

XÂU KÍ TỰ (STRING)



Khái quát về string:



String là lớp để **xử lý xâu kí tự** trong ngôn ngữ lập trình C++. Các bạn có thể nghĩ **String** như **một mảng kí tự** nhưng có thể **mở rộng, thu hẹp** và **hỗ trợ rất nhiều hàm xử lý xâu** thông dụng.

Cú pháp khai báo:

```
string name_string;
```



1. Nhập xuất string:

Khi nhập xuất string bạn cần chú ý:
Nếu chuỗi ký tự bạn nhập **không có dấu cách** thì bạn có thể **sử dụng cin** để nhập, trường hợp **có dấu cách** thì bạn cần dùng hàm **getline**.

Nhập string không có dấu cách:

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

int main(){
    string s, t;
    cin >> s >> t; // nhập s, t từ bàn phím
    cout << t << ' ' << s << endl;
}
```

INPUT

28tech dev

OUTPUT

dev 28tech

1. Nhập xuất string:

Trong trường hợp xâu bạn nhập **có dấu cách**, nếu sử dụng **cin**, bạn **chỉ nhập được từ đầu tiên của xâu đó**, vì bản chất của **cin** là nó sẽ dừng nhập khi gặp khoảng trắng.

Nhập string có dấu cách:

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

```
int main(){
    string s;
    getline(cin, s);
    cout << s << endl;
}
```

INPUT

28tech dev

OUTPUT

28tech dev

1. Nhập xuất string:

Khi dùng `getline`, bản chất cách hoạt động của `getline` sẽ dừng nhập tới khi gặp dấu xuống dòng, vì thế hãy đảm bảo trước khi nhập `getline`, trong bộ nhớ đệm bàn phím không còn thừa dấu `enter` do `cin` để lại từ câu lệnh nhập trước.

Tình huống xảy ra trôi lệnh:

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

int main(){
    int x; cin >> x; // cin sẽ để lại enter
    //trong bộ đệm bàn phím
    string s;
    getline(cin, s); // getline đọc phải
    //phím enter và kết thúc việc nhập
    cout << s << endl;
}
```

INPUT

28
Tech

OUTPUT

1. Nhập xuất string:

Cách xử lý: Hãy nhớ rằng không phải cứ trước getline là bạn cần xóa bộ đệm, bao giờ trước getline mà có câu lệnh cin thì mới cần phải xóa bộ đệm. Các bạn xóa đi phím enter trong bộ đệm bằng câu lệnh `cin.ignore(1)`

Xử lý tình huống bị trôi lệnh:

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

int main(){
    int x; cin >> x; // cin sẽ để lại enter
    trong bộ đệm bàn phím
    string s;
    cin.ignore(1); // Xóa 1 kí tự khỏi bộ đệm
    bàn phím chính là phím enter của cin để lại
    getline(cin, s); // getline đọc phải phím
    enter và kết thúc việc nhập
    cout << s << endl;
}
```

INPUT

28
Tech dev

OUTPUT

Tech dev



2. Các hàm xử lý thông dụng:

Hàm size và length: Cả 2 hàm này đều dùng để xác định chiều dài chuỗi, hay nói cách khác là số những kí tự xuất hiện trong chuỗi.

Ví dụ:

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

int main(){
    string s = "28tech C++ Python";
    cout << s.size() << endl;
    cout << s.length() << endl;
}
```

OUTPUT

17
17

2. Các hàm xử lý thông dụng:

Duyệt chuỗi: Bạn có thể dùng `cout` để in ra toàn bộ kí tự trong chuỗi hoặc sử dụng chỉ số tương tự như mảng để truy cập vào từng kí tự xuất hiện trong chuỗi.

Ví dụ:

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

int main(){
    string s = "28tech-C-Python";
    for(int i = 0; i < s.size(); i++){
        cout << s[i] << ' ';
    }
    cout << endl;
    //for each
    for(char x : s){
        cout << x << ' ';
    }
}
```

OUTPUT

```
28tech-C-Python
28tech-C-Python
```


2. Các hàm xử lý thông dụng:

Hàm `push_back` và `pop_back`: Tương tự như vector thì string cũng hỗ trợ thêm 1 phần tử vào cuối hay xóa 1 phần tử ở cuối xâu. Ở đây các phần tử của string chính là kiểu `char` vì thế các bạn chỉ có thể `push_back` từng kí tự một.

Ví dụ:

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

int main(){
    string s = "28tech";
    s.push_back('@'); // s.push_back("@")
    //hoặc s.push_back("@123") là sai
    cout << s << endl;
    s.pop_back();
    cout << s << endl;
}
```

OUTPUT

```
28tech@
28tech
```

2. Các hàm xử lý thông dụng:

Hàm insert: Bạn có thể chèn 1 chuỗi ký tự vào vị trí bất kì trong chuỗi ban đầu. Bạn chỉ cần cung cấp **chỉ số cần chèn** và **chuỗi cần chèn** cho hàm này.

