Mục lục

1 Giới thiệu cơ bản về kí tự và xâu kí tự trong C++

1.1 Kiểu dữ liệu kí tự - char trong C++

1.1.1 Lý thuyết

- Kiểu dữ liệu ký tự, xét về mặt bản chất, nó lưu một kí tự trong bảng **ASCII** và sử dụng nó để xử lí trong chương trình.

- Như bao kiểu dữ liệu khác, cách khai báo kiểu dữ liệu char trong C++ như sau:

```
char x;
char a = 'c';
```

Với như 2 dòng trên là khai báo 2 biến x và a với kiểu dữ liệu char. Trong đó, x chưa gán sẵn giá trị và a mang giá trị là kí tự c.

1.1.2 Các ứng dụng

- **Bảng ASCII trong C++**

- Với bảng ASCII, ta hoàn toàn có thể chuyển đổi từ :

+) kí tự chữ in thường sang số thứ tự của kí tự trong bảng Alphabet. Gọi giá trị đó là v, ta có:

$\Rightarrow v = x -' a'$

VD : x = 'a' $\Rightarrow v = 0$

x = 's' $\Rightarrow v = 18$

+) kí tự chữ in hoa sang số thứ tự của kí tự trong bảng Alphabet:

$\Rightarrow v = x -' A'$

VD : x = 'A' $\Rightarrow v = 0$

x = 'S' $\Rightarrow v = 18$

+) kí tự số sang giá trị số:

$\Rightarrow v = x -' 0'$

VD : x = '0' $\Rightarrow v = 0$

x = '7' $\Rightarrow v = 7$

+) Kiểm tra kí tự có phải là chữ in thường hay không

$\Rightarrow' a' \leq x \leq 'z'$

+) Kiểm tra kí tự có phải là chữ in hoa hay không

$\Rightarrow' A' \leq x \leq 'Z'$

+) Kiểm tra kí tự có phải là số hay không

$\Rightarrow' 0' \leq x \leq '9'$

- Về cơ bản việc trừ đi char cho nhau là ta đang lấy các số thứ tự trong bảng ASCII trừ cho nhau. Do các ký tự in hoa, in thường và các chữ số đứng cạnh nhau trong bảng ASCII nên ta có thể làm các điều như trên.

1.2 Kiểu dữ liệu xâu kí tự - String trong C++

1.2.1 Lý thuyết

- String hay còn gọi là xâu kí tự là một chuỗi các kí tự được lưu như 1 mảng trong C++.

- Cũng như các kiểu dữ liệu khác, string được khai báo như sau:

```
string s;
string a = "abc";
```

- Với việc khai báo như trên, ta khai báo string s rỗng và string a với giá trị là "abc".

- Việc sử dụng string, cũng tương tự như ta sử dụng mảng char, gọi s là string hiện tại, ta có:

+) s.resize(n) được dùng để thay đổi kích thước của mảng thành n.

+) $s[i]$ được dùng để truy cập vào phần tử thứ i của s.

+) s.size() được dùng để lấy kích thước của xâu hiện tại tính từ phần tử số 0.

Lưu ý: cũng như mảng, string đánh dấu các phần tử từ số 0. Nên nếu s.size() = n thì các phần tử được đánh dấu từ $0 \rightarrow n-1$.

- Tiếp đến, chúng ta có 2 cách đọc xâu với 2 công dụng khác nhau:

+) Để đọc 1 chuỗi xâu cho tới khi gặp 1 dấu cách hoặc 1 dấu xuống dòng, ta đọc như sau:

```
cin >> s;
```

Ví dụ input: abcxyz abcd

Nếu đọc dùng lệnh trên, xâu s của ta có giá trị = "abcxyz".

+) Để đọc hết tất cả kí tự trên 1 dòng ta đọc như sau:

```
getline(cin , s);
```

Ví dụ input: abcxyz abcd

Nếu đọc dùng lệnh trên, xâu s của ta có giá trị = "abcxyz abcd". Tức ta đọc hết 1 dòng kể cả các dấu cách.

