

# Tổng hợp cách dùng string c++

Các thư viện cần sử dụng:

```
#include<bits/stdc++.h> // Thư viện tổng hợp, bao gồm <string>
using namespace std; // Sử dụng không gian std

int main()
{
    ios_base::sync_with_stdio(0); // Tắt đồng bộ, tiết kiệm
                                  // thời gian chạy khi dùng cin, cout
}
```

Cách khai báo biến string, nhập và xuất string:

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

int main()
{
    ios_base::sync_with_stdio(0);
    string s; // Khai báo biến string s
    cin>>s; // Nhập dữ liệu vào string s từ bàn phím
            // Dùng getline(cin, s) nếu string chứa dấu 'khoảng trắng'
    cout<<s; // In string s ra màn hình
}
```

Truy cập từng kí tự trong string s:

```
for (int i=0; i<=s.size()-1; i++) // string s chứa kí tự từ
{                                  // vị trí 0 đến s.size()-1
    cout<<s[i]; // s[i] là từng kí tự trong string s
}
```

Hàm lấy chiều dài chuỗi string s:

```
s.size()
s.length()
```

Cách ghép chuỗi:

```
string a,b,c;
a = "AAA";
b = "BBB";
c = b+a; // c="BBBAAA"
c = a+b; // c="AAABBB"
c = c+'e'; // Chèn kí tự 'e' vào sau string c
c = 'e'+c; // Chèn kí tự 'e' vào trước string c
```

Cách chèn chuỗi:

```
//tên_biến.insert(vị_trí_chèn, chuỗi_chèn);  
string a,b;  
a = "AAA";  
b = "BBB";  
a.insert(1,b); // a="ABBBAA"
```

Cách xóa chuỗi:

```
//tên_biến.erase(vị_trí_xóa, số_lượng);  
string a;  
a = "AAABBB";  
a.erase(2,3); // a="AAB"
```

Lấy chuỗi con liên tiếp trong một biến string:

```
// tên_biến.substr(vị_trí_bắt_đầu, số_lượng);  
string a,b;  
a = "AAABBB";  
b = a.substr(1,3); // b = "AAB"  
// Từ vị trí 1, lấy 3 kí tự
```

Tìm kiếm chuỗi con trong chuỗi string s:

```
// tên_biến.find(chuỗi_con, vị_trí_tìm);  
string s;  
s = "AAABBB";  
long long t = s.find("AB",0); // Nên gán vào biến long long t  
// Đừng sử dụng trực tiếp  
cout<<t; // t = 2 (chuỗi "AB" nằm ở vị trí 2 trong string s)  
t = s.find("AB",5);  
cout<<t; // t = -1 (không tồn tại "AB" kể từ vị trí 5)
```

Sắp xếp các kí tự trong chuỗi (theo mã ASCII):

```
sort(s.begin(), s.end()); // Sắp xếp từ nhỏ đến lớn  
sort(s.begin(), s.end(), greater<char>()); //Sắp xếp từ lớn đến nhỏ
```

# 1. Chuỗi kí tự

**Định nghĩa:** Chuỗi kí tự trong ngôn ngữ lập trình C thực chất là mảng một chiều của các ký tự. Các hằng chuỗi ký tự được đặt trong cặp dấu nháy kép "".

**Khai báo:** Cú pháp khai báo (gần giống với mảng số nguyên):

```
char tên_biến[độ_dài];
```

Ví dụ: `char s[20];`

**Nhập chuỗi:** Cú pháp: có 2 cách:

Nhập hết cả 1 dòng:

```
gets(<biến_chuỗi>);
```

Nhập đến dấu khoảng trắng:

```
scanf("%s", &<biến_chuỗi>);
```

ví dụ:

```
char s[20];  
gets(s);  
scanf("%s", &s);
```

**Xuất chuỗi:** Cú pháp: có 2 cách:

```
puts(<biến_chuỗi>);
```

```
printf("%s", <biến_chuỗi>);
```

ví dụ:

```
char s[20] = "abcd";  
puts(s); // in ra màn hình "abcd"
```

### Một số hàm xử lý chuỗi:

STT	Hàm & Mục đích
1	<b>strcpy(s1, s2);</b>
	Sao chép chuỗi s2 cho chuỗi s1.
2	<b>strcat(s1, s2);</b>
	Nối chuỗi s2 vào cuối chuỗi s1.
3	<b>strlen(s1);</b>
	Trả về độ dài của chuỗi s1.
4	<b>strcmp(s1, s2);</b>
	Trả về 0 nếu s1 và s2 là như nhau; nhỏ hơn 0 nếu s1<s2; lớn hơn 0 nếu s1>s2.
5	<b>strchr(s1, ch);</b>
	Trả về con trỏ tới vị trí đầu tiên của ch trong s1.
6	<b>strstr(s1, s2);</b>
	Trả về con trỏ tới vị trí đầu tiên của chuỗi s2 trong chuỗi s1.

### Một số ví dụ:

```
// Khai báo một string theo kiểu mảng. Với ký tự null ở cuối.  
char mystring[] = { 't', 'h', 'i', 's', ' ', 'i', 's',  
, ' ', 't', 'e', 'x', 't', '\0' };  
  
// Đây là một string literal  
char mystring[] = "this is text";
```

```
#include <stdio.h>

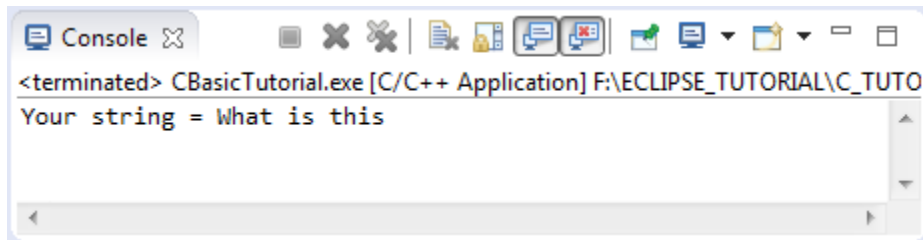
int main() {

    // Khai báo một String (String literal).
    char s1[] = "What is this";

