# STL-Strings

To use strings in a program, you need to include a header called string.
For example:

```
#include <string>
```

**Declaring a string**

```
string str = "rishabh";
```

It declares a string of value “rishabh”

```
string str(10);
```

It declares a string of size 10.

```
string s(5, 'N');
```

It declares a string of size 5 with all characters ‘N’.

```
string abc(str);
```

It declares a copy of the string str.

**Taking Input**

We use cin to input the string.

*cin >> str;*

Using getline() function: To input the string with space we use getline() function instead of cin.

```
string s;
getline(cin, s);

cout << "Rishabh is " << s << endl;
```

Input: *a peaceful soul*

Output: *Rishabh is a peaceful soul*

**Throwing Output**

We use cout to throw output to the terminal.

*cout << str;*

**Different functions of string**

1. append(): Inserts additional characters at the end of the string (can also be done using '+' or '+=' operator). Its time complexity is O(N) where N is the size of the new string.

```
string s1 = "fam", s2 = "ily";
s1.append(s2);
cout << s1 << endl;
```

```
string s1 = "fam", s2 = "ily";
s1 = s1 + s2;
cout << s1 << endl;
```

Output: *family*

2. assign(): Assigns new string by replacing the previous value (can also be done using '=' operator).

```
string s = "nincompoop";
s.assign("Rishabh");
cout << s << endl;
```

```
string s = "nincompoop";
s = "Rishabh";
cout << s << endl;
```

Output: *Rishabh*

3. at(): Returns the character at a particular position (can also be done using '[ ]' operator). Its time complexity is O(1).

```
string s = "nincompoop";
cout << s.at(3) << endl;
cout << s[3] << endl;
```

Output: *c*

*c*

4. begin(): Returns an iterator pointing to the first character. Its time complexity is O(1).

5. clear(): Erases all the contents of the string and assign an empty string ("") of length zero. Its time complexity is O(1).

```
string s = "nincompoop";
s.clear();
cout << s << endl;
```

No Output, as string is empty.

6. compare(): Compares the value of the string with the string passed in the parameter and returns an integer accordingly. Its time complexity is O(N + M) where N is the size of the first string and M is the size of the second string.

```
string s1 = "abc", s2 = "xyz";
cout << s2.compare(s1) << endl;
```

Output: *1513239* - basically a value greater than 0 denoting s2 is greater than s1.

```
string s1 = "abc", s2 = "abc";
cout << s1.compare(s2) << endl;
```

Output: *0* - as both string are equal.

```
string s1 = "xyz", s2 = "abc";
cout << s2.compare(s1) << endl;
```

Output: *-1513239* - basically a value less than 0 denoting s2 is less than s1.

7. c_str(): Convert the string into C-style string (null terminated string) and returns the pointer to the C-style string. Its time complexity is O(1).

8. empty(): Returns a boolean value, true if the string is empty and false if the string is not empty. Its time complexity is O(1).

```
string s = "nincompoop";
s.clear();
if(s.empty())
    cout << "given string is empty" << endl;
```

Output: *given string is empty*

9. end(): Returns an iterator pointing to a position which is next to the last character. Its time complexity is O(1).

10. erase(): Deletes a substring of the string. Its time complexity is O(N) where N is the size of the new string.

```
string s = "nincompoop";
s.erase(3,3);
cout << s << endl;
```

Output: *ninpoop*

11. find(): Searches the string and returns the first occurrence of the parameter in the string. Its time complexity is O(N) where N is the size of the string.

```
string s = "nincompoop";
cout << s.find("com") << endl;
```

Output: *3*

12. insert(): Inserts additional characters into the string at a particular position. Its time complexity is O(N) where N is the size of the new string.

```
string s = "nincompoop";
s.insert(2, "lol");
cout << s << endl;
```

Output: *nilolncompoop*

13. length(): Returns the length of the string. Its time complexity is O(1).

```
string s = "nincompoop";
cout << s.length() << endl;
```

Output: *10*

14. resize(): Resize the string to the new length which can be less than or greater than the current length. Its time complexity is O(N) where N is the size of the new string.

```
string s = "nincompoop";
s.resize(6);
cout << s << endl;
```

Output: *nincom*

15. size(): Returns the length of the string. Its time complexity is O(1).

```
string s = "nincompoop";
cout << s.size() << endl;
```

Output: *10*

16.substr(): Returns a string which is the copy of the substring. Its time complexity is O(N) where N is the size of the substring. For example:

```
string s = "nincompoop";
cout << s.substr(3, 4);
```

Output: *comp*

17.stoi(): Returns the strings converted to int datatype.

```
string s = "786";
int x = stoi(s);
cout << x + 2 << endl;
```

Output: *788*

Note:

1. To convert an integer to a string, we use to_string() function. Example

```
int x = 786;
cout << to_string(x) + "2" << endl;
```

Output: *7682*.

2. Sorting a string: To sort a string, we need to include a header file known as algorithm in our code like this

```
#include <algorithm>
```

Then we can use sort() function that is present in above header file on our string. Sort() function takes 2 arguments viz. iterator to start of the string and iterator to end of the string.

```
string s = "xcmnzdsfka";
sort(s.begin(), s.end());
cout << s << endl;
```

Output: *acdfkmnsxz*

Keep coding and improve your coding skills!