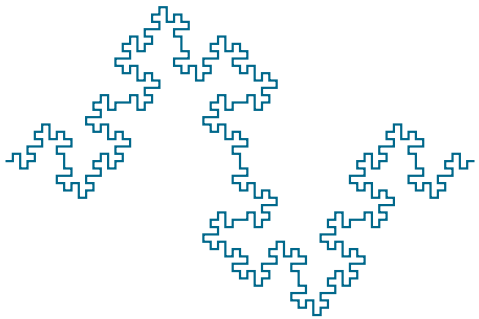
Lớp String của thư viện chuẩn STL C++

Trần Việt Khoa
HUE UNIVERSITY OF SCIENCES



Ngày 28 tháng 12 năm 2020

Giới thiệu

- C++ định nghĩa một dãy các ký tự thành một một đối tượng gọi là chuỗi (xâu) thuộc lớp std::string. Lớp string lưu các ký tự dưới dạng một chuỗi các byte và cho phép truy cập vào từng byte một của chuỗi.
- So sánh string và mảng ký tự.
  - Mảng ký tự lưu các ký tự liên tiếp nhau và kết thúc bằng ký tự biên (null). Lớp string định nghĩa các đối tượng dưới dạng luồng (stream) ký tự.
  - Mảng ký tự lưu trữ static với bộ nhớ cố định dẫn đến tình trạng thiếu hoặc thừa. String cấp phát động nên không bị tình trạng trên và có thể lưu lớn hơn.
  - Mảng ký tự xử lý nhanh hơn string.
  - Mảng ký tự không có nhiều hàm dựng sẵn để thao tác chuỗi như lớp string.

Các hàm input

1. getline(): Người dùng nhập một luồng (stream) các ký tự lưu vào bộ nhớ đối tượng. stream có thể chứa ký tự trắng khác với phương thức dùng cin hoặc scanf("%s",&str);
2. str.push_back(): Được sử dụng để nhập vào một ký tự vào cuối chuỗi *str*.
3. str.pop_back(): Có từ bản dịch C++11 [1](for strings), sử dụng xóa ký tự cuối của chuỗi *str*.

[1] g++ -std=c++11 yourfile.cpp -o yourprogram

Ví dụ về hàm input

```
#include <iostream>
using namespace std;
int main()
{
// Khai báo đối tượng thuộc lớp string.
string str;
getline(cin, str);
cout << str << endl;
return 0;
}
```

- Input: Programming is fun
- Output: Programming is fun
- Hàm getline(cin, str) chuyển dữ liệu từ luồng (cin) vào đối tượng str.

Ví dụ về hàm input

```
#include <iostream>
using namespace std;
int main()
{
// Khai báo đối tượng thuộc lớp string.
string str;
for (int i='a'; i<='z'; ++i)
        str.push_back(i);
str.pop_back();
cout << str << endl;
return 0;
}
```

- Output: abcdefghijklmnopqrstuvwxy
- Hàm push_back(i) đẩy các ký tự i='a' đến 'z' vào đối tượng str.

Các hàm về kích thước

1. str.capacity(): Hàm này trả về dung lượng được phân bổ cho chuỗi, có thể bằng hoặc lớn hơn kích thước của chuỗi *str*. Không gian bổ sung được phân bổ để khi các ký tự mới được thêm vào chuỗi, các hoạt động có thể được thực hiện một cách hiệu quả.
2. str.resize(): Hàm này thay đổi kích thước của chuỗi *str*, kích thước có thể tăng hoặc giảm.
3. str.length(): Hàm này trả về độ dài của chuỗi *str*.