    // In ra nội dung của string.
    printf("Your string = %s", s1);

    return 0;
}
```

Kết quả chạy ví dụ:



```
#include <stdio.h>

int main() {

    // Khai báo 1 mảng ký tự có 100 phần tử
    // dùng để lưu trữ string người dùng nhập vào từ bàn
    phím.
    char s1[100];

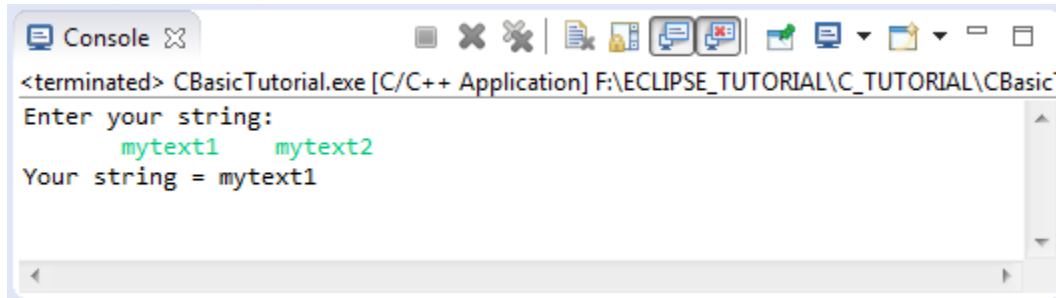
    printf("Enter your string: \n");

    // Hàm scanf chờ đợi người dùng nhập vào.
    // Nó sẽ quét string (định dạng bởi %s) và truyền vào
    biến s1.
    // Chú ý định dạng %s sẽ quét lấy đoạn string đầu
    tiên không chứa khoảng trắng.
    scanf("%s", s1);

    printf("Your string = %s", s1);
}
```

```
    return 0;
}
```

Kết quả chạy ví dụ:



```
#include <stdio.h>

// Khai báo sử dụng thư viện string.h
#include <string.h>

int main() {

    // Khai báo một String (String literal).
    char s1[] = "This is ";

    // Khai báo một string theo kiểu khai báo mảng
    // Với phần tử null ở cuối.
    char s2[] = { 't', 'e', 'x', 't', '\0' };

    // Hàm: size_t strlen(const char *str)
    // Sử dụng hàm strlen để kiểm tra độ dài chuỗi.
    // size_t là kiểu dữ liệu số nguyên không dấu.
    size_t len1 = strlen(s1);
    size_t len2 = strlen(s2);

    printf("Length of s1 = %d \n", len1);
    printf("Length of s2 = %d \n", len2);

    // Khai báo một mảng ký tự có 100 phần tử
    // (100 phần tử là đủ dùng trong trường hợp này).
    char mystr[100];
```

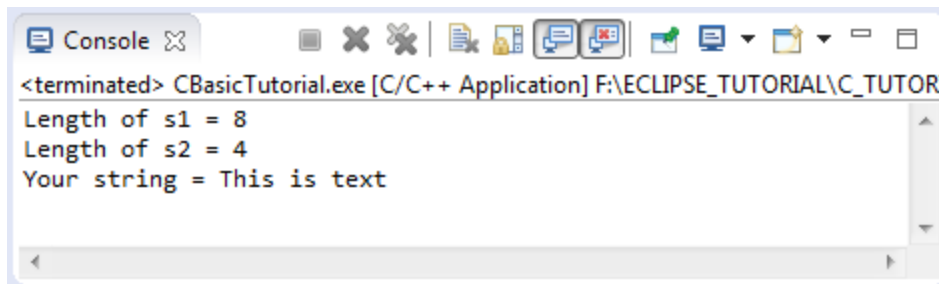
```
// Hàm: char *strcpy(char *dest, const char *src)
// Copy string s1 vào mystr
strcpy(mystr, s1);

// Hàm: char *strcat(char *dest, const char *src)
// Sử dụng hàm strcat để nối s2 vào mystr
strcat(mystr, s2);

// In ra nội dung của string.
printf("Your string = %s", mystr);

return 0;
}
```

Kết quả chạy ví dụ:



## 2. C++ Style String

C++ cung cấp cho bạn class **string**, nó giúp bạn làm việc dễ dàng hơn với các chuỗi. Các phương thức mà class **string** cung cấp vẫn hỗ trợ để làm việc với các **C-Style** string.

### 2.1- Khai báo thư viện string

Để sử dụng **string** bạn phải khai báo chỉ thị tiền xử lý (Preprocessor Directives) **#include <string>** và khai báo sử dụng không gian tên **std**.

```
// Khai báo chỉ thị tiền xử lý (Preprocessor Directives)

#include <string>

// Khai báo sử dụng không gian tên std.

using namespace std;
```

Khai báo string:

```
// Khai báo một string.

string mystring = "Hello World";

// Nếu bạn không khai báo sử dụng namespace std.
// Bạn phải sử dụng tên đầy đủ:

std::string mystring = "Hello World";
```

Nhập xuất :

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

int main()
{
    string s;
```



```

    //Nhập đến dấu khoảng trắng (chuỗi không có khoảng
    trắng)
    cin >> s;

    //Nhập cả 1 dòng (chuỗi có khoảng trắng)
    getline(cin, s);

    //Xuất chuỗi
    cout << s;
}

```

### Lấy chiều dài chuỗi:

Ta sử dụng 2 hàm là **size()** hoặc **length()** để lấy chiều dài chuỗi.

Ví dụ: viết chương trình nhập tên người dùng, In ra độ dài của tên.

```

#include <iostream>
#include <string>
using namespace std ;
int main()
{
    string name;
    cout << "What is your name? ";
    getline(cin,name);
    cout << "The length of your name is: "<<
name.length()<<endl;
    cout << "The size of your name is: "<<
name.size()<<endl ;
}

```

### Truy cập một phần tử bất kỳ trong chuỗi:

Chuỗi thực chất là một mảng các phần tử kiểu char. Do đó, ta có thể truy xuất các phần tử trong chuỗi tương tự như truy xuất với mảng thông thường.

