Introduction to Strings

Strings are defined as a stream of characters. Strings are used to represent text and are generally represented by enclosing text within quotes as: *"This is a sample string!"*.

Different programming languages have different ways of declaring and using Strings. We will learn to implement strings in **C/C++ and Java**.

Strings in C/C++

In C/C++, Strings are defined as an array of characters. The difference between a character array and a string is that the string is terminated with a special character '\0'.

Declaring Strings: Declaring a string is as simple as declaring a one-dimensional array. Below is the basic syntax for declaring a string.

```
char str_name[size];
```

In the above syntax, *str_name* is any name given to the string variable and size is used to define the length of the string, i.e the number of characters strings will store. Please keep in mind that there is an extra terminating character which is the Null character ('\0') used to indicate the termination of string which differs strings from normal character arrays.

Initializing a String: A string can be initialized in different ways. We will explain this with the help of an example. Below is an example to declare a string with the name as *str* and initialize it with **"GeeksforGeeks"**.

1. char str[] = "GeeksforGeeks";
2. char str[50] = "GeeksforGeeks";
3. char str[] = {'G','e','e','k','s','f','o','r','G','e','e','k','s','\0'};
4. char str[14] = {'G','e','e','k','s','f','o','r','G','e','e','k','s','\0'};

Printing a string array: Unlike arrays we do not need to print a string, character by character. The C/C++ language does not provide an inbuilt data type for strings but it has an access specifier "**%s**" which can be used to directly print and read strings.

```
1
2  // C/C++ program to illustrate strings
3
4  #include<bits/stdc++.h>
5
6  int main()
7  {
8      // declare and initialize string
9      char str[] = "Geeks";
10
11     // print string
12     printf("%s",str);
13
14     return 0;
15 }
16
```

Output:

```
Geeks
```

Passing strings to function: As strings are character arrays, so we can pass strings to function in the same way we pass an array to a function. Below is a sample program to do this:

```
// C/C++ program to illustrate how to
// pass strings to function

#include<bits/stdc++.h>

void printStr(char str[])
{
    printf("String is : %s",str);
}

int main()
{
    // declare and initialize string
    char str[] = "GeeksforGeeks";

    // print string by passing string
    // to a different function
    printStr(str);

    return 0;
}
```

Output:

```
String is : GeeksforGeeks
```

std::string Class in C++

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

string Class 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.
- Implementation of character array is faster than std:: string. Strings are slower when compared to implementation than character array.
- Character array does not offer much inbuilt functions to manipulate strings. String class defines a number of functionalities which allow manifold operations on strings.

Declaration Syntax: Declaring string using string class is simple and can be done using the *string* keyword as shown below.

```
string string_name = "Sample String";
```

Sample Program:

```cpp

// C++ program to illustrate strings

#include<bits/stdc++.h>
using namespace std;

int main()
{
    // declare and initialize string
    string str = "Geeks";

    // print string
    cout<<str;

    return 0;
}

```

Output:

```
Geeks
```

To learn about std::string in details, refer: std::string class in C++.

Strings in Java

String is a sequence of characters. In java, objects of String are immutable which means a constant and cannot be changed once created.

Creating a String

There are two ways to create string in Java:

- *String literal*

```
String s = “GeeksforGeeks”;
```

- Using *new* keyword

```
String s = new String (“GeeksforGeeks”);
```

String Methods

1. **int length():** Returns the number of characters in the String.

```
"GeeksforGeeks".length();  // returns 13
```

2. **Char charAt(int i):** Returns the character at i^{th} index.

```
"GeeksforGeeks".charAt(3); // returns  ‘k’
```

3. **String substring (int i):** Return the substring from the i^{th} index character to end.

```
"GeeksforGeeks".substring(3); // returns “ksforGeeks”
```

4. **String substring (int i, int j):** Returns the substring from i to j-1 index.

```
"GeeksforGeeks".substring(2, 5); // returns “eks”
```

5. **String concat(String str):** Concatenates specified string to the end of this string.

```
String s1 = ”Geeks”;

String s2 = ”forGeeks”;

String output = s1.concat(s2); // returns “GeeksforGeeks”
```

6. **int indexOf (String s):** Returns the index within the string of the first occurrence of the specified string.

```
String s = ”Learn Share Learn”;

int output = s.indexOf(“Share”); // returns 6
```

7. **int indexOf (String s, int i):** Returns the index within the string of the first occurrence of the specified string, starting at the specified index.

```
String s = ”Learn Share Learn”;

int output = s.indexOf(‘a’,3);// returns 8
```

8. **Int lastIndexOf(String s):** Returns the index within the string of the last occurrence of the specified string.

```
String s = ”Learn Share Learn”;

int output = s.lastIndexOf(‘a’); // returns 14
```

9. **boolean equals(Object otherObj):** Compares this string to the specified object.

```
Boolean out = “Geeks”.equals(“Geeks”); // returns true

Boolean out = “Geeks”.equals(“geeks”); // returns false
```