2 Bài tập

2.1 Bài A - Nhập xâu 1

2.1.1 Hướng dẫn

- Ta chỉ cần dùng lệnh cin để đọc xâu với bài tập này, vì là xâu liên tiếp.

2.1.2 Code mẫu

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

int main() {
    ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
    cin >> a;
    cout << a;
}
```

2.2 Bài B - Đảo ngược xâu

2.2.1 Hướng dẫn

- Ta in ra từng kí tự của xâu s theo thứ tự ngược lại với thứ tự ban đầu.

2.2.2 Code mẫu

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

string s;

int main(){
    ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
    cin >> s;
    for(int i = s.size() - 1 ; i >= 0 ; i--)
        cout << s[i];
    return 0;
}
```

2.3 Bài C - Đếm số lần xuất hiện của ký tự

2.3.1 Hướng dẫn

- Ta duyệt qua tất cả n phần tử trong xâu s ban đầu và kiểm tra $s[i]$ có bằng x hay không.

2.3.2 Code mẫu

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

int n;
char x;
string s;

int main(){
    ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
    cin >> n >> x;
    cin >> s;

    int cnt = 0;
    for(int i = 0 ; i < s.size() ; i++){
        if(s[i] == x)
            cnt++;
    }

    cout << cnt << endl;
    return 0;
}
```

Đọc code đầy đủ hơn ở *đây*

2.4 Bài D. Kiểm tra đối xứng

2.4.1 Hướng dẫn

- Đầu tiên, ta có vị trí đối xứng với vị trí i trong xâu độ dài n và các phần tử đánh dấu từ $0 \rightarrow n-1$ là $n-i-1$.

- Vậy ta thấy, kiểm tra xâu s có phải xâu đối xứng hay không, ta kiểm tra $s[i] = s[n-i-1] \ \forall \ i < n$ với n là độ dài xâu được đánh dấu từ $0 \rightarrow n-1$.

2.4.2 Code mẫu

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

int n;
string s;

int main(){
    ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
    cin >> n;
    cin >> s;

    bool ok = true;
    for(int i = 0 ; i < s.size() ; i++)
        if(s[i] != s[n - i - 1]){
            ok = false;
            break;
        }

    if(ok == true)
        cout << "YES" << endl;
    else cout << "NO" << endl;
    return 0;
}
```

2.5 Bài E - Ký tự đầu tiên và cuối cùng

2.5.1 Hướng dẫn

- Bài toán của ta có 2 vấn đề cần giải quyết như sau:

+) Kí tự đầu tiên của s là gì?

+) Kí tự cuối cùng của xâu t là gì?

- Câu trả lời cho 2 câu hỏi trên như sau:

+) Kí tự đầu của xâu s như ta biết chính là $s[0]$.

+) Kí tự cuối cùng của xâu t, ta có thể lấy nó bằng $t[t.size() - 1]$ nghĩa là ta lấy phần tử cuối cùng của xâu t.

2.5.2 Code mẫu

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

string s , t;
```

```
int main(){
    ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
    cin >> s;
    cin >> t;

    cout << s[0] << " " << t[t.size() - 1] << " ";
    if(s[0] == t[t.size() - 1])
        cout << 1;
    else cout << 0;
    return 0;
}
```

Đọc code đầy đủ hơn ở *đây*

2.6 Bài F - So sánh tên hai người

2.6.1 Hướng dẫn

- Ta ứng dụng hàm getline() và sử dụng s.size() và t.size() để so sánh độ dài 2 xâu.

2.6.2 Code mẫu

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

string s , t;

int main(){
    ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
    getline(cin , s);
    getline(cin , t);

    if(s.size() > t.size())
        cout << s;
    else cout << t;
    return 0;
}
```

2.7 Bài G - Xóa ký tự trắng trong xâu

2.7.1 Hướng dẫn

- Đọc vào bằng lệnh getline(), sau đấy ta sẽ in ra tất cả các kí tự trong xâu s ngoại trừ các kí tự là dấu cách. Ta có thể làm điều trên bằng cách so sánh $s[i] != ' '$.

