C++ Standard Library: The String Class

The **string** class is part of the C++ standard library. A string represents a sequence of characters.

To use the string class, # include the header file:

#include <string>

**Constructors:**

* string ()   
  - creates an empty string ("")
* string ( other\_string )   
  - creates a string identical to other\_string
* string ( other\_string, position, count )   
  - creates a string that contains count characters from other\_string, starting at position. If count is missing (only the first two arguments are given), all the characters from other\_string, starting atposition and going to the end of other\_string, are included in the new string.
* string ( count, character )   
  - create a string containing character repeated count times

Examples:

string s1; // s1 = ""

string s2( "abcdef" ); // s2 = "abcdef"

string s3( s2 ); // s3 = "abcdef"

string s4( s2, 1 ); // s4 = "bcdef"

string s5( s2, 3, 2 ); // s5 = "de"

string s6( 10, '-' ); // s6 = "----------"

The **string** class also has a destructor that takes care of freeing the memory storing the characters when the object is destroyed.

**Constant Member Functions:**

These functions do not modify the string.

* const char \* data ()   
  - returns a C-style null-terminated string of characters representing the contents of the string
* unsigned int length ()   
  - returns the length of the string
* unsigned int size ()   
  - returns the length of the string (i.e., same as the length function)
* bool empty ()   
  - returns true if the string is empty, false otherwise

**Operators Defined for string:**

* *Assign* =   
  string s1;  
  string s2;  
  ...  
  s1 = s2; // the contents of s2 is **copied** to s1
* *Append* +=   
  string s1( "abc" );  
  string s2( "def" );  
  ...  
  s1 += s2; // s1 = "abcdef" now
* *Indexing* []   
  string s( "def" );  
  char c = s[2]; // c = 'f' now  
  s[0] = s[1]; // s = "eef" now
* *Concatenate* +   
  string s1( "abc" );  
  string s2( "def" );  
  string s3;  
  ...  
  s3 = s1 + s2; // s3 = "abcdef" now
* *Equality* ==   
  string s1( "abc" );  
  string s2( "def" );  
  string s3( "abc" );  
  ...  
  bool flag1 = ( s1 == s2 ); // flag1 = false now  
  bool flag2 = ( s1 == s3 ); // flag2 = true now
* *Inequality* !=   
  - the inverse of equality
* *Comparison* <, >, <=, >=   
  - performs case-insensitive comparison   
  string s1 = "abc";  
  string s2 = "ABC";  
  string s3 = "abcdef";  
  ...  
  bool flag1 = ( s1 < s2 ); // flag1 = false now  
  bool flag2 = ( s2 < s3 ); // flag2 = true now

**Member Functions:**

* void swap ( other\_string )   
  - swaps the contents of this string with the contents of other\_string.   
  string s1( "abc" );  
  string s2( "def" );  
  s1.swap( s2 ); // s1 = "def", s2 = "abc" now
* string & append ( other\_string )   
  - appends other\_string to this string, and returns a reference to the result string.
* string & insert ( position, other\_string )   
  - inserts other\_string into this string at the given position, and returns a reference to the result string.
* string & erase ( position, count )   
  - removes count characters from this string, starting with the character at the given position. If count is ommitted (only one argument is given), the characters up to the end of the string are removed. If both position and count are omitted (no arguments are given), the string is cleared (it becomes the empty string). A reference to the result string is returned.
* unsigned int find ( other\_string, position )   
  - finds other\_string inside this string and returns its position. If position is given, the search starts there in this string, otherwise it starts at the beginning of this string.
* string substr ( position, count )   
  - returns the substring starting at position and of length count from this string

## #2: Source Code:

/\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*|

\* String replace Program |

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*|

\* Takes three string input from the user

\* Replaces all the occurances of the second string

\* with the third string from the first string

\* @author Swashata

\*/

/\*\* Include Libraries \*/

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

/\*\* Define the max char length \*/

#define MAX\_L 4096

/\*\* Prototypes \*/

void replace (char \*, char \*, char \*);

int main(void) {

char o\_string[MAX\_L], s\_string[MAX\_L], r\_string[MAX\_L]; //String storing variables

printf("Please enter the original string (max length %d characters): ", MAX\_L);

fflush(stdin);

gets(o\_string);

printf("\nPlease enter the string to search (max length %d characters): ", MAX\_L);

fflush(stdin);

gets(s\_string);

printf("\nPlease enter the replace string (max length %d characters): ", MAX\_L);

fflush(stdin);

gets(r\_string);

printf("\n\nThe Original string\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

puts(o\_string);

replace(o\_string, s\_string, r\_string);

printf("\n\nThe replaced string\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

puts(o\_string);

return 0;

}