10. **boolean equalsIgnoreCase (String anotherString):** Compares string to another string, ignoring case considerations.

```
Boolean out= “Geeks”.equalsIgnoreCase(“Geeks”); // returns true

Boolean out = “Geeks”.equalsIgnoreCase(“geeks”); // returns true
```

11. **int compareTo(String anotherString):** Compares two string lexicographically.

```
int out = s1.compareTo(s2);  // where s1 ans s2 are

                             // strings to be compared


This returns difference s1-s2. If :

out < 0  // s1 comes before s2

out = 0  // s1 and s2 are equal.

out > 0   // s1 comes after s2.
```

12. **int compareToIgnoreCase(String anotherString):** Compares two string lexicographically, ignoring case considerations.

```
 int out = s1.compareToIgnoreCase(s2);

// where s1 ans s2 are

// strings to be compared


 This returns difference s1-s2. If :

 out < 0  // s1 comes before s2

 out = 0   // s1 and s2 are equal.

 out > 0   // s1 comes after s2.
```

Note- In this case, it will not consider case of a letter (it will ignore whether it is uppercase or lowercase).

13. **String toLowerCase():** Converts all the characters in the String to lower case.

```
String word1 = “HeLLo”;

String word3 = word1.toLowerCase(); // returns “hello"
```

14. **String toUpperCase():** Converts all the characters in the String to upper case.

```
String word1 = "HeLLo";

String word2 = word1.toUpperCase(); // returns "HELLO"
```

15. **String trim():** Returns the copy of the String, by removing whitespaces at both ends. It does not affect whitespaces in the middle.

```
String word1 = " Learn Share Learn ";

String word2 = word1.trim(); // returns "Learn Share Learn"
```

16. **String replace (char oldChar, char newChar):** Returns new string by replacing all occurrences of *oldChar* with *newChar*.

```
String s1 = "feeksforfeeks";

String s2 = "feeksforfeeks".replace('f' ,'g'); // returns "geeksgorgeeks"
```

Note:- s1 is still feeksforfeeks and s2 is geeksgorgeeks

Program to illustrate all string methods:

```java
// Java code to illustrate different constructors and methods String class.
import java.io.*;
import java.util.*;
class Test
{
    public static void main (String[] args)
    {
        String s= "GeeksforGeeks";// or String s= new String ("GeeksforGeeks");
        // Returns the number of characters in the String.
        System.out.println("String length = " + s.length());
// Returns the character at ith index.
        System.out.println("Character at 3rd position = "+ s.charAt(3));
        // Return the substring from the ith index character to end of string
        System.out.println("Substring " + s.substring(3));
        // Returns the substring from i to j-1 index.
        System.out.println("Substring = " + s.substring(2,5));
        // Concatenates string2 to the end of string1.
        String s1 = "Geeks";
        String s2 = "forGeeks";
        System.out.println("Concatenated string = " +s1.concat(s2));
        // Returns the index within the string
        // of the first occurrence of the specified string.
        String s4 = "Learn Share Learn";
        System.out.println("Index of Share " + s4.indexOf("Share"));
```

```
        // Returns the index within the string of the
        // first occurrence of the specified string,
        // starting at the specified index.
        System.out.println("Index of a = " + s4.indexOf('a',3));
        // Checking equality of Strings
        Boolean out = "Geeks".equals("geeks");
        System.out.println("Checking Equality " + out);
        out = "Geeks".equals("Geeks");
        System.out.println("Checking Equality " + out);
        out = "Geeks".equalsIgnoreCase("gEeks ");
        System.out.println("Checking Equality " + out);
        int out1 = s1.compareTo(s2);
        System.out.println("If s1 = s2 " + out);
        // Converting cases
        String word1 = "GeeKyMe";
        System.out.println("Changing to lower Case " +word1.toLowerCase());
        // Converting cases
        String word2 = "GeekyME";
        System.out.println("Changing to UPPER Case " + wrd1.toUpperCase());
        // Trimming the word
        String word4 = " Learn Share Learn ";
        System.out.println("Trim the word " + word4.trim());
        // Replacing characters
        String str1 = "feeksforfeeks";
        System.out.println("Original String " + str1);
        String str2 = "feeksforfeeks".replace('f' ,'g') ;
        System.out.println("Replaced f with g -> " + str2);
    } }
```

Output :

```
String length = 13
Character at 3rd position = k
Substring ksforGeeks
Substring = eks
Concatenated string = GeeksforGeeks
Index of Share 6
Index of a = 8
Checking Equality false
Checking Equality true
Checking Equality false
If s1 = s2 false
Changing to lower Case geekyme
Changing to UPPER Case GEEKYME
Trim the word Learn Share Learn
Original String feeksforfeeks
Replaced f with g -> geeksgorgeeks
```