Nguồn : Vnoi.info

**0. Kiểu chuỗi của C và hạn chế**

Khi mới học C, chắc các bạn đều rất bối rối khi làm việc với xâu ký tự, việc sử dụng con trỏ lưu xâu ký tự rất phức tạp, dễ gây lỗi khiến nhiều người cho rằng nó không bằng xâu ký tự trong Pascal.

Các chương trình C++ có thể sử dụng chuỗi theo cách thức cũ của Ngôn ngữ C (trong bài viết này gọi là C-String): mảng các ký tự kết thúc bởi ký tự mã ASCII là 0 (ký tự \0) cùng với các hàm thư viện khai báo trong <string.h>. Có nhiều bất tiện khi dùng theo cách thức này:

* Người lập trình phải chủ động kiểm soát bộ nhớ cấp phát cho chuỗi ký tự. Nói chung là phải am hiểu và rất thông thạo về kỹ thuật dùng bộ nhớ và con trỏ thì chương trình mới tránh được các lỗi về kỹ thuật;
* Không thể gán giá trị hay sử dụng phép toán + (ghép chuỗi) và các phép toán so sánh như: > (lớn hơn), < (nhỏ hơn),… mà phải gọi các hàm thư viện trong <string.h>;
* Nếu dùng kỹ thuật cấp phát động thì phải quản lý việc cấp thêm bộ nhớ khi chuỗi dãn ra (chẳng hạn do ghép chuỗi) và phải hủy bộ nhớ (khi không dùng nữa) để tránh việc cạn kiệt bộ nhớ của máy tính trong trường hợp có nhiều chương trình hoạt động đồng thời.

**1. Kiểu chuỗi string trong thư viện STL của C++**

Thư viện chuẩn STL (Standard Template Library) cung cấp kiểu string (xâu ký tự), giúp các bạn tránh khỏi hoàn toàn các phiền phức nêu trên.Các chỉ thị #include cần khai báo để sử dụng string:

**#include <string>**

**using** std**::**string;

*//using namespace std;*

Nếu bạn muốn tận dụng các hàm của C-String, cần chuyển đổi giữa 2 kiểu dữ liệu này:

* Chuyển từ string sang C-String:
* string s;

s.c\_str();

* Chuyển từ C-String sang string:
* **char\*** s\_old **=** "ABC";

string **s**(s\_old);

**2. Các phương thức, phép toán tiện ích của kiểu string**

Kiểu string của STL hỗ trợ các nhóm phương thức và phép toán tiện ích sau đây.

**a) Các phép toán và phương thức cơ bản**

* Các toán tử +, += dùng để ghép hai chuỗi và cũng để ghép một ký tự vào chuỗi;
* Các phép so sánh theo thứ tự từ điển: == (bằng nhau), != (khác nhau), > (lớn hơn), >= (lớn hơn hay bằng), < (nhỏ hơn), <= (nhỏ hơn hay bằng);
* Hàm length() và phép lấy chỉ số [] để duyệt từng ký tự của chuỗi: nếu s là biến kiểu string thì s[i] là ký tự thứ i của s với 0≤i<s.length()0≤i<s.length();
* Phép gán = dùng để gán biến kiểu string bằng một chuỗi, hoặc bằng string khác, chẳng hạn: string s="ABCDEF"; hay s1=s2; mà không cần copy xâu.
* Những constructor thường sử dụng nhất:
* string();
* string(**const** **char** **\***str); *// char\* là kiểu dữ liệu xâu của C*

string(**const** string **&** str);

* Có thể dùng toán tử << với cout để xuất một chuỗi ra màn hình hoặc dùng toán tử >> với cin để nhập một chuỗi ký tự đến khi gặp một khoảng trống thì dừng.
* **char** st[]**=**"ABCDEF";
* string s;
* s**=**"XYZ";
* cout **<<** s **<<** endl;
* s**=**st;

cout **<<** s.length() **<<** " : " **<<** s **<<** endl;

* Một vấn đề thường nảy sinh trong các ứng dụng có sử dụng C-string: một C-String chưa khởi tạo cần được gán NULL. Tuy nhiên, rất nhiều hàm thư viện của C-String sẽ gặp sự cố trong thời gian chạy khi gặp đối tượng C-String là NULL. Chẳng hạn, lệnh
* **char\*** x **=** NULL;

cout **<<** strlen(x);

được một số trình biên dịch chấp nhận, nhưng với nhiều hiện thực khác của thư viện C-String, thì gặp lỗi trong thời gian chạy. string không gặp vấn đề này, ta hoàn toàn có thể cho 1 xâu là rỗng mà không gặp bất cứ lỗi nào: string s="";

String thực chất là một vector<char> có bổ sung thêm một số hàm và thuộc tính, do đó, nó có toàn bộ các tính chất của 1 vector, như hàm size(), push\_back(), toán tử [], …

