**Introduction to String**

[Report Issue](https://github.com/LeetCode-Feedback/LeetCode-Feedback/issues)

A string is actually an array of unicode characters. You can perform almost all the operations we used in an array. You can try it out by yourself.

However, there are some differences. In this article, we will go through some of them which you should be aware of when dealing with a string. These features might vary a lot from one language to another.

*Compare Function*

String has its own compare function (we will show you the usage of compare function in the code below).

However, there is a problem:

Can we use "==" to compare two strings?

It depends on the answer to the question:

Does the language support operator overloading?

1. If the answer is yes (like C++), we may use "==" to compare two strings.
2. If the answer is no (like Java), we may not use "==" to compare two strings. When we use "==", it actually compares whether these two objects are the same object.

Let's run the following example and compare the results:

**C++**

#include <iostream>

int main() {

string s1 = "Hello World";

cout << "s1 is \"Hello World\"" << endl;

string s2 = s1;

cout << "s2 is initialized by s1" << endl;

string s3(s1);

cout << "s3 is initialized by s1" << endl;

// compare by '=='

cout << "Compared by '==':" << endl;

cout << "s1 and \"Hello World\": " << (s1 == "Hello World") << endl;

cout << "s1 and s2: " << (s1 == s2) << endl;

cout << "s1 and s3: " << (s1 == s3) << endl;

// compare by 'compare'

cout << "Compared by 'compare':" << endl;

cout << "s1 and \"Hello World\": " << !s1.compare("Hello World") << endl;

cout << "s1 and s2: " << !s1.compare(s2) << endl;

cout << "s1 and s3: " << !s1.compare(s3) << endl;

}

**Java**

// "static void main" must be defined in a public class.

public class Main {

public static void main(String[] args) {

// initialize

String s1 = "Hello World";

System.out.println("s1 is \"" + s1 + "\"");

String s2 = s1;

System.out.println("s2 is another reference to s1.");

String s3 = new String(s1);

System.out.println("s3 is a copy of s1.");

// compare using '=='

System.out.println("Compared by '==':");

// true since string is immutable and s1 is binded to "Hello World"

System.out.println("s1 and \"Hello World\": " + (s1 == "Hello World"));

// true since s1 and s2 is the reference of the same object

System.out.println("s1 and s2: " + (s1 == s2));

// false since s3 is refered to another new object

System.out.println("s1 and s3: " + (s1 == s3));

// compare using 'equals'

System.out.println("Compared by 'equals':");

System.out.println("s1 and \"Hello World\": " + s1.equals("Hello World"));

System.out.println("s1 and s2: " + s1.equals(s2));

System.out.println("s1 and s3: " + s1.equals(s3));

// compare using 'compareTo'

System.out.println("Compared by 'compareTo':");

System.out.println("s1 and \"Hello World\": " + (s1.compareTo("Hello World") == 0));

System.out.println("s1 and s2: " + (s1.compareTo(s2) == 0));

System.out.println("s1 and s3: " + (s1.compareTo(s3) == 0));

}

}

*Immutable or Mutable*

Immutable means that you can't change the content of the string once it's initialized.

1. In some languages (like C++), the string is mutable. That is to say, you can modify the string just like what you did in an array.
2. In some other languages (like Java), the string is immutable. This feature will bring several problems. We will illustrate the problems and solutions in the next article.

You can determine whether the string in your favorite language is immutable or mutable by testing the modification operation. Here is an example:

**C++**

#include <iostream>

int main() {

string s1 = "Hello World";

s1[5] = ',';

cout << s1 << endl;

}

Java

// "static void main" must be defined in a public class.

public class Main {

public static void main(String[] args) {

String s1 = "Hello World";

s1[5] = ',';

System.out.println(s1);

}

}

*Extra Operations*

Compare to an array, there are some extra operations we can perform on a string. Here are some examples:

**C++**

#include <iostream>

int main() {

string s1 = "Hello World";

// 1. concatenate

s1 += "!";

cout << s1 << endl;

// 2. find

cout << "The position of first 'o' is: " << s1.find('o') << endl;

cout << "The position of last 'o' is: " << s1.rfind('o') << endl;

// 3. get substr

cout << s1.substr(6, 5) << endl;

}

**Java**

// "static void main" must be defined in a public class.

public class Main {

public static void main(String[] args) {

String s1 = "Hello World";

// 1. concatenate

s1 += "!";

System.out.println(s1);

// 2. find

System.out.println("The position of first 'o' is: " + s1.indexOf('o'));

System.out.println("The position of last 'o' is: " + s1.lastIndexOf('o'));

// 3. get substring

System.out.println(s1.substring(6, 11));

}

}

You should be aware of the time complexity of these built-in operations.

For instance, if the length of the string is N, the time complexity of both finding operation and substring operation is O(N).

Also, in languages which the string is immutable, you should be careful with the concatenation operation (we will explain this in next article as well).

Never forget to take the time complexity of built-in operations into consideration when you compute the time complexity for your solution.