

Câu 2:(Lập Trình)

- Nhập vào chuỗi S, và gán S là Họ và Tên của sinh viên
- Tăng kích thước của chuỗi S lên thành 36 ký tự
- VD: ký tự của chuỗi S là ViSaoLangLe thì chuỗi S mới 36 ký tự là ViSaoLangLeViSaoLangLeViSaoLangLeViS
- Sắp xếp các ký tự của chuỗi mới vào một ma trận 6 x 6 (lần lượt theo từng hàng)
- Đọc các phần tử của ma trận theo từng cột để tạo thành ciphertext
- Giải mã ciphertext, so sánh kết quả với chuỗi ban đầu.

```
#include<iostream>
#include<string>
using namespace std;
int main() {
    string s,c;
    int k[5];
    char A[5][5];
    cout<<"Nhập chuỗi s: ";
    getline(cin,s);
    cout<<"Nhập khóa k: "<<endl;
    for(int i=0;i<5;i++){
        cout<<"k"<<i+1<<": ";
        cin>>k[i];
    }
    if(s.size()<25) {

        int dem;
        dem= 25 - s.size();
        int staticChuoi=0;
        while (dem!=0) {
            s=s+s[staticChuoi];
            staticChuoi++;
            dem--;
        }
    }
    int index=0;
    for (int i=0;i<5;i++){
        for (int j=0;j<5;j++){
            A[i][j]=s[index];
```

```

        cout<<A[i][j]<<" ";
        index ++;
    }
    cout<<endl;
}
cout<<"-----"<<endl;
//ma hoa
for (int j=0;j<5;j++){
    for (int i=0; i<5;i++){
        c=c+A[i][k[j]-1];
    }
}
cout<<endl<<"chuoi ciphertext: "<<c<<endl<<endl;

//giai ma
string giaima;
int index1=0;
for (int i=0;i<5;i++){
    for (int j=0;j<5;j++){
        A[j][k[i]-1]=c[index1];
        index1++;
    }
}

for(int i=0;i<5;i++){
    for(int j=0;j<5;j++)
        giaima=giaima + A[i][j];
}
cout<<"Chuoi plaintext: "<<giaima<<endl;

return 0;
}

```

Một trong những kỹ thuật mật mã được sử dụng nhiều trong quá khứ là lập một ma trận gồm nhiều hàng và cột. Sau đó ghi thông điệp(Sau khi đã loại bỏ các khoảng trắng) vào từng hàng nếu đầy hàng thì xuống hàng kế tiếp. Sau khi đã ghi xong thì người ta ghi lại các từ đó theo cột và gửi thông điệp đó đi. Ví dụ ta muốn gửi một thông điệp như sau:

- "Chiều nay chúng ta sẽ xem phim ở rạp REX nhé." (Viết vào ma trận theo hàng ngang mã hóa theo cột dọc)

- Giả sử ta chia làm 5 cột thì nó như sau :

Ma trận
C h i e u
n a y e h
u n g t a
s e x e m
p h i m o
r a p r e
x n h e .

- Thông điệp được ghi như sau:

- "Cnusprxhanehaniygxiphectemreuhamoe."

- Hãy viết chương trình mã hóa và giải mã thông điệp

```
#include <stdio.h>

#include<iostream>

#include <cctype>

#include <stdlib.h>

#include<conio.h>

char *crypt(char *tdiep, int column)
{
    char tam[255], *result;
    int i = 0, k = 0, n, j=0;

    while(tdiep[i] != 0)
    {
        if (isalnum(tdiep[i]))
            tam[k++] = tdiep[i];
        i++;
    }
    tam[k] = 0;
    result = (char *)malloc(k+1);
    for (i=0; i<column; i++)
    {
        n = 0;
        while(n+i < k)
        {
            result[j++] = tolower(tam[n+i]);
            n += column;
        }
    }
}
```

```
    }  
}  
result[k] = 0;  
return result;  
}  
  
int main()  
{  
    char thongdiep[255], *mahoa;  
    int col;  
  
    printf("\nNhap thong diep can ma hoa : ");  
    gets(thongdiep);  
    printf("\nCho biet so cot : ");  
    scanf("%d", &col);  
    mahoa = crypt(thongdiep, col);  
    printf("\nThong diep da duoc ma hoa thanh : %s", mahoa);  
    getch();  
}
```

ĐỀ CƯƠNG AN TOÀN BẢO MẬT THÔNG TIN

Bài tập 1:

- Lập trình nhập một chuỗi kí tự từ bàn phím, cộng mỗi phần tử của chuỗi với 3. Hiện chuỗi mới ra màn hình.

