string after swapping is : ";
    cout << str2 << endl;

    return 0;
}
```

Output

```
The new copied character array is : geeksforgeeks
The 1st string before swapping is : geeksforgeeks is for geeks
The 2nd string before swapping is : geeksforgeeks rocks
```

```
The 1st string after swapping is : geeksforgeeks rocks
The 2nd string after swapping is : geeksforgeeks is for geeks
```

Must Read: C++ String Class and its Applications



This article is contributed by **Manjeet Singh**. If you like GeeksforGeeks and would like to contribute, you can also write an article using write.geeksforgeeks.org or mail your article to review-team@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks. Please write comments if you find anything incorrect, or if you want to share more information about the topic discussed above.

480

Related Articles

1. Behavior of virtual function in the derived class from the base class and abstract class
2. How to convert a class to another class type in C++?
3. Base Class Pointer Pointing to Derived Class Object in C++
4. Difference between Base class and Derived class in C++
5. Can a C++ class have an object of self type?
6. Hiding of all Overloaded Methods with Same Name in Base Class in C++
7. Why is the Size of an Empty Class Not Zero in C++?
8. Simulating final Class in C++
9. What happens when more restrictive access is given to a derived class method in C++?