Ví dụ: Viết chương trình nhập tên người dùng. Sau đó, in ra các kí tự của tên người dùng trên từng dòng một.

```

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

```

```

{
    string name;
    cout << "What is your name? ";
    getline(cin, name);
    for(int i = 0; i < name.size(); i++)
        cout << name[i] << endl;
}

```

### Ghép chuỗi:

Để ghép chuỗi, ta dùng phép toán “+”.

Ví dụ: Viết chương trình nhập lần lượt họ và tên vào 2 chuỗi firstname và lastname khác nhau. Sau đó, ghép 2 chuỗi này vào chuỗi là fullname.

```

#include <iostream>
#include <string>
using namespace std;
int main()
{
    string firstname , lastname , fullname ;
    cout << "What is your firstname name? ";
    getline(cin,firstname);
    cout << "What is your lastname name? ";
    getline(cin,lastname);
    // ghép chuỗi
    fullname = firstname + " " + lastname ;
    cout<<"Your fullname is: " <<fullname ;
}

```

### Chèn chuỗi:

Để chèn chuỗi trong C++, ta dùng hàm insert theo cấu trúc sau:

**insert( position, str) ;**

Trong đó:

position là vị trí cần chèn ( kiểu int ).

str là chuỗi cần chèn (kiểu string hoặc char) .

Ví dụ: Chèn chuỗi “lap trinh” vào chuỗi “Hoc that de” để tạo thành chuỗi “Hoc lap trinh that de”.

```

#include <iostream>
#include <string>
using namespace std ;
int main()
{
    string myString = "Hoc that de" ;
    myString.insert(4, "Lap trinh ") ;
    //vị trí cần chèn vị trí khoảng trắng đầu tiên bên
    position=4
    cout<<myString ;
}

```

### Xoá chuỗi:

Để xoá chuỗi trong C++, ta dùng hàm erase theo cấu trúc sau:

**erase (position, number) ;**

Trong đó:

position là vị trí cần xoá ( kiểu int ).

number là số kí tự cần xoá (kiểu int) .

Ví dụ: Xoá chuỗi “Khong” trong chuỗi “Khong thich hoc lap trinh” để tạo thành chuỗi “thich hoc lap trinh” .

```

#include <iostream>
#include <string>
using namespace std ;
int main()
{
    string myString = "Khong thich hoc lap trinh" ;
    myString.erase( 0 , 6 ) ;
    // vị trí cần xoá là vị trí 0 và số lượng kí tự cần
    xoá là 6
    cout<<myString ;
}

```

### Tìm kiếm chuỗi con

**find (string& str, pos) ;**

Nếu không quy định giá trị pos thì hiểu mặc nhiên là 0; nếu tìm có thì phương thức trả về vị trí xuất hiện đầu tiên, ngược lại trả về giá trị -1

**rfind (string& str, pos) ;**

Tìm kiếm ngược lại (tìm từ phải sang trái)

### Một số phương thức của chuỗi:

<code>empty()</code> ;	Kiểm tra chuỗi rỗng
<code>clear()</code> ;	Xóa chuỗi thành chuỗi rỗng
<code>front()</code> ;	Lấy ký tự đầu tiên của chuỗi
<code>back()</code> ;	Lấy ký tự cuối cùng của chuỗi

## 2.2- Các phương thức của String và một số ví dụ (đọc thêm)

Dưới đây là danh sách các phương thức của **String**.

### 2.2.1- `length()`

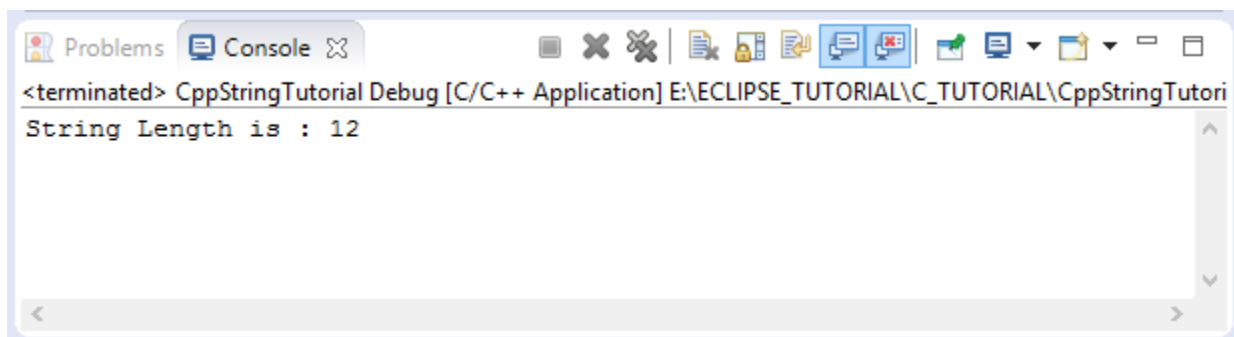
**`length()`** là một trong các phương thức thông dụng nhất của **String**, nó trả về độ dài chuỗi (Số ký tự của chuỗi).

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

int main() {
    string str = "This is text";
    int len = str.length();