Ví dụ:

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

int main(){
    string s = "28tech";
    s.insert(2, "@@@@"); // chèn chuỗi vào
    //chỉ số 2
    cout << s << endl;
}
```

OUTPUT

28@@@@tech

2. Các hàm xử lý thông dụng:

Hàm erase: Bạn có thể chỉ rõ chỉ số bắt đầu xóa kí tự và số lượng kí tự muốn xóa.

Ví dụ:

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

int main(){
    string s = "28@@@tech";
    s.erase(2, 3); //Xóa 3 kí tự từ chỉ số 2
    cout << s << endl;
    string t = "28@@@tech";
    t.erase(2); //Xóa mọi kí tự từ chỉ số 2
    cout << t << endl;
}
```

OUTPUT

```
28tech
28
```

2. Các hàm xử lý thông dụng:

Hàm find: Giả sử bạn cần kiểm tra sự tồn tại của chuỗi con t trong chuỗi s. Hàm find trả về chỉ số đầu tiên của chuỗi t trong chuỗi s nếu s có chứa t, ngược lại hàm này trả về giá trị `string::npos`.

Ví dụ:

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

int main(){
    string s = "28tech tech";
    string t = "tech";
    if(s.find(t) != string::npos){
        cout << "FOUND\n";
    }
    else{
        cout << "NOT FOUND\n";
    }
}
```

OUTPUT

FOUND

3. So sánh và cộng xâu:



String đã được **nạp chồng** các toán tử so sánh, vì thế các bạn có thể **so sánh 2 xâu** theo thứ tự từ điển với **các toán tử so sánh**, ngoài ra các bạn còn có thể **sử dụng toán tử + để nối 2 xâu**.

So sánh hai xâu:

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

```
int main(){
    string s = "28tech";
    string t = "28tech";
    cout << (s == t) << endl;
    cout << (s < t) << endl;
    cout << (s <= t) << endl;
    cout << (s != t) << endl;
}
```

OUTPUT

```
1
0
1
0
```

Nối hai xâu:

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

```
int main(){
    string s = "28tech";
    string t = "dev";
    string st = s + " " + t;
    string ts = t + " " + s;
    cout << st << endl;
    cout << ts << endl;
}
```

OUTPUT

```
28tech dev
dev 28tech
```



4. Nhắc lại các hàm kiểm tra kí tự:



Ở bài if else các bạn đã học cách tự viết các câu lệnh if để kiểm tra loại kí tự, bây giờ các bạn có thể sử dụng các hàm có sẵn trong thư viện `<ctype.h>`.

Hàm	Chức năng
<code>isdigit(char c)</code>	Kiểm tra chữ số
<code>islower(char c)</code>	Kiểm tra chữ in thường
<code>isupper(char c)</code>	Kiểm tra in hoa
<code>isalpha(char c)</code>	Kiểm tra chữ cái
<code>int tolower(char c)</code>	Chuyển thành chữ in thường
<code>int toupper(char c)</code>	Chuyển thành chữ in hoa



4. Nhắc lại các hàm kiểm tra kí tự:

Hàm chuyển xâu thành in thường:

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

void inthuong(string &s){
    for(int i = 0; i < s.size(); i++){
        s[i] = tolower(s[i]);
    }
}

int main(){
    string s = "abcd XYZ 123@";
    inthuong(s);
    cout << s << endl;
}
```

OUTPUT

abcd xyz 123@

4. Nhắc lại các hàm kiểm tra kí tự:

Hàm chuyển chuỗi thành in hoa:

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

void inhoa(string &s){
    for(int i = 0; i < s.size(); i++){
        s[i] = toupper(s[i]);
    }
}

int main(){
    string s = "abcd XYZ 123@";
    inhoa(s);
    cout << s << endl;
}
```

OUTPUT

ABCD XYZ 123@



4. Nhắc lại các hàm kiểm tra kí tự:

Phân loại kí tự:

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

int main(){
    string s = "abcd XYZ 123@";
    int alpha = 0, digit = 0, special = 0;
    for(char x : s){
        if(isdigit(x)) ++digit;
        else if(isalpha(x)) ++alpha;
        else special++;
    }
    cout << "Chu cai: " << alpha << endl;
    cout << "Chu so: " << digit << endl;
    cout << "Ki tu dac biet: " << special << endl;
}
```

OUTPUT

Chu cai: 7
Chu so: 3
Ki tu dac biet: 3

5. Chuyển chuỗi thành số và ngược lại:



Để chuyển chuỗi gồm các chữ số thành số, ta dùng **hàm stoi** để chuyển một chuỗi thành **số int** và **hàm stoll** để chuyển một chuỗi thành **số long long**.