VÍ DỤ VỀ HÀM KÍCH THƯỚC

```
#include<iostream>
using namespace std;
int main()
{
        // Khai báo và khởi tạo chuỗi bằng phép gán.
        string str = "Programming is Fun";
        // Hiển thị chuỗi.
        cout << "The initial string is: "; cout << str << endl;
        // Thay đổi lại kích thước chuỗi bằng resize().
        str.resize(11);
        cout << "The string after resize operation is : ";
        cout << str << endl;
        // in khả năng chứa của chuỗi.
        cout << "The capacity of string is : ";
        cout << str.capacity() << endl;
        //in chiều dài chuỗi.
        cout<<"The length of the string is :"<<str.length()<<endl;
return 0;
}
```

- Output:
  The initial string is: Programming is Fun
  The string after resize operation is : Programming
  The capacity of string is : 18
  The length of the string is :11

Các hàm về duyệt chuỗi

1. str.begin(): Hàm trả về một con trỏ (iterator) đến đầu chuỗi *str*.
2. str.end(): Hàm trả về một con trỏ (iterator) đến cuối chuỗi *str*.
3. str.rbegin(): Hàm trả về một con trỏ ngược (iterator) đến cuối chuỗi *str*.
4. str.rend(): Hàm trả về một con trỏ ngược (iterator) đến đầu chuỗi *str*.

Ví dụ về hàm duyệt

```
#include<iostream>
using namespace std;
int main()
{
// Khai báo và khởi tạo chuỗi.
string str = "Dogma I am God";
// Khai báo biến lặp.
string::iterator it;
// Declaring reverse iterator
string::reverse_iterator it1;
// Duyệt in các ký tự của chuỗi nhờ biến lặp.
cout << "The string using forward iterators is : ";
for (it=str.begin(); it!=str.end(); it++)
        cout << *it;
cout << endl;
// Duyệt chiều ngược lại bằng con trỏ ngược.
cout << "The reverse string using reverse iterators is : ";
for (it1=str.rbegin(); it1!=str.rend(); it1++)
        cout << *it1;
cout << endl;
return 0;
}
```

- Output:
  The string using forward iterators is : Dogma I am God
  The reverse string using reverse iterators is : doG ma I amgoD

Các toán tử truy cập

1. string::operator[]: Phép toán cho phép truy cập từng ký tự của chuỗi bằng chỉ mục qua toán tử [].
2. string::at: Hàm này tương tự toán tử tải bội trên.
3. str.length(): Hàm này trả về độ dài của chuỗi *str*.

Ví dụ về toán tử và hàm truy cập

```
#include <iostream>
using namespace std;
int main ()
{
        //Khai báo và dựng (hàm dựng) đối tượng.
        string str ("Programming is Fun");
        //Duyệt in bằng chỉ mục với toán tử [].
        //tương tự mảng chỉ số i=0..str.length()-1.
        for (unsigned i=0; i<str.length(); ++i)
                cout << str[i];
        //Duyệt in bằng chỉ mục với hàm at().
        for (unsigned i=0; i<str.length(); ++i)
                cout << str.at(i);
return 0;
}
```

- Output:
  Programming is Fun
  Programming is Fun

Các hàm xử lý chuỗi

1. str.find(ch, pos): Hàm tìm ký tự *ch* tính từ vị trí *pos* đến cuối chuỗi *str* và về vị trí xuất hiện đầu tiên nếu tìm thấy và -1 nếu không tìm thấy. Tương tự str.find(string& s, pos) cho việc tìm chuỗi *s* trong chuỗi *str*.
2. str.substr(pos, num): Trả về chuỗi con có *num* ký tự tính từ vị trí *pos* lấy từ xâu *str*.
3. str.insert(pos, s): Chèn xâu *s* tại vị trí chèn *pos* vào xâu gọi phương thức *str*.
4. str.erase(pos, num): Hàm xóa *num* ký tự từ vị trí *pos* của chuỗi *str*.
5. str.replace(): Hàm thay thế chuỗi, tham khảo [2].