    cout << "String Length is : " << len << endl;
    return 0;
}
```

Kết quả chạy ví dụ:



### 2.2.2- append

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

int main() {

    string s1 = "One";
    string s2 = "Two";
    string s3 = "Three";

    cout << "s1 = " + s1 << endl;
    cout << "s2 = " + s2 << endl;
    cout << "s3 = " + s3 << endl;
    cout << " ----- " << endl;

    // Nối thêm s2 vào s1.
    // Làm thay đổi s1 và trả về tham chiếu của s1.
    string s = s1.append(s2);

    cout << "s1 = " + s1 << endl; // ==> OneTwo
    cout << "s = " + s << endl;   // ==> OneTwo

    cout << " ----- " << endl;

    // Nối s2 và s3 vào s1.
    // Làm thay đổi s1 và trả về tham chiếu của s1.
    s = s1.append(s2).append(s3); // ==>
OneTwoTwoThree

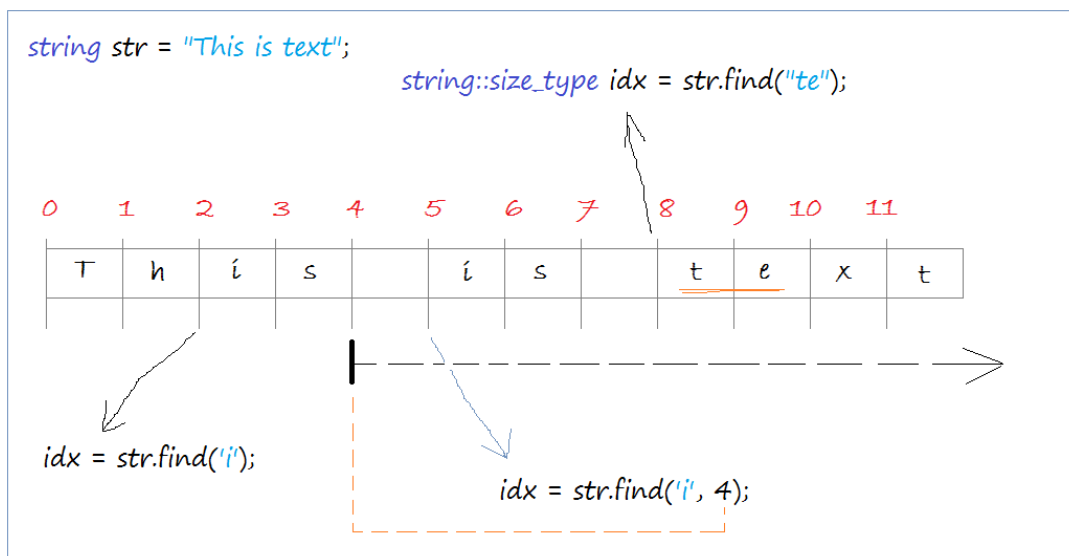
    cout << "s1.append(s2).append(s3) = " + s <<
endl;
}
```

Kết quả chạy ví dụ:

```
Problems Console
<terminated> CppStringTutorial Debug [C/C++ Application] E:\ECLIPSE_TUTORIAL\C_TUTORIAL\CppStringTutorial
s1 = One
s2 = Two
s3 = Three
-----
s1 = OneTwo
s = OneTwo
-----
s1.append(s2).append(s3) = OneTwoTwoThree
```

### 2.2.3- find

**find** là phương thức tìm vị trí xuất hiện của một chuỗi con trong chuỗi hiện tại. Phương thức này trả về hằng số **string::npos** nếu không tìm thấy.



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

int main() {

    string str = "This is text";

    // Tìm vị trí xuất hiện ký tự 'i' đầu tiên.
    // ==> 2
```

```

string::size_type idx = str.find('i');

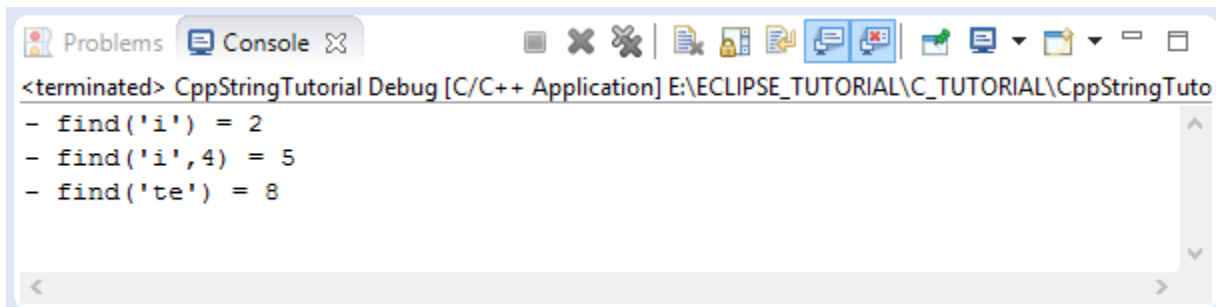
cout << "- find('i') = " << idx << endl;

// Tìm vị trí xuất hiện ký tự 'i' đầu tiên
// tính từ chỉ số thứ 4 trở về cuối chuỗi.
// ==> 5
idx = str.find('i', 4);
cout << "- find('i',4) = " << idx << endl;

// Tìm vị trí xuất hiện chuỗi con "te" đầu tiên.
// ==> 8
idx = str.find("te");
cout << "- find('te') = " << idx << endl;
}

```

Kết quả chạy ví dụ:



```

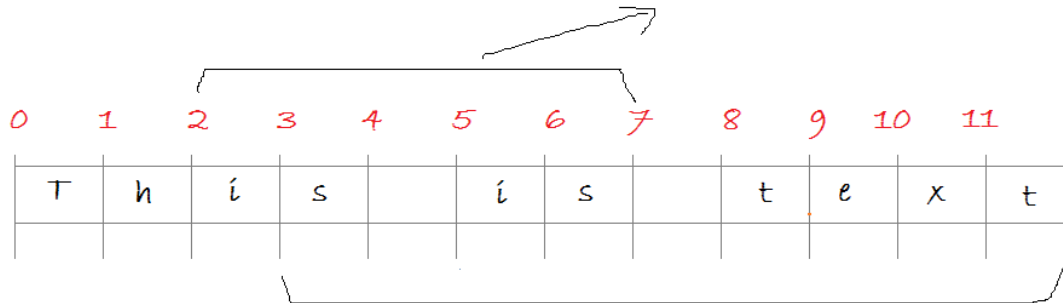
<terminated> CppStringTutorial Debug [C/C++ Application] E:\ECLIPSE_TUTORIAL\C_TUTORIAL\CppStringTuto
- find('i') = 2
- find('i',4) = 5
- find('te') = 8

```

## 2.2.4- substr

```
string str = "This is text";
```

```
string substr = str.substr(2, 7);
```



```
string substr = str.substr(3);
```

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

int main() {

    string str = "This is text";

    // Trả về chuỗi con từ chỉ số thứ 3 tới cuối chuỗi.
    string substr = str.substr(3);

    cout << "- str.substr(3)=\"" << substr << endl;

    // Trả về chuỗi con từ chỉ số thứ 2 cho tới chỉ số 7
    substr = str.substr(2, 7);

    cout << "- str.substr(2, 7) =" << substr << endl;

}
```