- Lập trình khôi phục lại chuỗi ban đầu.

int main()

```
{ string s;

    cout << "Moi nhap chuoi: "; cin >> s;

    for (int i = 0; i < s.size(); i++) { s[i] = s[i] + 3; }

    cout << "Chuoi vua nhap la: " << s << "\n";

    for (int i = 0; i < s.size(); i++) { s[i] = s[i] - 3; }

    cout << "Chuoi sau khi ma hoa la: " << s;

    return 0; }
```

* KỸ THUẬT THAY THẾ:

1. MẬT MÃ CAESAR

◆ Bài tập 1:

Định nghĩa: Mỗi kí tự trong bảng chữ cái được thay thế bởi một kí tự khác cùng bảng, cách sau nó ba vị trí.

Ví dụ: Plaintext: Ciphertext:

a b c d e f g h i j k l m n o p q r s t u v w x y z || D E F G H I J K L M N O P Q R S T U V W X Y
Z A B C

a	b	c	d	e	f	g	h	i	j	k	l	m
0	1	2	3	4	5	6	7	8	9	10	11	12

n	o	p	q	r	s	t	u	v	w	x	y	z
13	14	15	16	17	18	19	20	21	22	23	24	25

Công thức mã hoá: $C = E(P, 3) = (P + 3) \bmod 26$

Công thức giải mã: $P = D(C, 3) = (C - 3) \bmod 26$

Ví dụ:

◆ Mã hóa: Với $P = 'A' = 0$ thì $C = (0 + 3) \bmod 26$

$$C = 3 \bmod 26 = 3 = 'D'$$

Với $P = 'X' = 23$ thì $C = (23+3) \bmod 26$

$$C = 26 \bmod 26 = 0 = 'A'$$

◆ **Giải mã:** Với $C = 'A' = 0$ thì $P = (0 - 3) \bmod 26$
 $P = -3 \bmod 26 = -3$

Chú ý: Đối với a thuộc Z_{26} thì $\mathbf{a + 26 = a}$

$$P = -3 + 26 = 23 = 'X'$$

Bài tập 1:

- Lập trình nhập một chuỗi ký tự từ bàn phím, mã hoá chuỗi bằng thuật toán CAESAR tổng quát với khoá K nhập từ bàn phím. Hiện chuỗi mới ra màn hình.

- Lập trình giải mã để khôi phục lại chuỗi ban đầu.

```
string A = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
```

```
int kyTu_So(char c) { for (int i = 0; i < 26; i++) {  
    if (A[i] == c) { return i; } } }
```

```
char maHoa(char chuoi, int matMa) {  
    int giatriMaChuoi = kyTu_So(chuoi);  
    giatriMaChuoi = (giatriMaChuoi + matMa) % 26;  
    return A[giatriMaChuoi]; }
```

```
char giaMa(char chuoi, int matMa) {  
    int giaTriGiaiChuoi = kyTu_So(chuoi);  
    giaTriGiaiChuoi = (giaTriGiaiChuoi + 26 - matMa) % 26;  
    return A[giaTriGiaiChuoi]; }
```

```
int main()
```

```
{ string chuoi; int matma;  
    cout << "Nhap plain text:"; cin >> chuoi;  
    cout << "Nhap khoa:"; cin >> matma;  
    for (int i = 0; i < chuoi.size(); i++){  
        chuoi[i] = maHoa(chuoi[i], matma);}  
    cout << "Chuoi ma hoa:" << chuoi << "\n";  
    for (int i = 0; i < chuoi.size(); i++){  
        chuoi[i] = giaMa(chuoi[i], matma); }  
    cout << "Chuoi giai ma:" << chuoi;
```

```
return 0; }
```

Bài tập 2:

Lập trình bẻ khoá mật mã Caesar bằng phương pháp Brute-force:

- Đầu vào chương trình là chuỗi kí tự cipher text thu được từ Bài tập 1.
- Hãy xác định khoá K đã sử dụng và nội dung của plain text ban đầu.

```
string A = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
```

```
int kyTu_So(char c) {
```

```
    for (int i = 0; i < 26; i++) {
```

```
        if (A[i] == c) {return i; } }
```

```
char maHoa(char chuoi, int matMa) {
```

```
    int giatriMaChuoi = kyTu_So(chuoi);
```

```
    giatriMaChuoi = (giatriMaChuoi + matMa) % 26;
```

```
    return A[giatriMaChuoi]; }
```

```
int main()
```

```
{    string chuoiDaMa = "KHPOVJAOBFSVP";
```

```
    string chuoiSudung = chuoiDaMa;
```

```
    for (int khoa = 0; khoa < 26; khoa++) {
```

```
        for (int i = 0; i < chuoiDaMa.size(); i++) {
```

```
            chuoiDaMa[i] = maHoa(chuoiDaMa[i], khoa);}
```

```
        cout << chuoiDaMa<<"\n"; cout << khoa;
```

```
        chuoiDaMa = chuoiSudung; }
```

```
    return 0; }
```

