Strings in C++

- A **string** is an ordered sequence of characters, enclosed in double quotation marks. It is part of the Standard Library.
- You need to include the <**string**> library to use the string data type. Alternatively, you can use a library that includes the string library.

```
#include <string>
using namespace std;
int main() {
string a = "I am learning C++";
return 0;
}
```

- The <**string**> library is included in the <**iostream**> library, so you don't need to include <string> separately, if you already use <**iostream**>.

The C-Style Character String

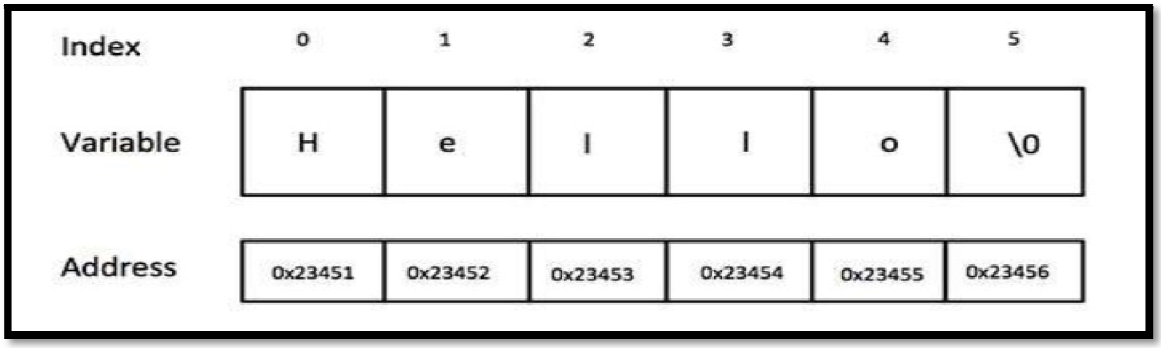
- The C-style character string originated within the C language and continues to be supported within C++.
- This string is actually a one-dimensional array of characters which is terminated by a **null** character '\0'.
- The following declaration and initialization create a string consisting of the word "Hello".

```
char greeting[6] = {'H', 'e', 'l', 'l', 'o', '\0'};
```

- To hold the null character at the end of the array, the size of the character array containing the string is one more than the number of characters in the word "Hello."
- If you follow the rule of array initialization, then you can write the above statement as follows:

```
char greeting[] = "Hello";
```

- Following is the memory presentation of above defined string in C/C++:



- Actually, you do not place the null character at the end of a string constant.
- The C++ compiler automatically places the '\0' at the end of the string when it initializes the array. Let us try to print above-mentioned string:

```
#include <iostream>
using namespace std;
int main () {
   char greeting[6] = {'H', 'e', 'l', 'l', 'o', '\0'};
   cout << "Greeting message: ";
   cout << greeting << endl;
   return 0;
}
```

- When the above code is compiled and executed, it produces the following result:

```
Greeting message: Hello
```

String User Input

- User can input string from keyboard using gets function as follows:

```
char str[20];
gets(str);
cout<<str;
```

String functions

- C++ supports a wide range of functions that manipulate null-terminated strings:

No.	Function & Purpose
1.	`strcpy(s1, s2);` Copies string s2 into string s1.
2.	`strcat(s1, s2);` Concatenates string s2 onto the end of string s1.
3.	`strlen(s1);` Returns the length of string s1.
4.	`strcmp(s1, s2);` Returns 0 if s1 and s2 are the same; less than 0 if s1<s2; greater than 0 if s1>s2.
5.	`strchr(s1, ch);` Returns a pointer to the first occurrence of character ch in string s1.
6.	`strstr(s1, s2);`

	Returns a pointer to the first occurrence of string s2 in string s1.