2.7.2 Code mẫu

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

string s;

int main(){
    ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
    getline(cin , s);

    for(int i = 0 ; i < s.size() ; i++)
        if(s[i] != ' ')
            cout << s[i];
    return 0;
}
```

Đọc code đầy đủ hơn ở *đây*

2.8 Bài H - Xóa chữ số trong xâu

2.8.1 Hướng dẫn

- Ta sẽ in ra hết tất cả các phần tử $s[i]$ không phải là chứ số bằng cách kiểm tra điều kiện

$$'0' \leq s[i] \leq '9'$$

2.8.2 Code mẫu

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

string s;

int main(){
    ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
    getline(cin , s);

    for(int i = 0 ; i < s.size() ; i++){
        if('0' <= s[i] && s[i] <= '9')
            continue;
        cout << s[i];
    }
    return 0;
}
```

2.9 Bài I - Đếm số lần xuất hiện của ký tự 2

2.9.1 Hướng dẫn

- Ta sẽ chạy 26 lần duyệt qua các kí tự từ $a \rightarrow z$ với mỗi lần duyệt như thế, ta lại duyệt qua n phần tử và đếm có bao nhiêu phần tử bằng với kí tự hiện tại.

2.9.2 Code mẫu

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

string s;

int main(){
    ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
    cin >> s;

    for(int x = 'a' ; x <= 'z' ; x++){
        int cnt = 0;
        for(int i = 0 ; i < s.size() ; i++){
            if(s[i] == x)
                cnt++;
        }
        cout << cnt << " ";
    }

    return 0;
}
```

Đọc code đầy đủ hơn ở *đây*

2.10 Bài J - Happy New Year!!!

2.10.1 Hướng dẫn

- Với bài tập này ta có thể thực hiện phép so sánh xâu s với xâu "Happy New Year!!!".

Lưu ý: ta phải sử dụng lệnh cin.ignore() ngay sau khi đọc số lượng test case để có thể xuống dòng.

2.10.2 Code mẫu

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

string s;
```

```cpp
int main(){
    ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);

    int test;
    cin >> test;
    cin.ignore();

    for(int i = 1 ; i <= test ; i++){
        getline(cin , s);
        if(s == "Happy New Year!!!"){
            cout << "YES" << endl;
        }else cout << "NO" << endl;
    }

    return 0;
}
```

Đọc code đầy đủ hơn ở *đây*

2.11 Bài K - Số từ trong xâu

2.11.1 Hướng dẫn

- Với bài này, đáp án của ta là số lượng dấu cách cộng 1. VÌ giữa 2 từ ngăn cách nhau bằng 1 dấu cách nên sẽ có số dấu cách + 1 từ.

2.11.2 Code mẫu

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

string s;

int main(){
    ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);

    getline(cin , s);
    int cnt = 0;
    for(int i = 0 ; i < s.size() ; i++)
        if(s[i] == ' ')
            cnt++;
    cout << cnt + 1 << endl;
    return 0;
}
```

Đọc code đầy đủ hơn ở *đây*

2.12 Bài L - Số lượng ký tự

2.12.1 Hướng dẫn

- Với bài này, ta cần thực hiện kiểm tra với mỗi ký tự thứ i là:
  +) s[i] có phải là chữ in hoa hay không.
  +) s[i] có phải là chữ in thường hay không.
  +) s[i] có phải là chữ số hay không.

- Đọc lại phần lí thuyết, ta hoàn toàn có thể kiểm tra các điều trên với 3 lệnh if.

2.12.2 Code mẫu

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

string s;

int main(){
    ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);

    getline(cin , s);
    int cnt1 = 0 , cnt2 = 0 , cnt3 = 0;
    for(int i = 0 ; i < s.size() ; i++){
        if('0' <= s[i] && s[i] <= '9')
            cnt1++;
        if('a' <= s[i] && s[i] <= 'z')
            cnt2++;
        if('A' <= s[i] && s[i] <= 'Z')
            cnt3++;
    }

    cout << cnt3 << endl << cnt2 << endl << cnt1 << endl;
    return 0;
}
```

2.13 Bài M - In hoa in thường

2.13.1 Hướng dẫn

- Với bài toán này, ta có thể phân biệt chữ đầu tiên của 1 từ và các chữ cái còn lại như sau:

+) Nếu là kí tự tại vị trí 0

⇒ Ký tự đầu

+) Nếu là kí tự tại vị trí i và có s[i - 1] là dấu cách

⇒ Ký tự đầu

+) Các trường hợp còn lại sẽ là các chữ khác chữ cái đầu của 1 chữ.