* Các hàm từ vector:
  + v.size(): Số lượng phần tử
  + v.empty(): Trả về 1 nếu chuỗi rỗng, 0 nếu ngược lại.
  + v.max\_size(): Trả về số lượng phần tử tối đa đã được cấp phát
  + v1 == v2: Trả về 1 nếu hai chuỗi giống nhau
  + v1 != v2: Trả về 1 nếu hai chuỗi khác nhau
  + v.begin(): Trả về **iterator** đầu tiên của chuỗi
  + v.end(): Trả về iterator cuối cùng của chuỗi (trỏ vào sau kí tự cuối cùng)
  + v.front(): Trả về phần tử đầu tiên của chuỗi
  + v.back(): Trả về phần tử cuối cùng của chuỗi
  + v1.swap(v2): Hoán đổi 2 chuỗi với nhau (giống việc hoán đổi giá trị của 2 biến)
* **#include <iostream>**
* **#include <conio.h>**
* **#include <string>**
* **using** **namespace** std;
* **int** **main**()
* {
* string s **=** "Hello string"; *// Khai báo biến kiểu string*
* cout **<<** "Noi dung string: " **<<** s **<<** endl; *// In nôi dung string ra màn hình*
* cout **<<** "Chieu dai cua string: " **<<** s.size() **<<** endl;
* *// Chiều dài*
* cout **<<** "Ky tu 0: " **<<** s[0] **<<** endl; *// In ký tự đầu tiên của xâu*
* cout **<<** "Ky tu 1: " **<<** s[1] **<<** endl; *// In ký tự thứ 2*
* cout **<<** "Ky tu 2: " **<<** s[2] **<<** endl; *// In ký tự thứ 3*
* getchar();
* **return** 0;

}

* Nhập một string trên 1 dòng (chú ý cin sẽ chỉ đọc đến dấu cách hoặc xuống dòng đầu tiên): istream& getline ( istream& in, string& str, char delimiter = ‘\n’);

Đọc 1 dòng văn bản từ istream in (có thể là file hay đối tượng chuẩn cin) từng ký tự đến khi ký tự delimiter được nhập vào (mặc định là \n)

*// getline with strings*

**#include <iostream>**

**#include <string>**

**using** **namespace** std;

**int** **main** ()

{

string str;

**short** age;

cout **<<** "Please enter full name and age"**<<** endl;

getline( cin, str) **>>** age;

cout **<<** "Thank you " **<<** str **<<** "!\n";

**return** 0;

}

**b) Các phương thức chèn, xóa, lấy chuỗi con:**

* Phương thức substr(int pos, int nchar) trích ra chuỗi con của một chuỗi cho trước, ví dụ str.substr(2,4) trả về chuỗi con gồm 4 ký tự của chuỗi str kể từ ký tự ở vị trí thứ 2 (ký tự đầu tiên của chuỗi ở vị trí 0).
* *//get substring*
* **#include <iostream>**
* **#include <string>**
* **#include <conio.h>**
* **using** **namespace** std;
* **int** **main** ()
* {
* string s**=**"ConCho chay qua rao";
* cout **<<** s.substr(2,4) **<<** endl;
* *// cout << new string(str.begin()+2, str.begin()+2+4);*
* getchar();
* **return** 0;

}

* Phương thức insert() chèn thêm ký tự hay chuỗi vào một vị trí nào đó của chuỗi str cho trước. Có nhiều cách dùng phương thức này:
  + str.insert(int pos, char\* s); chèn s (mảng ký tự kết thúc \0) vào vị trí pos của str;
  + str.insert(int pos, string s); chèn chuỗi s (kiểu string) vào vị trí pos của chuỗi str;
  + str.insert(int pos, int n, int ch); chèn n lần ký tự ch vào vị trí pos của chuỗi str;
* *// inserting into a string*
* **#include <iostream>**
* **#include <string>**
* **#include <conio.h>**
* **using** **namespace** std;
* **int** **main** ()
* {
* string str**=**"day la .. xau thu";
* string istr **=** "them";
* str.insert(8, istr);
* cout **<<** str **<<** endl;
* getchar();
* **return** 0;

}

* Phương thức str.erase(int pos, int n) xóa n ký tự của chuỗi str kể từ vị trí pos; nếu không quy định giá trị n thì tất cả các ký tự của str từ vị trí pos trở đi sẽ bị xóa
* *// erase from a string*
* **#include <iostream>**
* **#include <string>**
* **#include <conio.h>**
* **using** **namespace** std;
* **int** **main** ()
* {
* string str**=**"day cung la xau thu";
* str.erase(0, 3); *// " cung la xau thu"*
* cout **<<** str **<<** endl;
* str.erase(6, 2);
* cout **<<** str **<<** endl; *// " cung xau thu"*
* getchar();
* **return** 0;

