String Class in C++

The standard C++ library provides a **string** class type that supports all the operations mentioned above, additionally much more functionality. We will study this class in C++ Standard Library but for now let us check following example:

At this point, you may not understand this example because so far we have not discussed Classes and Objects. So can have a look and proceed until you have understanding on Object Oriented Concepts.

```
#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 : " << str3 << endl;

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

   // total lenghth of str3 after concatenation
   len = str3.size();
   cout << "str3.size() :  " << len << endl;

   return 0;
}
```

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

```
str3 : Hello
str1 + str2 : HelloWorld
str3.size() :  10
```

cin and strings

The extraction operator can be used on `cin` to get strings of characters in the same way as with fundamental data types:

```
string mystring;
cin >> mystring;
```

However, `cin` extraction always considers spaces (whitespaces, tabs, new-line...) as terminating the value being extracted, and thus extracting a string means to always extract a single word, not a phrase or an entire sentence.

To get an entire line from `cin`, there exists a function, called `getline`, that takes the stream (`cin`) as first argument, and the string variable as second. For example:

```
// cin with strings
#include <iostream>
#include <string>
using namespace std;

int main ()
{
  string mystr;
  cout << "What's your name? ";
  getline (cin, mystr);
  cout << "Hello " << mystr << ".\n";
  cout << "What is your favorite team? ";
  getline (cin, mystr);
  cout << "I like " << mystr << " too!\n";
  return 0;
}
```

```
What's your name? Homer Simpson
Hello Homer Simpson.
What is your favorite team? The Isotopes
I like The Isotopes too!
```

Notice how in both calls to `getline`, we used the same string identifier (`mystr`). What the program does in the second call is simply replace the previous content with the new one that is introduced.

C-String manipulation

C++ provides following two types of string representations:

- The C-style character string.
- The string class type introduced with Standard C++.

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'. Thus a null-terminated string contains the characters that comprise the string followed by a **null**.

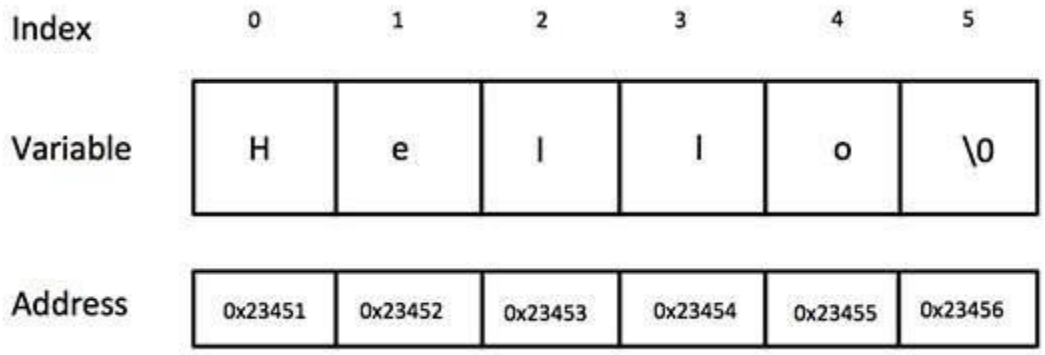
The following declaration and initialization create a string consisting of the word "Hello". To hold the null character at the end of the array, the size of the character array containing the string is one more than the number of characters in the word "Hello."

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

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 result something as follows:

```
Greeting message: Hello
```

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

	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.

strlen(s1);

3

Returns the length of string s1.

strcmp(s1, s2);

4

Returns 0 if s1 and s2 are the same; less than 0 if s1<s2; greater than 0 if s1>s2.

strchr(s1, ch);

5

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

strstr(s1, s2);

6

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 << "strcpy( str3, str1) : " << str3 << endl;

   // concatenates str1 and str2
   strcat( str1, str2);
   cout << "strcat( str1, str2): " << str1 << endl;