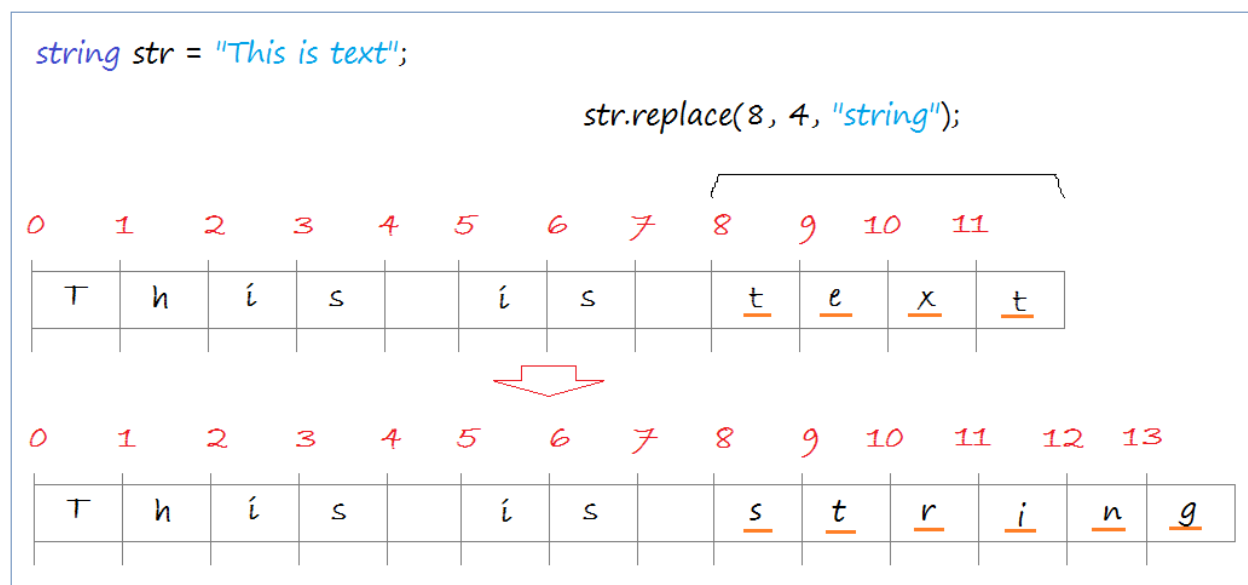


Kết quả chạy ví dụ:

```
Problems Console
<terminated> CppStringTutorial Debug [C/C++ Application] E:\ECLIPSE_TUTORIAL\C_TUTORIAL\CppStringTutor
- str.substr(3)=s is text
- str.substr(2, 7) =is is t
```

## 2.2.5- replace

Một số phương thức liên quan tới thay thế.



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

int main() {

    string str = "This is text";

    // Thay thế các chuỗi con 4 ký tự bắt đầu từ vị trí
    8
```

```

// bởi chuỗi "string", và trả về tham chiếu
string s2 = str.replace(8, 4, "string");

cout << "- str=" << str << endl; // This is string
cout << "- s2=" << s2 << endl;    // This is string
cout << " ----- " << endl;

// Sét lại str
str = "This is text";

// Thay thế chuỗi con 4 ký tự bắt đầu từ chỉ số 8,
bởi một chuỗi con của
// một chuỗi khác, 6 ký tự bắt đầu từ chỉ số 0.
// ==> It is string
string s3 = str.replace(8, 4, "string is important",
0, 6);

cout << "- str=" << str << endl; // It is string
cout << "- s3=" + s3 << endl;    // It is string
cout << " ----- " << endl;

}

```

Kết quả chạy ví dụ:

```

<terminated> CppStringTutorial Debug [C/C++ Application] E:\ECLIPSE_TUTORIAL\C_TUTORIAL\CppStringT
- str=This is string
- s2=This is string
-----
- str=This is string
- s3=This is string
-----

```

Một ví dụ kết hợp giữa **find & replace** để thay thế tất cả các chuỗi con cụ thể bởi một chuỗi mới.

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

// Một hàm tiện ích tìm kiếm và thay thế các chuỗi con
bởi một chuỗi mới.
// search: Chuỗi cần tìm.
// replace: Chuỗi thay thế.
string replaceAll(string subject, const string& search,
const string& replace) {
    size_t pos = 0;
    // Hàm find sẽ trả về string::npos nếu không tìm
    thấy chuỗi con.
    while ((pos=subject.find(search,pos))!=string::npos)
    {
        subject.replace(pos, search.length(), replace);
        pos += replace.length();
    }
    return subject;
}

int main() {

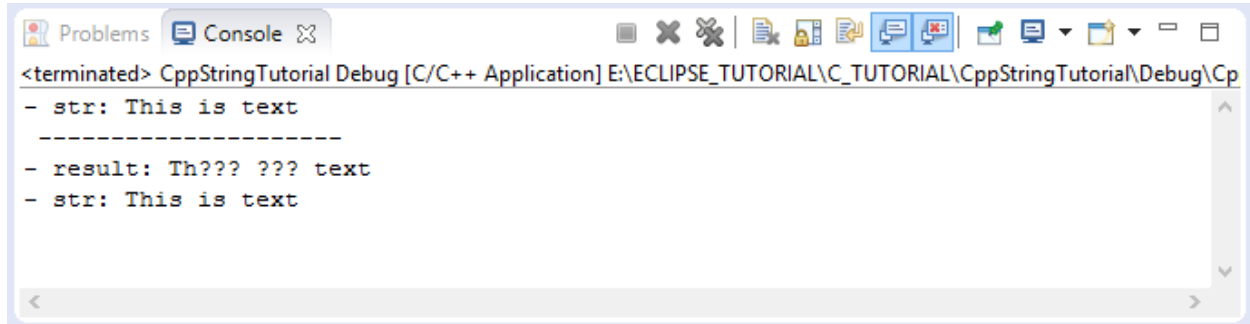
    string str = "This is text";

    cout << "- str: " << str << endl;
    cout << " ----- " << endl;

    // Tìm kiếm các chuỗi con "is" và thay thế bởi "???"
    string result = replaceAll(str,"is", "???");

    cout << "- result: " << result << endl;
    cout << "- str: " << str << endl;
}
```