}

**c) So sánh**

Bạn có thể đơn giản là sử dụng những toán tử quan hệ (==, !=, <, <=, >=) được định nghĩa sẵn. Tuy nhiên, nếu muốn so sánh một phần của một chuỗi thì sẽ cần sử dụng phương thức compare():

**int** compare ( **const** string**&** str ) **const**;

**int** compare ( **const** **char\*** s ) **const**;

**int** compare ( **size\_t** pos1, **size\_t** n1, **const** string**&** str ) **const**;

**int** compare ( **size\_t** pos1, **size\_t** n1, **const** **char\*** s) **const**;

**int** compare ( **size\_t** pos1, **size\_t** n1, **const** string**&** str, **size\_t** pos2, **size\_t** n2 ) **const**;

**int** compare ( **size\_t** pos1, **size\_t** n1, **const** **char\*** s, **size\_t** n2) **const**;

Hàm trả về 0 khi hai chuỗi bằng nhau và lớn hơn hoặc nhỏ hơn 0 cho trường hợp khác Ví dụ:

*// comparing apples with apples*

**#include <iostream>**

**#include <string>**

**using** **namespace** std;

**int** **main** ()

{

string str1 ("green apple");

string str2 ("red apple");

**if** (str1.compare(str2) **!=** 0)

cout **<<** str1 **<<** " is not " **<<** str2 **<<** "\n";

**if** (str1.compare(6,5,"apple") **==** 0)

cout **<<** "still, " **<<** str1 **<<** " is an apple\n";

**if** (str2.compare(str2.size()**-**5,5,"apple") **==** 0)

cout **<<** "and " **<<** str2 **<<** " is also an apple\n";

**if** (str1.compare(6,5,str2,4,5) **==** 0)

cout **<<** "therefore, both are apples\n";

**return** 0;

}

**d) Các phương thức tìm kiếm và thay thế**

* Phương thức find() tìm kiếm xem một ký tự hay một chuỗi nào đó có xuất hiện trong một chuỗi str cho trước hay không. Có nhiều cách dùng phương thức này:
* str.find(**int** ch, **int** pos **=** 0); tìm ký tự ch kể từ vị trí pos đến cuối chuỗi str
* str.find(**char** **\***s, **int** pos **=** 0); tìm **s** (mảng ký tự kết thúc ‘\0’) kể từ vị trí pos đến cuối

str.find(string**&** s, **int** pos **=** 0); tìm chuỗi s kể từ vị trí pos đến cuối chuỗi.

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.

*//find substring*

**#include <iostream>**

**#include <string>**

**#include <conio.h>**

**using** **namespace** std;

**int** **main** ()

{

string str**=**"ConCho chay qua rao";

cout **<<** str.find("chay") **<<** endl; *// 7*

cout **<<** (**int**)str.find("Chay") **<<** endl; *// -1*

getchar();

**return** 0;

}

* Hàm tìm kiếm ngược (rfind)
* *//find from back*
* **#include <iostream>**
* **#include <string>**
* **#include <conio.h>**
* **using** **namespace** std;
* **int** **main** ()
* {
* string str**=**"ConCho chay qua chay qua rao";
* cout **<<** str.find("chay") **<<** endl; *// 7*
* cout **<<** (**int**)str.rfind("chay") **<<** endl; *// 16*
* getchar();
* **return** 0;

}

* Phương thức replace() thay thế một đoạn con trong chuỗi str cho trước (đoạn con kể từ một vị trí pos và đếm tới nchar ký tự ký tự về phía cuối chuỗi) bởi một chuỗi s nào đó, hoặc bởi n ký tự ch nào đó. Có nhiều cách dùng, thứ tự tham số như sau:
* str.replace(**int** pos, **int** nchar, **char** **\***s);
* str.replace(**int** pos, **int** nchar, string s);
* str.replace(**int** pos, **int** nchar, **int** n, **int** ch);
* string str**=**"con cho la con cho con. Con meo ko phai la con cho";
* str.replace(4, 3, "CHO"); *// "con CHO la con cho con. Con meo ko phai la con cho";*
* cout **<<** str **<<** endl;

getchar();

**e) Tách xâu**

Trong việc xử lý xâu ký tự, không thể thiếu được các thao tác tách xâu ký tự thành nhiều xâu ký tự con thông qua các ký tự ngăn cách. Các hàm này có sẵn trong các ngôn ngữ khác như Visual Basic, Java, hay thậm chí là trong <string.h>. Với STL, các bạn có thể dễ dàng làm điều này với stringstream:

string S **=** "Xin chao tat ca cac ban"; *// Khởi tạo giá trị của xâu*

stringstream **ss**(S); *// Khởi tạo stringstream từ xâu S*

**while** (ss **>>** token) { *// Đọc lần lượt các phần của xâu. Các phần tách nhau bởi dấu cách hoặc xuống dòng.*

cout **<<** token **<<** endl;

}

Output:

Xin

chao

tat

ca

cac

ban

Chú ý rằng, cách này cũng có thể dễ áp dụng nếu bạn muốn chuyển số thành xâu (hoặc ngược lại), tách 1 xâu thành nhiều số.