Chuyển chuỗi thành số:

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

int main(){
    string s = "123422";
    int n = stoi(s);
    cout << n << endl;
    string t = "0001111129293293";
    long long m = stoll(t);
    cout << m << endl;
}
```

OUTPUT

```
123422
1111129293293
```



5. Chuyển chuỗi thành số và ngược lại:



Để chuyển một số thành chuỗi ta dùng hàm `to_string` (C++11 trở lên).

Chuyển số thành chuỗi:

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

int main(){
    int a = 12345;
    string s = to_string(a);
    cout << s << endl;
    s += "678";
    cout << s << endl;
}
```

OUTPUT

```
12345
12345678
```



6. Bài toán số lớn:



Khi gặp bài toán mà số lượng chữ số của số đầu bài cho lên tới hàng nghìn chữ số thì các bạn không thể dùng int hay long long để lưu. Trong trường hợp này các bạn lưu như một xâu kí tự.

Tính tổng chữ số của một số có nhiều chữ số:

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

int main(){
    string s = "17823717237128231723712378123123782371273123";
    int sum = 0;
    for(char x : s){
        sum += x - '0';
    }
    cout << sum << endl;
}
```

OUTPUT

163



7. Các bài toán liên quan tới tần suất xuất hiện của kí tự trong xâu:



Để đếm tần suất xuất hiện của các kí tự xuất hiện trong xâu các bạn có thể **sử dụng map hoặc mảng để đếm**, vì các kí tự thường gặp đều có mã ASCII từ 0 tới 255 nên sử dụng **mảng đếm có 256 phần tử** là có thể đếm được kí tự xuất hiện.



Các bạn sử dụng 2 cách trên để đếm tần xuất và in ra theo thứ tự xuất hiện trong xâu, mỗi kí tự xuất hiện 1 lần.



7. Các bài toán liên quan tới tần suất xuất hiện của kí tự trong xâu:

Cách 1: Sử dụng mảng đếm

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

int main(){
    string s = "abcdabcdzzzza";
    int cnt[256] = {0};
    for(char x : s){
        cnt[x]++;
    }
    for(int i = 0; i < 256; i++){
        if(cnt[i]){
            cout << (char)i << ' ' << cnt[i] << endl;
        }
    }
}
```

OUTPUT

```
a 3
b 2
c 2
d 2
z 4
```

7. Các bài toán liên quan tới tần suất xuất hiện của kí tự trong xâu:

Cách 2: Sử dụng map

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

```
int main(){
    string s = "abcdabcdzzzza";
    map<char, int> mp;
    for(char x : s)
        mp[x]++;
    for(auto it : mp){
        cout << it.first << ' ' << it.second << endl;
    }
}
```

OUTPUT

a 3

b 2

c 2

d 2

z 4

8. String với mảng, vector, set, map:

Sắp xếp các tử trong mảng hoặc vector theo thứ tự từ điển tăng dần, giảm dần

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

int main(){
    string a[] = {"28tech", "dev", "dsa", "c++", "python"};
    sort(a, a + 5);
    for(string x : a){
        cout << x << ' ';
    }
    cout << endl;
    vector<string> v = {"28tech", "dev", "dsa", "c++", "python"};
    sort(begin(v), end(v), greater<string>());
    for(string x : v){
        cout << x << ' ';
    }
}
```

OUTPUT

28tech c++ dev dsa python
python dsa dev c++ 28tech



8. String với mảng, vector, set, map:

Lọc ra số lượng từ khác nhau trong mảng

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

int main(){
    string a[] = {"28tech", "dev", "dsa", "c++", "python",
"28tech", "dev", "Python"};
    set<string> se;
    for(string x : a){
        se.insert(x);
    }
    cout << se.size() << endl;
    for(string x : se){
        cout << x << ' ';
    }
}
```

OUTPUT

```
6
28tech Python c++ dev dsa python
```

8. String với mảng, vector, set, map:

Đếm số lần xuất hiện các từ trong mảng

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

int main(){
    string a[] = {"28tech", "dev", "dsa", "c++", "python",
    "c++", "dev", "Python"};
    map<string, int> mp;
    for(string x : a){
        mp[x]++;
    }
    for(auto it : mp){
        cout << it.first << ' ' << it.second << endl;
    }
}
```

OUTPUT

```
28tech 1
Python 1
c++ 2
dev 2
dsa 1
python 1
```



Qua các ví dụ trên, bạn hoàn toàn có thể khai báo mảng và vector string để có thể sort, cũng như đưa string vào set map...