2. MẬT MÃ AFFINE

- ◆ Kí tự P ban đầu được thay thế bởi kí tự C theo công thức:

$$C = E(P, \{a, b\}) = (aP + b) \bmod 26$$

- ◆ Công thức giải mã:

$$P = D(C, \{a, b\}) = a^{-1}(C - b) \bmod 26$$

Trong đó khoá K chính là cặp 2 số nguyên $\{a, b\}$ thuộc Z_{26} . Nếu $a = 1$ thì thành mật mã Caesar.

- ◆ **Bài tập 1:** Xác định các giá trị có thể có của a trong Z_{26} , và tính a^{-1} tương ứng

Các giá trị của a : 1, 3, 5, 7, 9, 11, 15, 17, 19, 21, 23, 25

Các a^{-1} tương ứng: 1, 9, 21, 15, 3, 19, 7, 23, 11, 5, 17, 25

Bài tập 2:

- Lập trình nhập một chuỗi kí tự từ bàn phím, mã hoá chuỗi bằng thuật toán Affine với cặp số $\{a,b\}$ nhập từ bàn phím. Hiện chuỗi mới ra màn hình.

- Lập trình giải mã Affine để khôi phục lại chuỗi ban đầu:

- Đầu vào chương trình là cặp số $\{a,b\}$ và chuỗi kí tự cipher text từ Bài tập 1.

- Đầu ra chương trình là chuỗi kí tự plain text

```
bool KiemTraA(int a) {
    for (int i = 1; i <= 25; i++) {
        if ((i*a) % 26 == 1) {return true;break; } }
    return false; }

int NghichDao(int a) {
    for (int i = 1; i <= 25; i++) {
        if ((i*a) % 26 == 1) { return i; } } }

string M = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";

int KiTu_So(char chuoi) {
    for (int i = 0; i < 26; i++) { if (chuoi == M[i]) { return i; } } }

char MaHoa(char chuoi, int soT1, int soT2) {
    int ma = KiTu_So(chuoi); ma = (soT1*ma + soT2) % 26;
    return M[ma]; }

char GiaiMa(char chuoi, int a, int b) {
    int giai = KiTu_So(chuoi);
    giai = NghichDao(a)*((giai - b + 26) % 26);
    return M[giai];}

int main()
{ bool check = false; string s; int a, b;
    cout << "Nhap chuoi : "; cin >> s;
    int N[] = { 1, 3, 5, 7, 9, 11, 15, 17, 19, 21, 23, 25 };
    while (!check) {
        cout << "Nhap a : ";cin >> a;
```



```

for (int i = 0; i < 12; i++) {
    if (KiemTraA(a) == true) { break; }
}
cout << "Nhap b : "; cin >> b;
for (int i = 0; i < s.size(); i++) {
    s[i] = MaHoa(s[i], a, b); cout << s[i];
}
cout << "Chuoi ma hoa la:" << s << endl;
//Giai ma
for (int i = 0; i < s.size(); i++) {
    s[i] = GiaiMa(s[i], a, b);
}
cout << "Chuoi giai ma : " << s << endl;
return 0;}

```

Bài tập 3:

Lập trình bẻ khoá mật mã Affine bằng phương pháp Brute-force:

- Đầu vào chương trình là chuỗi kí tự cipher text thu được từ Bài tập 1.
- Hãy xác định cặp số $\{a, b\}$ đã sử dụng và nội dung của plain text ban đầu.

```

int NghichDao(int a) {
    for (int i = 1; i <= 25; i++) {
        if ((i*a) % 26 == 1) { return i; } } }
string M = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
int KiTu_So(char chuoi) {
    for (int i = 0; i < 26; i++) {
        if (chuoi == M[i]) { return i; } } }
char GiaiMa(char chuoi, int a, int b) {
    int giai = KiTu_So(chuoi);
    giai = NghichDao(a)*((giai - b + 26) % 26);
    return M[giai] }
int main()

```

```

{ bool check = false; string s, s2; s = "JQE"; s2 = s;

    int N[] = { 1, 3, 5, 7, 9, 11, 15, 17, 19, 21, 23, 25 };

    for (int b = 0; b < 26; b++)

        for (int j = 0; j < 12; j++) {

            for (int i = 0; i < s.size(); i++) { s[i] = GiaiMa(s[i], N[j], b); }

            cout << N[j] << ", " << b << ", " << s << endl; s = s2;;

        } return 0; }

```

3. MẬT MÃ MONOALPHABETIC

Định nghĩa: Mỗi kí tự trong bảng chữ cái được thay thế bởi một kí tự bất kì khác cùng bảng.