- Tiếp theo, ta cần làm:

+) Nếu kí tự i là kí tự đầu chữ thì xác định xem là in hoa hay in thường, nếu là in thường đổi thành in hoa.

+) Nếu kí tự i là kí tự không phải đầu chữ thì xác định xem là in hoa hay i thường, nếu là in hoa thì đổi thành in thường.

- Đối với việc kiểm tra in hoa in thường, ta đã biết. Còn chuyển đổi từ in thường thành in hoa và ngược lại, ta làm thế nào?

- Ta có cách làm như sau:

+) x là kí tự in thường và đổi thành in hoa, ta trừ x đi khoảng cách trong bằng ASCII giữa 'a' và 'A'.

⇒ $x -= 'a' - 'A'$;

+) x là kí tự in hoa và đổi thành in thường, ta cộng x thêm khoảng cách trong bằng ASCII giữa 'a' và 'A'.

⇒ $x -= 'a' - 'A'$;

2.13.2 Code mẫu

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

int main()
{
    string s;
    getline(cin , s);

    for(int i = 0 ; i < s.size() ; i++){
        if(i == 0 || (i > 0 && s[i - 1] == ' ')){
            if('a' <= s[i] && s[i] <= 'z'){
                s[i] -= ('a' - 'A');
            }
```

```
        }else{
            if('A' <= s[i] && s[i] <= 'Z'){
                s[i] += ('a' - 'A');
            }
        }
    }
    cout << s << endl;
}
```

Đọc code đầy đủ hơn ở *đây*

2.14 Bài N - So sánh 2 xâu

2.14.1 Hướng dẫn

- Với bài toán này, ta chỉ quan tâm tới từng vị trí i riêng biệt và kiểm tra có vị trí nào không thỏa mãn điệu kiện hay không.

- Để có thể biến đổi s[i] = t[i] thì 2 kí tự s[i] và t[i] phải thỏa mãn điều kiện sau:

+) cả 2 là dấu ?. Vì ta có thể điền bất kì kí tự nào vào 2 chỗ dấu hỏi chấm.

+) 1 ký tự là dấu hỏi và ký tự còn lại là chữ cái. Vì ta có thể điền chữ cái đã được cố định vào dấu hỏi chấm còn lại.

+) là 2 ký tự giống nhau. Vì giống nhau nên thỏa mãn.

- Vậy ta còn sót lại 1 trường hợp, 2 ký tự khác nhau. Vì thế, với mỗi vị trí i, ta chỉ cần kiểm tra s[i] khác t[i] hay không, nếu có ta kết luận k thể biến đổi 2 xâu thành bằng nhau.

2.14.2 Code mẫu

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

int main()
{
    int numTest;
    cin >> numTest;

    for(int i = 1 ; i <= numTest ; i++){
        string s , t;
        cin >> s >> t;

        bool ok = true;
        for(int i = 0 ; i < s.size() ; i++){
            if(s[i] != t[i] && s[i] != '?' && t[i] != '?'){
                ok = false;
                break;
```

```
            }
        }

        if(ok)
            cout << "1" << endl;
        else cout << "0" << endl;
    }
}
```

Đọc code đầy đủ hơn ở *đây*

2.15 Bài O - Số giả nguyên tố 4

2.15.1 Hướng dẫn

- Với bài toán này, ta chia thành 2 điều kiện như sau:

+) Là một số thập phân hữu hạn dương mà từng chữ số của nó hoặc từng chữ số ở phần thập phân của nó là số nguyên tố.