   // total lenghth of str1 after concatenation
   len = strlen(str1);
   cout << "strlen(str1) : " << len << endl;

   return 0;
}
```

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

```
strcpy( str3, str1) : Hello
strcat( str1, str2): HelloWorld
strlen(str1) : 10
```

C- string manipulation

A string can be created by using a pointer or an array of characters.

Using pointer char * string_name;
where: string_name is a pointer.
Example: char *lastname;
char * firstname;

Using array char string_name[size];
where: size is the number of characters including the null character \0 stored in the array..
Example: char lastname[30];
char firstname[20];

STRING DECLARATION AND INITIALIZATION

A string can be declared and initialized by using a pointer or an array of characters.

Using pointer char * string_name = "string_value";
where: string_name is a pointer.
Example: char *lastname = "Nguyen";
char * firstname = "Phuong";

Using array char string_name[size] = "string-value";
where: size is the number of characters including the null character \0 stored in the array..
Example: char lastname[30] = "Nguyen";
char firstname[20] = "Phuong";

STRING INPUT AND OUTPUT

Table below lists the commonly available library functions for string input and output

C++ Routine	Description
cout	String output to the screen
cin	String input from the keyboard, but cin does not input a space character. It stops input a string when it reads a space character.
cin.getline(str, length,char)	String input from the keyboard. cin.getline( ) read a space character.

	str - a string of character pointer or a character array. length - an integer constant or a variable indicating the maximum number of input characters including the null character. char - an optional character constant or variable specifying the terminating character. If this optional third argument is omitted, the default terminating character is the newline (\n) character. Pressing the enter key generates a newline character. A statement such as cin.getline(message,80,'x') will stop accepting characters whenever the x key is pressed.
chr = cin.get( )	Input single character from the keyboard. chr - a character constant variable

Example	Output

#include<iostream>
using namespace std;
int main( )
{char message1[80];
char *message2;
cout <<"Enter a string for message1: \n";
cin.getline(message1,80);
cout << "Enter a string for message2: \n";
cin.getline(message2,80);
cout <<message1<< " and " <<message2;
}
	Enter a string for message1:
	Good morning
	Enter a string for message2:
	have a nice day
	Good morning and have a nice day

STRING LIBRARY FUNCTIONS

Extensive collections of string-handling functions are included with all C++ compilers. The common of these are listed below. To call these functions, you need to include the header file <string.h> in your program.

String copy

strcpy(string1,string2) - Copies string2 to string1. String1 needs to have enough space to store string2. The strcpy will overwrite string1.

Example: char message1[80] ="Good morning", message2[80]="Hello World";
cout <<message1<<endl;
strcpy(message1,message2);
cout << message1;

Output
Good morning
Hello World

String concatenation

strcat(string1,string2) - concatenates string2 to string1. String1 needs to have enough space to append
string2.

Example: char message1[80] ="Good morning", message2[80]=" and have a nice day";
cout <<message1<<endl;
strcat(message1,message2);
cout << message1<<endl;
cout << message2;
Output
Good morning
Good morning and have a nice day
and have a nice day

String comparison
strcmp(string1, string2) - Compares string1 to string2. Returns a negative integer if string1<string2, 0 if string1
is equal to string2, and a positive integer if string1 > string2.

Example: char message1[80] = "It's a cow";
char *mesaage2 = "It's a Cow";
if (strcmp(string1,string2))
cout << "string 1 is greater than string2";
else
cout << "string1 is smaller than string2";
Ouput

string 1 is greater than string2

Note: When strcmp compare the character c in string1 with the character C in string2. The character c is greater than the character C because the asscii value of character c is 99 and the asscii value of character C is only 67. See the Appendix B ASCII character set in the back of your text book.

String length
strlen(string1) - Return the length of the string, excluding the null character

Example: char message[80] = "Hello world";
int i;
i = strlen(message);
cout << i << " characters";
Output
11 characters

Note: a space is counted as one character.

CHARACTER STRING FUNCTIONS

C++ provides several functions that allow you to test and manipulate character data. The function prototypes are found in the header file name <ctype.h>. Remember to add the line #include <ctype.h> in program that use these functions. The table below lists and describes the character functions. Each function expects one integer argument - the ASCII value of the character to be tested. Each function returns a non-zero value (true) if the condition tested is true and 0 (false) if the condition tested is false.

C++ Functions	Description
isalpha(character)	Returns a nonzero number if the character is a letter ('A' - 'Z', 'a' - 'z'); otherwise it returns zero.
isalnum(character)	Returns a nonzero number if the character is a letter ('A' - 'Z', 'a' - 'z', or '0' - '9'; otherwise it returns zero.
isdigit(character)	Returns a nonzero number if the character is digit (0 through 9); otherwise it returns a zero.
isspace(character)	Returns a nonzero number if the character is a whitespace (tab, space, newline); otherwise it returns a zero.
isupper(character)	Returns a nonzero number if the character is uppercase; otherwise it returns a zero.
islower(character)	Returns a nonzero number if the character is lowercase; otherwise it returns a zero.
toupper(character)	Return the uppercase equivalent if the character is lowercase; otherwise it returns the character unchanged.
tolower(character)	Return the lowercase equivalent if the character is uppercase; otherwise it returns the character unchanged.

C++ Functions	Description
toupper(character)	Return the uppercase equivalent if the character is lowercase; otherwise it returns the character unchanged.
tolower(character)	Return the lowercase equivalent if the character is uppercase; otherwise it returns the character unchanged.
atoi(string)	Converts an ASCII string to an integer (include #<stdlib.h> in your program)
atof(string)	Converts an ASCII string to an float (include #<stdlib.h> in your program)

The example below will convert each lowercase character of a string to uppercase character and vice versa.

Example