[2] http://www.cplusplus.com/reference/string/string/replace/

Ví dụ về hàm find() và substr()

```
#include <iostream>
using namespace std;
int main ()
{
        string str="We think in generalities, but we live in details.";
        //Cắt từ think ở trong chuỗi, vị trí bắt đầu 3 độ dài 5.
        string str1 = str.substr (3,5);
        //Tìm vị trí từ live trong chuỗi.
        size_t pos = str.find("live");
        //Cắt xâu con bắt đầu từ từ live đến cuối chuỗi.
        string str2 = str.substr (pos);
        //In chuỗi str1 và str2.
        cout << str1 << ' ' << str1 << '\n';
        return 0;
}
```

- Output:
  think live in details.

Ví dụ về hàm find() và erase()

```
#include <iostream>
using namespace std;
void eraseAllSubStr(string & mainStr, const string & toErase){
        size_t pos = string::npos;
        // Lặp việc tìm xâu con trong xâu cho đến khi không tìm thấy.
        while ((pos  = mainStr.find(toErase) )!= string::npos)
                // nếu tìm thấy thì xóa.
                mainStr.erase(pos, toErase.length());
}
int main() {
        string s = "One fish, two fish, red fish, blue fish, Nemo fish";
        string p = "fish";
        eraseAllSubStr(s,p);
        cout << s << '\n';
        return 0;
}
```

- Output:
  One , two , red , blue , Nemo
- string::npos là một hằng tĩnh là giá trị lớn nhất cho một phần tử có kiểu với kích thước size_t. Giá trị này, khi được sử dụng làm giá trị cho tham số len (hoặc sublen) trong các hàm thành viên của chuỗi, có nghĩa là "cho đến khi kết thúc chuỗi". Là một giá trị trả về, nó thường được sử dụng để chỉ ra không có kết quả khớp.

Ví dụ về hàm replace()

```
#include <iostream>
#include <algorithm>
using namespace std;
int main()
{
        string s = "C**";
        const char x = '*';
        const char y = '+';
        //Thay thế các ký tự * bằng + trong chuỗi.
        replace(s.begin(), s.end(), x, y);
        cout << s;
        return 0;
}
```

- Output:
  C++

Các toán tử xử lý chuỗi

- Lớp string thực chất là một vector<char> có bổ sung thêm một số phương thức và thuộc tính. Do đó, nó có toàn bộ tính chất của một vector.
- Lớp string cung cấp một số phép toán tải bội để xử lý chuỗi như: nối chuỗi, so sánh, gán.
- Trong C++, bạn vẫn có thể dùng kiểu char* nếu muốn. Có thể chuyển từ kiểu string sang chuỗi char* bằng phương thức c_str().
- Các phương thức sau tượng tự như lớp vector.

Các toán tử trên chuỗi

- Phép toán +: dùng để nối chuỗi, ví dụ: string str="Hello"+ ""+ "world!"; Tương tự như vậy với phép toán +=.
- Phép toán =: phép gán dùng để gán chuỗi.
- Các phép toán quan hệ: <, >, <=, >=, ==, !=. Trả về true nếu biểu thức quan hệ nhận giá trị đúng và false nếu ngược lại.

Bài toán 1. Chuyển đổi hoa thường

1. Viết hàm chuyển đổi một chuỗi sang chữ hoa.
2. Viết hàm chuyển đổi một chuỗi sang chữ thường.
3. Bài toán này có thể giải bằng rất nhiều cách, tuy nhiên theo quan điểm dùng STL phương án sau được đánh giá rất tốt. Sử dụng hàm transform() trong thư viện <algorithm> để chuyển

Hàm chuyển đổi hoa thường

```
#include<bits/stdc++.h>
using namespace std;
string StringToUpper(string str){
        transform(str.begin(), str.end(), str.begin(), ::toupper);
        return str;
}
string StringToLower(string str){
        transform(str.begin(), str.end(), str.begin(), ::tolower);
        return str;
}
int main() {
        string data = "This is a sample string";
        string str1=StringToUpper(data); cout <<str1<<endl;
        string str2=StringToLower(str1); cout <<str2<<endl;
        return 0;
}
```