+) Tổng các chữ số của nó là số nguyên tố

Vậy với 2 điều kiện trên, ta xét như sau:

+) Từng chữ số ở phần thập phân của nó là số nguyên tố.

+) Tổng các chữ số của nó là số nguyên tố.

- Vậy các bước thực hiện của ta là như sau:

+) Tìm vị trí của dấu phẩy.

+) Có được vị trí dấu phẩy, ta tính tổng các số và xác định xem có phải tổng các chữ số là số nguyên tố hay không.

+) Tiếp theo, ta có thể xác định các số sau dấu phẩy có phải tất là số nguyên tố hay không.

- Nếu 1 trong 2 điều kiện trên không thỏa mãn thì in ra "KHONG" còn lại in ra "CO".

2.15.2 Code mẫu

```
    #include <bits/stdc++.h>

using namespace std;

bool isPrime(int x){
    if(x < 2) return false;
    for(int i = 2 ; i * i <= x ; i++){
        if(x % i == 0)
            return false;
    }
```

```
    return true;
}

int main(){
    string s;
    cin >> s;

    int viTri = -1;

    for(int i = 0 ; i < s.size() ; i++){
        if(s[i] == ','){
            viTri = i;
            break;
        }
    }

    int Tong = 0;
    for(int i = 0 ; i < s.size() ; i++)
        if(i != viTri) Tong += (s[i] - '0');

    bool allNguyenTo = true;
    for(int i = viTri + 1 ; i < s.size() ; i++){
        if(isPrime(s[i] - '0') == false){
            allNguyenTo = false;
            break;
        }
    }

    if(allNguyenTo == true && isPrime(Tong) == true){
        cout << "CO" << endl;
    }else{
        cout << "KHONG" << endl;
    }
}
```

Đọc code đầy đủ hơn ở *đây*

2.16 Bài P - Tạo số lớn nhất

2.16.1 Hướng dẫn

- Với bài toán này, để tạo nên số lớn nhất, ta nhận thấy việc ta chèn số x và ngay trước số đầu tiên bé hơn nó là tối ưu.

- Vậy việc ta cần làm là duyệt tìm số đầu tiên bé hơn x và thêm nó vào ngay trước số đó.

2.16.2 Code mẫu

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

int main()
{
    string s;
    char x;
    cin >> s >> x;
    int pos = -1;
    for(int i = 0 ; i < s.size() ; i++)
    {
        if(s[i] < x)
        {
            pos = i;
            break;
        }
    }
    if(pos == -1)
    {
        for(int i = 0 ; i < s.size() ; i++)
            cout << s[i];
        cout << x;
    }
    else
    {
        for(int i = 0 ; i < pos ; i++)
            cout << s[i];
        cout << x;
        for(int i = pos ; i < s.size() ; i++)
            cout << s[i];
    }
}
```

Đọc code đầy đủ hơn ở *đây*

2.17 Bài Q - Ký tự xuất hiện nhiều nhất

2.17.1 Hướng dẫn

- Với bài toán này, ta duyệt từng kí tự từ $a \rightarrow z$, đếm số lần xuất hiện trong xâu và tìm chữ cái nào có số lần xuất hiện tối đa và có thứ tự từ điển nhỏ nhất.

2.17.2 Code mẫu

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

string s;

int main(){
    ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
    cin >> s;

    char res;
    int mx = 0;

    for(int x = 'a' ; x <= 'z' ; x++){
        int cnt = 0;
        for(int i = 0 ; i < s.size() ; i++){
            if(s[i] == x)
                cnt++;
        }

        if(cnt > mx){
            mx = cnt;
            res = x;
        }
    }

    cout << res << " " << mx << endl;

    return 0;
}
```

2.18 Bài R - Xâu con liên tiếp dài nhất có cùng ký tự

2.18.1 Hướng dẫn

- Với bài toán này, ta sẽ so sánh mỗi s[i] với s[i - 1] và duy trì 1 biến đếm. Biến đếm ấy thay đổi với mỗi i như sau:

+) Nếu s[i] khác s[i - 1], độ dài đoạn con giống nhau hiện tại = 1.