```
#include<iostream>
#include<string>
#include<cctype>
using namespace std;
int main( )
{ char name[20];
cout<<"Enter your name:\n ";
cin.getline(name,20);
for( int i = 0; i < strlen(name) ; i++)
{ if (islower(name[i]) )
//convert to uppercase
name[i] = toupper(name[i]);
else
//convert to lowercase
name[i] = tolower(name[i]);
}
//Display the result
cout << "The conversion is:\n";
cout << name << endl;
}
```

Output

```
Enter your name:
Phuong D. Nguyen
The conversion is:
pHUONG d. nGUYEN
```

You can rewrite the example above using the pointer.

Example

```
#include<iostream>
#include<string>
#include<cctype>
using namespace std;
int main()
{ char *name;
cout<<"Enter your name:\n ";
cin.getline(name,20);
for( int i = 0; i < strlen(name) ; i++)
{ if (islower(*(name + i) ) )
//convert to uppercase
*( name + i) = toupper(*(name + i));
else
//convert to lowercase
*( name + i) = tolower(*(name + i));
}
//Display the result
cout << "The conversion is:\n";
cout << name << endl;
}
```

Output

```
Enter your name:
Phuong D. Nguyen
The conversion is:
pHUONG d. nGUYEN
```

Write a function that returns the number of digits in a given null-terminated string.

```
#include<iostream>
#include<cctype>
using namespace std;
int numAlphas(const char* s)
{
	int count = 0;
	for (int i = 0; s[i] != '\0'; i++)
	{
		if (isdigit(s[i]))
		{
			count++;
		}
	}
	return count;
}

int main()
{
	char str[] = "a12bc3d";
```

```
	cout << numAlphas(str);

}
```

C Strings and Pointers

```
// Create your own strlen function
#include <iostream>
using namespace std;
int myStrLen(char str[]);

int main()
{
	char s[15] = "Hello World";
	cout << myStrLen(s);
	return 0;
}
//----------------------------------------------------------------
int myStrLen(char str[])
{
	int i = 0;
	while (str[i] != '\0')
		i++;
	return i;
}
```

Or

```
int myStrLen(char *str)
{
	char *first = str;
	while (*str != '\0')
		str++;
	return str - first;
}
```

Or

```
int myStrLen(char *str)
{
	char *first = str;
	while (*str)
		str++;
	return str - first;

}
```

```
// create your own strcpy function
#include <iostream>
using namespace std;
void myStrcpy(char str2[], char str1[]);
int main()
{
	char s1[15] = "Hello World";
	char s2[30];
	myStrcpy(s2, s1);
	cout << s2;
	return 0;
}
//-------------------------------------------------------------
void myStrcpy(char *to, char * from)
{
	while (*to = *from)
	{
		to++;
		from++;
	}

}
```

Or

```
void myStrcpy(char *to, char * from)
{
	while (*to++ = *from++);

}
```