* 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 '\0'.

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 N
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="Ger", 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()**: Returns the character at a particular position (can also be done using '[]' operator). It's 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 S1 = "abc", S2 = "xyz";
Cout << S2.compare(S1) << endl;
```

Output : 15 13239 – basically a value greater than 0 denoting S2 is greater than S1.

(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 → O(1).

(viii) empty() :- It basically returns a boolean value, true if string is empty and false if the string is not empty. It's 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. It's time complexity is O(1).

(x) erase() :→ Delete a substring of the string. It's 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 returns the first occurence 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("G") << 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. It's time complexity is O(N).

```
S.resize(6);
```

(xv) substr() :→ Returns a string which is the copy of the substring. It's time complexity is O(N).

```
String S = "Germany";
S.substr(3,4);
cout << s; // many
```

(xvi) Stoi() :- Returns 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; // aeGmnry
```

Keep coding and have a smile on your face.

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

☺ DREAMS ONN ☺