+) Nếu s[i] = s[i - 1], đọ dài đoạn con giống nhau hiện tại tăng thêm 1. Ta sẽ cập nhật max dựa trên điều này.

2.18.2 Code mẫu

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

```
string s;

int main(){
    ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
    cin >> s;
    int res = 1 , cur = 1;
    for(int i = 1 ; i < s.size() ; i++){
        if(s[i] == s[i - 1])
            cur++;
        else cur = 1;
        res = max(res , cur);
    }
    cout << res << endl;
    return 0;
}
```

Đọc code đầy đủ hơn ở *đây*

2.19 Bài S - Xóa ký tự

2.19.1 Hướng dẫn

- Với bài toán này, ta có nhận xét như sau. Nếu có K thao tác và mỗi thao tác, ta có thể xóa 1 kí tự ở đầu hoặc ở cuối. Việc này đồng nghĩa với việc thực hiện hết K thao tác, ta sẽ tạo ra được 1 đoạn con liên tục có độ dài là N - K bất kì.

- Vì vậy, ta cần kiểm tra trong tất cả các đoạn con có độ dài N - K, có đoạn con nào là xâu đối xứng hay không?

2.19.2 Code mẫu

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

int n , k;
string s;

int main(){
    ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
    cin >> n >> k;
    cin >> s;

    for(int i = 0 ; i + (n - k) - 1 < s.size() ; i++){

        int l = i , r = i + (n - k) - 1;
        bool ok = true;
        while(l < r){
            if(s[l] != s[r]){
                ok = false;
```

```
                break;
            }
            l++;
            r--;
        }

        if(ok){
            for(int j = i ; j <= i + (n - k - 1) ; j++)
                cout << s[j];
            return 0;
        }
    }

    cout << "No" << endl;

    return 0;
}
```

Đọc code đầy đủ hơn ở *đây*

2.20 Bài T - Số xâu con liên tiếp có ký tự đầu bằng cuối

2.20.1 Hướng dẫn

Với bài toán này, vì ta chỉ quan tâm với kí tự đầu và cuối có bằng nhau hay không, ta thu hẹp về bài toán đến số cặp ký tự bằng nhau và ta sẽ chạy duyệt qua 26 lần với 26 ký tự từ $a \rightarrow z$ và duy trì 1 biến đếm số lượng số bằng ký tự đang xét hiện tại với mỗi lần duyệt qua n phần tử.

Với mỗi lần duyệt qua n phần tử, ta đếm như sau. Mỗi lần gặp s[i] bằng kí tự đang xét hiện tại, ta cộng vào kết quả cuối cùng số lượng bằng với số kí tự bằng s[i] ở trước nó + 1.

2.20.2 Code mẫu

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

string s;

int main(){
    ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
    cin >> s;

    long long res = 0;

    for(int x = 'a' ; x <= 'z' ; x++){
```

```
        int cnt = 0;
        for(int i = 0 ; i < s.size() ; i++){
            if(s[i] == x){
                cnt++;
                res += cnt;
            }
        }
    }

    cout << res << endl;
    return 0;
}
```

Đọc code đầy đủ hơn ở *đây*

2.21 Bài U - Kiểm tra in hoa, in thường

2.21.1 Hướng dẫn

- Đối với bài toán này, ta sẽ duyệt qua 26 lần các kí tự và mỗi lần kiểm tra trong xâu có kí tự đó in hoa và in thường hay không.