Kết quả chạy ví dụ:



```
<terminated> CppStringTutorial Debug [C/C++ Application] E:\ECLIPSE_TUTORIAL\C_TUTORIAL\CppStringTutorial\Debug\Cp
- str: This is text
-----
- result: Th??? ??? text
- str: This is text
```

### 2.2.6- insert

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

int main() {

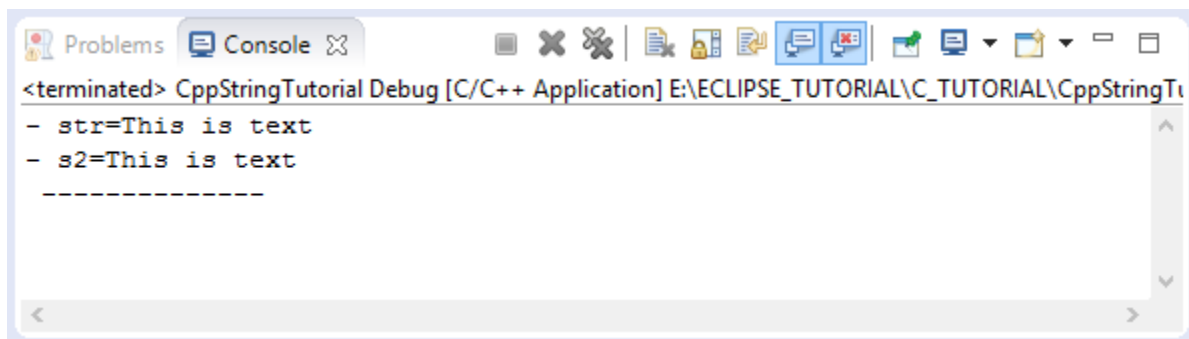
    string str = "This text";

    string s2 = str.insert(5, "is ");

    cout << "- str=" << str << endl; // This is text
    cout << "- s2=" << s2 << endl;    // This is text
    cout << " ----- " << endl;

}
```

Kết quả chạy ví dụ:



```
<terminated> CppStringTutorial Debug [C/C++ Application] E:\ECLIPSE_TUTORIAL\C_TUTORIAL\CppStringTu
- str=This is text
- s2=This is text
-----
```

### 2.2.7- upper/lower

Mặc dù trong lớp *string* không cung cấp cho bạn một phương thức nào để chuyển đổi một chuỗi thành chuỗi chữ hoa hoặc chuỗi chữ thường, nhưng bạn có thể làm điều đó với các thư viện khác trong C++.

Sử dụng hàm **transform** của không gian tên (namespace) **std**:

```
#include <iostream>
#include <algorithm>
#include <string>

using namespace std;

int main() {

    string str = "This is text";

    cout << "- str= " << str << endl;

    cout << " -----" << endl;

    // Chuyển thành chữ hoa
    std::transform(str.begin(), str.end(), str.begin(),
::toupper);

    cout << "- str= " << str << endl;

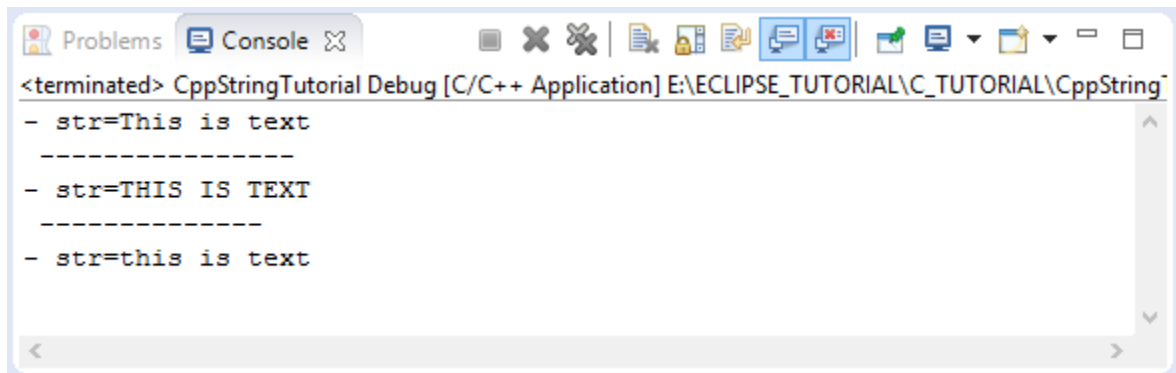
    // Reset str
    str = "This is text";

    cout << " -----" << endl;

    // Chuyển thành chữ thường.
    std::transform(str.begin(), str.end(), str.begin(),
::tolower);

    cout << "- str= " << str << endl;
}
```

Kết quả chạy ví dụ:



```
<terminated> CppStringTutorial Debug [C/C++ Application] E:\ECLIPSE_TUTORIAL\C_TUTORIAL\CppString
- str=This is text
-----
- str=THIS IS TEXT
-----
- str=this is text
```

## Một số bài tập

Nguồn: <http://viettruong92.blogspot.com>

1. Đếm có bao nhiêu khoảng trắng trong chuỗi.
2. Nhập vào một chuỗi, hãy loại bỏ những khoảng trắng thừa trong chuỗi.
3. Nhập vào hai chuỗi s1 và s2, nối chuỗi s2 vào s1. Xuất chuỗi s1 ra màn hình
4. Đổi tất cả các kí tự có trong chuỗi thành chữ thường (không dùng hàm strlwr).
5. Đổi tất cả các kí tự trong chuỗi sang chữ in hoa (không dùng hàm struppr).
6. Viết chương trình đổi những kí tự đầu tiên của mỗi từ thành chữ in hoa.
7. Viết chương trình đổi chữ xen kẽ 1 chữ hoa và 1 chữ thường. Ví dụ: nhập ABCDEfgh đổi thành AbCdEfGh
8. Viết chương trình đảo ngược các kí tự trong chuỗi. Ví dụ: nhập ABCDE, xuất ra màn hình là:EDCBA
9. Viết chương trình tìm kiếm 1 kí tự xem có trong chuỗi không, nếu có xuất ra vị trí của từ chữ kí tự đó. (Vd: xâu a là “ho chi minh”: nhập ‘m’=>kết quả là 3)
10. Viết 1 chương trình đếm một ký tự xuất hiện bao nhiêu lần trong chuỗi.(vd:xâu a nhập là “ho chi minh”, nhập ‘i’ =>kq: 2)
11. Nhập vào chuỗi s1 và s2, cho biết vị trí xuất hiện của chuỗi s2 trong s1.
12. Viết chương trình tìm kiếm tên trong chuỗi họ tên. Nếu có thì xuất ra là tên này đã nhập đúng, ngược lại thông báo là đã nhập sai.
13. Viết chương đảo vị trí của từ đầu và từ cuối. Ví dụ: nhập “bo an co” xuất ra “co an bo”
14. Viết hàm cắt chuỗi họ tên thành chuỗi họ lót và chuỗi tên.  
Ví dụ: chuỗi họ tên là:”Nguyễn Văn A” cắt ra 2 chuỗi là chuỗi họ lót:”NguyễnVăn”,chuỗi tên là:”A”
15. Nhập một chuỗi bất kỳ, sau đó hỏi người dùng cần tách bắt đầu từ đâu trong chuỗi trở về sau.

Ví dụ: Nhập chuỗi S1:”Trường Đại Học Tôn Đức Thắng”. Người nhập muốn tách bắt đầu từ chữ “Tôn” thì sẽ xuất ra chuỗi “Tôn Đức Thắng” ra màn hình

16. Viết hàm kiểm tra xem chuỗi có đối xứng hay không?.
17. Viết hàm tra xem trong chuỗi có kí tự số hay không nếu có tách ra thành một mảng số riêng.
18. Nhập một chuỗi bất kì, yêu cầu nhập 1 kí tự muốn xóa. Thực hiện xóa tất cả những kí tự đó trong chuỗi.
19. Đổi các từ ở đầu câu sang chữ hoa và những từ không phải đầu câu sang chữ thường.

Ví dụ: nGuYen vAN a đổi thành: Nguyen Van A

20. Viết chương trình đảo ngược thứ tự các từ có trong chuỗi

Ví dụ: Nhập: lap trinh bang ngon ngu c

Xuất ra màn hình là: c ngu ngon bang trinh lap

21. Cho chuỗi str, nhập vào vị trí vt và số kí tự cần xóa n, hãy xóa n kí tự tính từ vị trí vt trong chuỗi str.
22. Nhập vào chuỗi str, chuỗi cần chèn strInsert và vị trí cần chèn vt. Hãy chèn chuỗi strInser vào chuỗi str tại vị trí vt.
23. Cho một xâu, nhập vào một từ ,viết chương trình, xóa từ đó trong xâu đã cho.
24. Viết chương trình tìm kiếm xem ký tự nào xuất hiện nhiều nhất trong chuỗi.
25. Nhập 1 chuỗi bất kì, liệt kê xem mỗi ký tự xuất hiện mấy lần.



## Bài giải tham khảo:

1

```
int demkhoangtrang(char *s) {
    int d = 0;
    while (strstr(s, " ") != NULL) {
        d++;
        s = strstr(s, " ") + 1;
    }
    return d;
}
//lam theo ham de quy
// s=strstr(s," ");
//if(s==NULL)
//return 0;
//else return 1+demkhoangtrang(s+1);
```

2

```
void xoakhoangtrang(char *s) {
    char *c = strstr(s, " ");

    while (c != NULL) {
        int t = strlen(s) - strlen(c);
        for (int i = t; i < strlen(s); i++) s[i] = s[i + 1];
        c = strstr(s, " ");
    }
    if (s[0] == ' ')
        s = s + 1; //xoa khang trang dau chuoai
    int n = strlen(s);
    if (s[n - 1] == ' ')
        s[n - 1] = '\0'; //xoa khoang trang cuoi chuoai

    cout << s;
    //for(int i=0;i<n;i++)
    //cout<<s[i];
}
```

3

```
void noichuoi(char *a, char *b) {
    strcat(a, b);
    puts(a);
}
```

4

```
void chuthuong(char *a) {
    for (int i = 0; i < strlen(a); i++)
        if (a[i] >= 65 && a[i] <= 90) a[i] = a[i] + 32;
    puts(a);
}
```

5

```

void chuhoa(char *a) {
    for (int i = 0; i < strlen(a); i++) a[i] = toupper(a[i]);
    //if(a[i]>=97 && a[i]<=122) a[i]=a[i]-32;
    puts(a);
}

```

6

```

void chuhoadau(char *s) {
    s[0] = toupper(s[0]);
    while (strstr(s, " ") != NULL) {
        s = strstr(s, " ") + 1;
        s[0] = toupper(s[0]);
    }
}

```

7

```

void chuxenke(char *a) {
    int n = strlen(a);
    for (int i = 0; i < n; i++) {
        if (i % 2 == 0)
            if ((int) a[i] >= 97 && (int) a[i] <= 122)
                a[i] =char((int) a[i] - 32);
        if (i % 2 == 1)
            if ((int) a[i] >= 65 && (int) a[i] <= 90)
                a[i] =char((int) a[i] + 32);
    }
    puts(a);
}

```

8

```

void daochuoi(char *s) {
    puts(strrev(s));
}

```

9

```

int vitri(char *a, char *b) {
    int kq, d = 0;
    if (strstr(a, b) != NULL) {
        kq = strlen(a) - strlen(strstr(a, b));
        for (int i = kq; i >= 0; i--)
            if (a[i] == ' ')
                d++;
        return d + 1;
    } else return -1;
}

```

10