Nếu không muốn sử dụng stringstream, các bạn cũng có thể tự xây dựng hàm tách xâu như sau:

string S **=** "Xin chao tat ca cac ban"; *// Khởi tạo giá trị của xâu*

string**::**iterator t, t2; *// Các biến lặp*

vector**<**string**>** split; *// Mảng các xâu (lưu kết quả tách)*

**for** (t**=**S.begin(); t**<**S.end();)

{

*// Lặp từ vị trí bắt đầu*

t2**=**find(t, S.end(), ' '); *// TÌm ký tự space ' ' đầu tiên*

*// kể từ vị trí t*

**if** (t**!=**t2) split.push\_back(string(t, t2)); *// Lấy xâu ký tự giữa 2 vị trí*

t **=** t2**+**1; *// Chuyển sang vị trí sau*

}

**for** (**int** i**=**0; i**<**splitìsize(); i**++**)

cout **<<** split[i] **<<** endl; *// In mảng các xâu ký tự*

getchar();

Output:

Xin

chao

tat

ca

cac

ban

Đoạn chương tr.nh sử dụng các kỹ thuật sau

* Phương thức find(vị\_trí\_đầu, vị\_trí\_cuối, ký\_tự\_tìm) dùng để tìm vị trí đầu tiên của ký\_tự\_tìm bắt đầu từ vị\_trí\_đầu. Hàm này trả về vị trí của ký tự tìm được (nếu tìm thấy) hoặc vị\_trí\_cuối (nếu không tìm thấy)
* string có thể khởi tạo từ một đoạn ký tự con của một xâu ký tự khác với cú pháp string(vị\_trí\_đầu, vị\_trí\_cuối)
* Đoạn chương trình thực hiện tách các xâu ký tự kể cả trong trường hợp có nhiều ký tự space nằm liên tiếp nhau. Một cách đơn giản hơn là bạn có thể gọi hàm strtok() trong string.h để làm việc này, nhưng không may là hàm này thao tác trên char\* chứ không phải string. Hàm thành viên c\_str() sẽ giúp bạn chuyển từ string thành dạng const charT\* c\_str () const;

Hàm này cũng tự động sinh ra ký tự null chèn vào cuối xâu.

Từ prototype ta cũng thấy được hàm trả về một hằng chuỗi, điều này đồng nghĩa với việc ta không thể thay đổi chuỗi trả về. Gọi phương thức c\_str();

string s **=** "some\_string";

cout **<<** s.c\_str() **<<** endl;

cout **<<** strlen(s.c\_str()) **<<** endl;

Sau đây là ví dụ bên trên được viết lại dùng hàm thành viên c\_str() và các hàm trong <string.h>

*// strings vs c-strings*

**#include <iostream>**

**#include <string.h>**

**#include <string>**

**using** std**::**string;

**int** **main** ()

{

**char\*** cstr;

**char\*** p;

string str ("Xin chao tat ca cac ban");

cstr **=** **new** **char** [str.size()**+**1];

strcpy (cstr, str.c\_str());

*// cstr là 1 bản sao c-string của str*

p**=**strtok (cstr," ");

**while** (p**!=**NULL)

{

cout **<<** p **<<** endl;

p**=**strtok(NULL," ");

}

**delete**[] cstr;

**return** 0;

}

Output:

Xin

chao

tat

ca

cac

ban

**f) Chuyển đổi hàng loạt với transform**

OutputIterator **transform**( InputIterator first,

InputIterator last,

OutputIterator result,

UnaryOperation unary\_op );

**#include <cctype> // for toupper**

**#include <string>**

**#include <algorithm> //for transform**

**using** **namespace** std;

**char** **alphabet**(**char** c)

{

**static** **char** ch **=** 'a';

**return** ch**++**;

}

**int** **main**()

{

string s("this is a lower case string");

transform(s.begin(), s.end(), s.begin(), toupper);

cout **<<** s **<<** endl;

transform(s.begin(), s.end(), s.begin(), alphabet);

cout **<<** s;

**return** 0;

}

**g) Một số phương thức khác**

Còn nhiều phương thức tiện ích khác như: append(), rfind(), find\_first\_not\_of(), find\_last\_not\_of(), swap(). Cách dùng các hàm này đều được trình bày trong hệ thống hướng dẫn (help) của các môi trường có hỗ trợ STL (trong VC++ là MSDN). Ngoài ra các phương thức như find\_first\_of() tương tự như find(), find\_last\_of() tương tự như rfind()