2.21.2 Code mẫu

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

string s;

int main()
{
    cin >> s;

    bool ok = false;

    for(int i = 'a' ; i <= 'z' ; i++){
        bool inThuong , inHoa;
        inThuong = inHoa = false;
        for(int j = 0 ; j < s.size() ; j++){
            if(s[j] == i)
                inThuong = true;
            if(s[j] == i - ('a' - 'A'))
                inHoa = true;
        }
        if(inThuong == true && inHoa == true)
            ok = true;
    }
```

```
    if(ok)
        cout << "YES";
    else cout << "NO";
}
```

Đọc code đầy đủ hơn ở *đây*

2.22 Bài V - So sánh 2 xâu theo thứ tự từ điển

2.22.1 Hướng dẫn

- Trong C++, có tồn tại các phép so sánh >,<,= cho kiểu dữ liệu string. Vì vật nếu ta muốn so sánh 2 xâu s và t theo thứ tự từ điển thì ta sẽ dùng các phép so sánh cũng tương tự như số nguyên.

2.22.2 Code mẫu

```
#include <bits/stdc++.h>
using namespace std;
string s,t;

int main()
{
    ios::sync_with_stdio(0);cin.tie(0);
    cin >> s >> t;
    if(s > t)
        cout << "YES";
    else cout << "NO";
    return 0;
}
```

2.23 Bài W - Sắp xếp xâu theo thứ tự điển

2.23.1 Hướng dẫn

- Vì trong C++ đã có sẵn phép so sánh cho string nên ta có thể sort 1 mảng string để xuất ra 1 mảng theo thứ tự từ điển.

2.23.2 Code mẫu

```
#include <bits/stdc++.h>
using namespace std;
int n;
string s[100005];

int main()
{
```

```
    ios::sync_with_stdio(0);
    cin.tie(0);
    cin >> n;
    for(int i = 1 ; i <= n ; i++)
        cin >> s[i];
    sort(s + 1 , s + 1 + n);
    for(int i = 1 ; i <= n ; i++)
        cout << s[i] << "\n";
}
```

Đọc code đầy đủ hơn ở *đây*

2.24 Bài X - Đếm xâu

2.24.1 Hướng dẫn

- Với bài toán này, ta sẽ sort mảng chứa n xâu lại và thực hiện duy trì 1 biến đếm như sau:

+) Nếu $s[i] == s[i-1]$, $cnt += 1$.

+) Nếu $s[i] != s[i-1]$, in ra cnt và s[i - 1] và gán cnt = 1.

- Và đến cuối cùng ta lại in ra 1 lần cnt nữa cùng xâu hiện tại.

2.24.2 Code mẫu

```
#include<bits/stdc++.h>
using namespace std;
const int MAXN=100005;

int n,cnt;
string s[MAXN];

int main(){
    ios::sync_with_stdio(0);
    cin.tie(0);
    cin>>n;
    for(int i=1; i<=n; i++) cin>>s[i];
    sort(s+1,s+n+1);
    cnt=1;
    for(int i=2; i<=n; i++){
        if(s[i]!=s[i - 1]){
            cout<<s[i - 1]<<' '<<cnt<<'\n';
            cnt=1;
        }
        else cnt++;
    }
    cout<<s[n]<<' '<<cnt<<'\n';
```

```
    return 0;
}
```

Đọc code đầy đủ hơn ở *đây*

2.25 Bài Y - Thứ tự từ điển bé nhất

2.25.1 Hướng dẫn

- Đối với bài toán này, ta nhận thấy về bản chất nó là sort lại tất cả các kí tự trong xâu theo thứ tự từ điển.

- Đối với kiểu dữ liệu string, ta có xâu s, C++ cũng hỗ trợ việc sort lại tất cả kí tự trong xâu s bằng lệnh sort(s.begin() , s.end()).

2.25.2 Code mẫu

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

string s;

int main(){
    cin >> s;
    sort(s.begin() , s.end());
    cout << s << endl;
    return 0;
}
```

2.26 Bài Z - Nhận diện khuông mặt

2.26.1 Hướng dẫn

- Về cơ bản, bài toán của ta đang yêu cầu kiểm tra với mỗi hình vuông 2x2 có tồn tại cả 4 chữ cái f , a , c , e hay không. Vậy ta phải giải quyết bài toán như thế nào?

- Việc kiểm tra 4 chữ cái trong hình vuông 2 x 2 có thể thực hiện bằng cách đưa cả 4 chữ cái vào 1 mảng có độ dài 4 và sort lại. Nếu mảng có giá trị đúng bằng a , c , e , f thì hình vuông đấy có đầy đủ cả 4 kí tự.