- Output:
  THIS IS A SAMPLE STRING
  this is a sample string

Bài toán 2. Tách tên ra khỏi họ và tên

1. Viết hàm chuyển đổi họ và tên của một người thành Tên + họ
2. Ví dụ: Trần Việt Khoa –> Khoa Trần Việt
3. Ý tưởng: tìm ký tự trắng cuối cùng (dĩ nhiên chuỗi dữ liệu ban đầu đã xóa hết ký tự trắng thừa), sau đó copy chuỗi con đến cuối và ghép lại.

Hàm cắt tên trong họ tên

```
#include <iostream>
#include <string>
using namespace std;
string lastName(string fullName){
        string last;
        int idx=fullName.find_last_of(" ");
        last = fullName.substr(idx+1);
        return last+" "+fullName.substr(0, idx);
}
int main() {
        string fullName="Tran Viet Khoa";
        cout<<lastName(fullName);
        return 0;
}
```

- Output:
  Khoa Tran Viet

Bài toán 3. Xóa ký tự trắng thừa trong chuỗi

1. Viết hàm xóa ký tự trắng thừa ờ đầu, giữa và cuối của chuỗi.
2. Ví dụ: st="Trần Việt Khoa -> st="Trần Việt Khoa".
3. Ý tưởng:
   - Xóa các ký tự trắng ở đầu chuỗi (giống hàm Ltrim của Excel).
   - Xóa các ký tự trắng ở cuối chuỗi (giống hàm Rtrim của Excel).
   - Xóa các ký tự trắng ở giữa bằng cách tìm hai ký tự trắng sát nhau và xóa một trong chúng.

Xóa ký tự trắng thừa trong chuỗi

```
#include <bits/stdc++.h>
using namespace std;
string traiteFullName(string &str){
        const char* t = " \t";
        //Xóa ký tự trắng đầu Ltrim.
        str.erase(0, str.find_first_not_of(t));
        //Xóa ký tự trắng cuối Rtrim.
        str.erase(str.find_last_not_of(t) + 1);
        //Xóa ký tự trắng ở giữa nếu có hai ký tự trắng liền nhau.
        str.erase(unique(begin(str), end(str),
                        [](unsigned char a, unsigned char b){
                                return isspace(a) && isspace(b);
                        }), end(str));
        return str;
}
int main() {
        string fullName="   Tran    Viet    Khoa   ";
        cout<<traiteFullName(fullName);
        return 0;
}
```

- Output:
  Tran Viet Khoa

Bài toán 4. Tách từ (tokenize)

1. Viết hàm tách các từ trong một chuỗi, ký tự phân biệt từ có thể là ký tự trắng hoặc ký tự đặc biệt.
2. Ví dụ: st="C*C++*Java-> C, C++, Java với ký tự phân tách (dilimiter) là '*'.
3. Trong thư viện <string.h> của ngôn ngữ C có hàm strtok(), strtnok() cho công việc này, riêng đối với lớp string của STL C++ không có, tuy nhiên chúng ta dễ dàng cài đặt hàm này bằng việc sử dụng các hàm find(), find_first_not_of().

Tách từ trong xâu

```
#include <bits/stdc++.h>
using namespace std;
vector<string> tokenize(string const &str, const char delim){
        size_t start, end = 0;  vector<string> out;
        while ((start = str.find_first_not_of(delim, end)) != string::npos){
                end = str.find(delim, start);
                out.push_back(str.substr(start, end - start));
        }
        return out;
}
int main(){
        string s = "C*C++*Java"; const char delim = '*';
        vector<std::string> out;        out=tokenize(s, delim);
        for (auto &s: out)
                cout << s << endl;
        return 0;
}
```

- Output:
  C
  C++
  Java