```
int diemtu(char *a, char *b) {
    int d = 0;
    while (strstr(a, b) != NULL) {
        d++;
        a = strstr(a, b) + 1;
    }

    return d;
}
```

11

```
int vitri(char *a, char *b) {
    int d = -1;
    if (strstr(a, b) != NULL)
        d = strlen(a) - strlen(strstr(a, b));
    return d;
}
```

12

```
void timten(char *a, char *b) {
    int n = strlen(a);
    char *s;
    for (int i = n - 1; i >= 0; i--)
        if (a[i] == ' ') {
            s = a + i + 1;
            break;
        }
    if (strcmpi(s, b) == 0) // strcmp==strcmp
        cout << s;
    else
        cout << "ban nhap sai ten\n";
}
```

13

```
void daochuoi(char *s) {
    char t[50], r[50];
    int i, l;
    for (i = strlen(s) - 1; i >= 0; i--)
        if (s[i] == ' ')
            break;
    strcpy(r, s + i + 1); //tim dctu cuoi
    s[i + 1] = '\0'; //xau s sau khi bo tu cuoi, co khoang trang o cuoi xau
    l = strlen(s) - strlen(strstr(s, " ")); //tim vi tri khoang trang dau
    tien trong chuoi
    strcpy(t, s + l + 1); //l la xau chau cac tu chinh giua
    s[l] = '\0'; //hien tai la xau chua tu dau tien
    strcat(strcat(strcat(r, " "), t), s); //noi cac xau lai voi nhau*/

    puts(r);
}
```

14

```
void cathoten(char *s) {
    char *t;
    int i;
    for (i = strlen(s) - 1; i >= 0; i--)
        if (s[i] == ' ' && s[i + 1] != ' ') break;
    t = s + i + 1;
    s[i] = '\\0';
    cout << "ho lot: ";
    puts(s);
    cout << "ho ten: ";
    puts(t);
}
```

15

```
void timtu(char *a, char *b) {

    char *t = strstr(a, b); //vi tri xuat hien b dau tien trong chuoai a
    puts(t);
}
```

16

```
bool ktdoixung(char *s) {
    char c[255];
    strcpy(c, s);
    if (strcmpi(s, strrev(c)) == 0) return true;
    else return false;
}
```

17

```
void tachso(char *s) {
    int a[100], j = 0;
    for (int i = 0; i < strlen(s); i++)
        if (s[i] >= '0' && s[i] <= '9') {
            a[j] = s[i]; //ki tu kieu char chuyen thanh kieu int(vd:1=>49)
            j++;
            for (int t = i; t < strlen(s); t++) s[t] = s[t + 1];
            i--;
        }
    puts(s);
    for (int i = 0; i < j; i++) cout << char(a[i]) << " ";
}
```

18

```
void xoakitu(char *a, char b) {
    int n = strlen(a);
    for (int i = 0; i < n; i++)
        if (a[i] == b) {
            for (int j = i; j < n; j++) a[j] = a[j + 1];
            i--;
            n--;
        }

    cout << a;
}
```

19

```
void chuhoa(char *s) {
    s[0] = toupper(s[0]);
    for (int i = 1; i < strlen(s); i++) //chua tim dc cach nao toi uu hon
        if(s[i]!=' ')
        {
            s[i + 1] = toupper(s[i + 1]);
            i++;
        } else
        {
            if (s[i] >= 'A' && s[i] <= 'Z') s[i] = s[i] + 32;
        }
    puts(s);
}
```

20

```
void daothutu(char *s) {
    char c[ 255];
    c[0] = '\\0';//ham strcat noi chuoì tai vi tri NULL

    for (int i = strlen(s) - 1; i >= 0; i--)
        if (s[i] == ' ') {
            strcat(strcat(c, s + i + 1), " ");
            s[i] = '\\0';
        }

    strcat(c, s);
    puts(c);
}
```

21

```
void xoa(char *s, int vt, int n) {
    strcpy(s + vt, s + vt + n);
    puts(s);
}
```

22

```
void chen(char *s, char *d, int vt) {
    char c[ 255];
    strcpy(c, s + vt);
    strcpy(s + vt, d);
    strcat(s, c);
    puts(s);
}
```

23

```
void xoatu(char *s, char *c) {
    while (strstr(s, c) != NULL) {
        int t = strlen(s) - strlen(strstr(s, c)), d = strlen(c);
        if ((s[t - 1] == ' ' && s[t + d] == ' ') || (s[t - 1] == ' ' && t + d
== strlen(s)))
            strcpy(s + t - 1, s + t + d);
        //truong hop tu giua va tu cuoi
        // s+t-1 la khoang trang
        if (s[t + d] == ' ' && t == 0) strcpy(s, s + t + d + 1);
        //truong hop xoa tu dau tien
    }
    puts(s);
}
```

24

```
void kituxuathienhieunhat(char *s) {
    int a[ 100],n = 0;
    for (int i = 0; i < (int) strlen(s); i++) {
        int d = 1;
        for (int j = i + 1; j < (int) strlen(s); j++)
            if (s[i] == s[j]) {
                d++;
                for (int k = j; k < (int) strlen(s); k++) s[k] = s[k + 1];
                j--;
            }
        a[n++] = d;
    }
    int max = a[0];

    for (int i = 1; i < (int) strlen(s); i++) if (a[i] > max) max = a[i];
    for (int i = 1; i < (int) strlen(s); i++)
        if (a[i] == max) cout << "ki tu " << s[i] << " xuất hiện nhiều nhất
là "<<a[i]<<" lần\n ";
}
```

25

```
void demkitu(char *s) {
    int a[ 100],n = 0;
    for (int i = 0; i < (int) strlen(s); i++) {
        int d = 1;
        for (int j = i + 1; j < (int) strlen(s); j++)
            if (s[i] == s[j]) {
                d++;
                for (int k = j; k < (int) strlen(s); k++) s[k] = s[k + 1];
                j--;
            }
        a[n++] = d;
    }
    for (int i = 0; i < (int) strlen(s); i++)
        cout << "ki tu " << s[i] << " xuất hiện " << a[i] << " lần\n";
}
```