- Following example makes use of few of the above-mentioned functions:

```
#include <iostream>
#include <cstring>
using namespace std;

int main () {
   char str1[10] = "Hello";
   char str2[10] = "World";
   char str3[10];
   int  len ;

   // copy str1 into str3
   strcpy( str3, str1);
   cout <<  str3 << endl;
   // concatenates str1 and str2
   strcat( str1, str2);
   cout << str1 << endl;
   // total lenghth of str1 after concatenation
   len = strlen(str1);
   cout << "length of str1 = " << len << endl;
   return 0;
}
```

- When the above code is compiled and executed, it produces result something as follows:

```
Hello
HelloWorld
```

```
length of str1 = 10
```

The String Class in C++

- The standard C++ library provides a string class type that supports all the operations mentioned above, additionally much more functionality.
- Let us check the following example:

```
#include <iostream>
#include <string>
using namespace std;
int main () {

   string str1 = "Hello";
   string str2 = "World";
   string str3;
   int  len ;

   // copy str1 into str3
   str3 = str1;
   cout << str3 << endl;

   // concatenates str1 and str2
   str3 = str1 + str2;
   cout << str3 << endl;

   // total length of str3 after concatenation
   len = str3.size();
   cout << len << endl;
return 0;
}
```

The output of the above program will be the same as the previous program.

Traversing a String (Iterate Over a String)

We can traverse a string using for loops, while loops and do while loops using a pointer to the first and the last index in the string:

```
string s="Hey, I am coding";
   for(int i=0;i<s.length();i++){
      cout<<s[i]<<" ";
   }
   cout<<endl;
```

Accessing Characters of String

- We can access the characters of a string using both iterators and pointer to the indices of the string.

```
#include <iostream>
using namespace std;
int main() {
string s="Hey, I am coding";
cout<<s<<endl;
for(int i=0;i<s.length();i++){
s[i]='A'; }
cout<<s<<endl;  )
for(int i=0;i<s.length();i++)
{ s[i]='B'; }
cout<<s<<endl;
```

```
return 0; }
```

The output will be:

```
Hey, I am coding

AAAAAAAAAAAAAAA

BBBBBBBBBBBBBBB
```

String Functions

- String is an object of the <string> class
- It has a variety of functions that users can utilize for a variety of operations.

Function	Description
`length()`	This function returns the length of the string.
`find()`	To detect the position of substring within the string. This function returns the index of the first occurrence of the specified substring or character.
`clear()`	It erases the contents of the string, which becomes an empty string.
replace()	It replaces the portion of the string that begins at character pos and spans len characters.
compare()	To compare two strings. It provides a way to lexicographically compare the content of the string objects with another string or a sub string. It returns 0 if strings are equal, <0: The calling string is less than the argument string, or > 0: The calling string is greater than the argument string.

Homework:

1- Write a C++ program to reverse a given string. Example: if the input string was "**Hello**" then the output string should be "**olleH**", **without** using built-in functions.

2- Write a C++ function called **`find()`** that searches for a certain character in a string. If the character is found, the function returns the index of that character in the string. Otherwise, it returns -1.

Example 1: **`find("Computer",'p')`** the output will be 3.

Example 2: **`find("Computer",'w'),`** the output will be -1.

Note: Do not use built-in function find().

3- Write a C++ program to swap a certain character in a string.

Example:

```
string s1 = "Computer"
```

If we want to swap the letter 'r' in s1 with the letter 's' then the output should be "Computes". The function should receive the original string, the old character and the new character. Example: swap(s1,'r','s'). The function should return the new string.

4- Write a C++ function called **`substr()`** that returns a substring from a string.

Example:

```
string s1 = "Science";
```

if we call the above function **`substr(s1, 2, 6),`** then the output should be "ience"

5- Write a C++ function called **`beginswith()`** that checks if a given string starts with a given character, **without** using built-in functions.

Example 1: **`beginswith("Programming",'r')`** the output will be false

Example 2: **`beginswith("Programming",'P')`** the output will be true

6- Write a C++ function called **`endswith()`** that checks if a given string ends with a given character, **without** using built-in functions.

Example 1: **`endswith("Computer",'y')`** the output will be false

Example 2: **`endswith("Computer",'r')`** the output will be true

7- Write a C++ function called **`count()`** that counts how many a specific character is repeated in a given string.

Example 1: **`count("statistics", 's')`** the output will be 3

Example 2: **`count("statistics", 'n')`** the output will be 0

8- Write a C++ program to remove any repeated character in a string. **Example:** if the input string was "samsung" then the output should be "samung"

9- Write a C++ program that prints all the repeated characters in a string. **Example:** if the input string was "Motorola", then the output string should be "o"

10- Write a C++ function called **`uppcase()`** that returns the uppercase of a given string. **Example:** if the input string was "program", then the output string should be "PROGRAM". **Note**: do not use any built-in function.