# std::string class in C++

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

std:: string vs Character Array

* A character array is simply an array of characters can terminated by a null character. A string is a class which defines objects that be represented as stream of characters.

* Size of the character array has to allocated statically, more memory cannot be allocated at run time if required. Unused allocated memory is wasted in case of character array. In 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.

* There is a threat of[array decay](https://www.geeksforgeeks.org/what-is-array-decay-in-c-how-can-it-be-prevented/) in case of character array. As strings are represented as objects, no array decay occurs.

* Implementation of character array is faster than std:: string. Strings are slower when compared to implementation than character array.

* Character array do not offer much inbuilt functions to manipulate strings. String class defines a number of functionalities which allow manifold operations on strings.

Operations on strings

Input Functions

1. getline() :- This function is used to store a stream of characters as entered by the user in the object memory.

2. push\_back() :- This function is used to input a character at the end of the string.

3. pop\_back() :- Introduced from C++11(for strings), this function is used to delete the last character from the string.

|  |
| --- |
| // C++ code to demonstrate the working of  // getline(), push\_back() and pop\_back()  #include<iostream>  #include<string> // for string class  **using** **namespace** std;  **int** main()  {      // Declaring string      string str;        // Taking string input using getline()      // "geeksforgeek" in givin output      getline(cin,str);        // Displaying string      cout << "The initial string is : ";      cout << str << endl;        // Using push\_back() to insert a character      // at end      // pushes 's' in this case      str.push\_back('s');        // Displaying string      cout << "The string after push\_back operation is : ";      cout << str << endl;        // Using pop\_back() to delete a character      // from end      // pops 's' in this case      str.pop\_back();        // Displaying string      cout << "The string after pop\_back operation is : ";      cout << str << endl;    **return** 0;    } |

Input:

geeksforgeek

Output:

The initial string is : geeksforgeek

The string after push\_back operation is : geeksforgeeks

The string after pop\_back operation is : geeksforgeek

Capacity Functions

4. 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.

5. resize() :- This function changes the size of string, the size can be increased or decreased.

6.length():-This function finds the length of the string

7.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 have to be made.

filter\_none

edit

play\_arrow

brightness\_4

|  |
| --- |
| // C++ code to demonstrate the working of  // capacity(), resize() and shrink\_to\_fit()  #include<iostream>  #include<string> // for string class  **using** **namespace** std;  **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

Iterator Functions

8. begin() :- This function returns an iterator to beginning of the string.

9. end() :- This function returns an iterator to end of the string.

10. rbegin() :- This function returns a reverse iterator pointing at the end of string.

11. rend() :- This function returns a reverse iterator pointing at beginning of string.

|  |
| --- |
| // C++ code to demonstrate the working of  // begin(), end(), rbegin(), rend()  #include<iostream>  #include<string> // for string class  **using** **namespace** std;  **int** main()  {      // Initializing string`      string str = "geeksforgeeks";        // Declaring iterator      std::string::iterator it;        // Declaring reverse iterator      std::string::reverse\_iterator it1;        // Displaying string      cout << "The string using forward iterators is : ";  **for** (it=str.begin(); it!=str.end(); it++)      cout << \*it;      cout << endl;        // Displaying reverse string      cout << "The reverse string using reverse iterators is : ";  **for** (it1=str.rbegin(); it1!=str.rend(); it1++)      cout << \*it1;      cout << endl;    **return** 0;    } |

Output:

The string using forward iterators is : geeksforgeeks

The reverse string using reverse iterators is : skeegrofskeeg

Manipulating Functions

12. copy(“char array”, len, pos) :- This function copies the substring in target character array mentioned in its arguments. It takes 3 arguments, target char array, length to be copied and starting position in string to start copying.

13. swap() :- This function swaps one string with other.

|  |
| --- |
| // C++ code to demonstrate the working of  // copy() and swap()  #include<iostream>  #include<string> // for string class  **using** **namespace** std;  **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);        // Diplaying char array      cout << "The new copied character array is : ";      cout << ch << endl << 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