- Về cách đọc input, theo như ta thấy, với 1 bảng nxm cũng sẽ tương tự như việc ta có n xâu độ dài m. Nên ta sẽ đọc từ input n xâu lần lượt có độ dài m.

2.26.2 Code mẫu

```cpp
#include <bits/stdc++.h>

using namespace std;

int n , m;
string s[505];

int main(){
    cin >> n >> m;
    for(int i = 0 ; i < n ; i++)
        cin >> s[i];
    int soMat = 0;

    for(int i = 0 ; i < n - 1 ; i++){
        for(int j = 0 ; j < m - 1 ; j++){
            char ok[] = {s[i][j] , s[i + 1][j] , s[i][j + 1] , s[i + 1][j + 1]};
            sort(ok , ok + 4);

            if(ok[0] == 'a' && ok[1] == 'c' && ok[2] == 'e' && ok[3] == 'f')
                soMat++;
        }
    }

    cout << soMat << endl;
    return 0;
}
```

2.27 Bài ZA - Xâu con dài nhất

2.27.1 Hướng dẫn

- Tiếp cận với 1 tư tưởng tham lam, ta nhận thấy mỗi lần xóa đi 1 loại kí tự, ta nên chọn loại kí tự có số lần xuất hiện bé nhất trong xâu hiện tại. Vì thế, ta phải thiết kế thuật toán sao cho có thể tìm ra kí tự xuất hiện ít nhất.

- Đến đây, ta sẽ sử dụng kĩ thuật mảng đếm để đếm các phần tử tồn tại trong xâu. Gọi cnt là mảng đếm của ta, có cnt[i] quản lí số lần xuất hiện của kí tự i trong xâu s. Như vậy mỗi lần xóa kí tự, ta cần tìm cnt[i] bé nhất > 0 trong mảng cnt và gán lại cho giá trị cnt[i] bé nhất > 0 đấy giá trị = 0. Mang ý nghĩa là không còn kí tự nào bằng i. Và ta làm thế cho tới khi trong mảng cnt còn đúng k giá trị cnt[i] dương.

2.27.2 Code mẫu

```cpp
#include <bits/stdc++.h>

using namespace std;
```

```
int n , k;
string s;
int cnt[26];

int main(){
    cin >> n >> k;
    cin >> s;

    int riengBiet = 0;

    for(int i = 0 ; i < s.size() ; i++){
        cnt[s[i] - 'a']++;
    }

    for(int i = 0 ; i < 26 ; i++){
        if(cnt[i] > 0)
            riengBiet++;
    }

    int res = 0;

    while(riengBiet > k){
        int mx = -1;
        for(int i = 0 ; i < 26 ; i++){
                if(cnt[i] == 0) continue;

                if(mx == -1)
                        mx = i;
                else if(cnt[i] < cnt[mx])
                        mx = i;
        }
        res += cnt[mx];
        cnt[mx] = 0;
        riengBiet--;
    }

    cout << res << endl;
    return 0;
}
```

Đọc code đầy đủ hơn ở *đây*

2.28 Bài ZB - Paliny 1

2.28.1 Hướng dẫn

- Với bài toán này, ta sẽ duyệt trâu toàn bộ các cặp (l , r) (l <= r) và kiểm tra xâu con trong đoạn [l , r] có phải là xâu đối xứng hay không.

2.28.2 Code mẫu

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

int n;
string s;

int main()
{
    cin >> n >> s;

    int res = 1;

    for(int i = 0 ; i < s.size() ; i++){
        for(int j = i ; j < s.size() ; j++){
            bool ok = true;
            int l = i , r = j;
            while(l < r){
                if(s[l] != s[r]){
                    ok = false;
                    break;
                }
                l++;
                r--;
            }
            if(ok == true)
                res = max(res , j - i + 1);
        }
    }

    cout << res << endl;
}
```

Đọc code đầy đủ hơn ở *đây*