

* For pro feeling, see in dark mode.

STRINGS

Character Array

- Need to know size before hand
- Larger size required for operations (concatenate or append)
- No terminating extra character

String

- No need to know size before hand
- Performing operations like concatenation and append is just easier.
- Terminated with a special character %!

To use Strings, we need to include a header called string.

#include<string>

Declaring a string :-

String str = "manmay2"; // declares a string of value manmay2.

String str(10); // declares a string of size 10.

String s(5, 'N'); // declares a string of size 5 with all characters

String abc(str); // declares a copy of the string str.

Taking Inputs :-

- (i) `cin >> str;` // Using 'cin' function to input a string.
- (ii) `String s;`
`getline(cin, s);` // To input a string with space.

Functions On Strings :-

(i) append() → Inserts additional characters at the end of the string.
 It can also be done via concatenation ('+').

Time Complexity :- $O(N)$, where 'N' is the size of new string.

String `S1 = "Germany"`, `S2 = "many"`;

`S1.append(S2);`

Or, `S1 = S1 + S2;`

`Cout << S1 << endl;` // Germany

(ii) assign() :- Assigns new string by replacing the previous value.

It can also be done using ('=') .

String `S = "India"`;

`S.assign("Germany");`

Or, `S = "Germany";`

`Cout << S << endl;` // Germany

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

String `S = "Germany"`;

`Cout << S.at(3) << endl;`

Or, `Cout << S[3] << endl;` //m

- (iv) `begin()` :- Returns an iterator pointing to the first character.
It's time complexity is $O(1)$.
- (v) `clear()` :- Erases all the contents from the string and assigns an empty string ("") of length zero. Its time complexity is $O(1)$.
- (vi) `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 $S_1 = "abc"$, $S_2 = "xyz"$;
`cout << S_2.Compare(S1) << endl;`
- Output : 1513239 - basically a value greater than 0 denoting S_2 is greater than S_1 .
- (vii) `C-str()` :- Converts the string into C-style string (null terminated string) and returns the pointer to the C-style string.
It time complexity is $\rightarrow O(1)$.
- (viii) `empty()` :- It basically returning a boolean value, true if string is empty and false if the string is not empty.
Its time complexity is $O(1)$.
 String $s = ""$;
`If (s.empty()) {`
 `cout << "True" ;`
- (ix) `end()` :- Returns an iterator pointing to a position which is next to the last character. Its time complexity is $O(1)$.

(x) `erase()` :> Delete a substring of the string. Its time complexity is $O(N)$ where N is the size of the new string.

String $s = "Germany hello Europe"$

$s.erase(7, 6);$

cout << s; // Germany Europe

(xi) `find()` :> Searches the string and returning the first occurrence of the parameter in the string. Its time complexity is $O(N)$ where N is the size of the string.

String $s = "Germany is my destination";$

cout << s.find("Ge") << endl; // 0

(xii) `insert()` :> inserts an additional characters into the string at a particular position. Its time complexity is $O(N)$ where N is the size of string.

String $s = "Germany";$

$s.insert(7, "Europe");$

cout << s; // GermanyEurope

(xiii) `length() / size()` :> Returns the length of the string.
Time complexity for both is $O(1)$.

(xiv) `resize()` :> Resize the string to a new length which can be less than or greater than the current length. Its time complexity is $O(N)$.

$s.resize(6);$

(xv) `substr()` :> Returns a string which is the copy of the substring. Its time complexity is $O(N)$.

String $s = "Germany";$

$s.substr(3, 4);$

cout << s; // many

(xvi) stoi() :- Returning the strings converted to int datatype.

String s = "789";

cout << stoi(s); // 789

(xvii) to_string() :- Use to convert integer to string

int x = 12;

cout << to_string(x); // 12

(xviii) Sorting a string :-

At first add a header →

#include <algorithm>

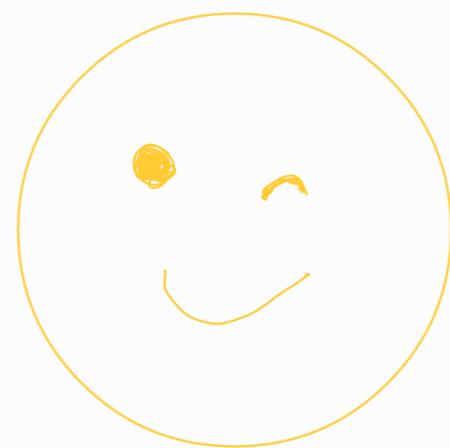
Then use sort() function that is present in above header file. Sort function takes 2 arguments viz. iterator to start of the string and iterator to end of the string.

String s = "Germany";

sort(s.begin(), s.end());

cout << s; // aeGmnrory

Keep coding and have a smile on your face.



!!!! KEEP LEARNING AND KEEP EXPLORING !!!!