- ◆ Bản rõ: ABCDEFGHIJKLMNOPQRSTUVWXYZ
- ◆ Bản mã: QWERTYUIOPASDFGHJKLZXCVBNM

Bài tập 1:

- Lập trình nhập một chuỗi kí tự từ bàn phím, mã hoá chuỗi bằng thuật toán Monoalphabetic với khoá K nhập từ bàn phím (Khóa K là một chuỗi gồm 26 chữ cái có trật tự bất kì). Hiện chuỗi mới ra màn hình.

Lập trình giải mã Monoalphabetic để khôi phục lại chuỗi ban đầu.

string M = "ABCDEFGHIJKLMNOPQRSTUVWXYZ".

```

string K = "qwertyuiopasdfghjklzxcvbnm";

char maHoa(char c) { for(int i = 0; i < 26; i++) { if(c == M[i]) { return K[i]; break; } } }

char giaMa(char c) { for(int i = 0; i < 26; i++) { if(c == K[i]) { return M[i]; break; } } }

int main() { string s; cin >> s; cout << "Nhap chuoi: "; cin >> s; cout << "Nhap khoa: "; cin >> k;

    for (int i = 0; i < s.size(); i++) { s[i] = maHoa(s[i]); }
    cout << "Chuoi ma hoa la: " << s;
    for (int i = 0; i < s.size(); i++) { s[i] = giaMa(s[i]); }
    cout << "Giai la: " << s;

    return 0; }

```

- ◆ **Bài tập 2:** Lập trình tính tần suất xuất hiện của các kí tự trong một đoạn văn bản cho trước.

```

string M = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";

int KiTu_So(char c) {

    for (int i = 0; i < 26; i++) { if (c == M[i]) { return i; } } }

char So_KiTu(int n) { return M[n]; }

void SoLanXuatHien(string a) {

    int b[27];

```

```

for (int i = 0; i < 27; i++) { b[i] = 0;}

for (int i = 0; i < a.size(); i++) {
    for (int j = 0; j < M.size(); j++) {
        if (a[i] == M[j]) {b[j]++;}
        else { b[26]++;} } }

for (int i = 0; i < 27; i++) {
    cout << M[i] << " : " << b[i] << endl;} }

int main() {
    string s; cout << "Nhap chuoi ki tu : "; getline(cin, s);
    SoLanXuatHien(s);    return 0; } }

```

3. MẬT MÃ PLAYFAIR <Chú ý bài tập mã hóa bằng tay trong vở>>

- ◆ Định nghĩa: Mật mã Playfair sẽ thay thế từng cặp 2 kí tự trong bản rõ bởi 2 kí tự tương ứng trong ma trận khoá 5 x 5.
- ◆ Ví dụ, nếu chọn từ khoá là *monarchy* thì ma trận khoá sẽ như sau:

M	O	N	A	R
C	H	Y	B	D
E	F	G	I/J	K
L	P	Q	S	T
U	V	W	X	Z

- + Lần lượt viết từng kí tự của khóa vào ma trận, từ trái sang phải, từ trên xuống dưới, bỏ các kí tự trùng lặp
- + Viết các ký tự còn lại trong bảng chữ cái vào ma trận theo thứ tự, I và J được coi như một ký tự .
- + Mỗi ký tự trong cặp plaintext sẽ được mã hoá bằng ký tự nằm cùng hàng với nó, nhưng cùng cột với ký tự kia
- + Giả sử trong plaintext có cặp kí tự HS, nó sẽ được thay thế bởi cặp BP.
 - ◆ Nếu cặp ký tự plaintext rơi vào cùng một hàng của ma trận thì mỗi ký tự được thay thế bởi ký tự bên phải nó.
 - ◆ Nếu ký tự plaintext rơi vào cột cuối cùng, thì ciphertext của nó là ký tự cùng hàng ở cột đầu tiên.

- ◆ Ví dụ, AR sẽ được mã hóa thành RM
- ◆ Nếu cặp ký tự plaintext rơi vào chung một cột của ma trận thì mỗi ký tự được thay thế bởi ký tự ngay sát dưới.
- ◆ Nếu ký tự plaintext rơi vào hàng cuối cùng, thì ciphertext của nó là ký tự cùng cột, ở hàng đầu tiên.
- ◆ Ví dụ, MU được mã hóa thành CM.
- ◆ Nếu hai ký tự trong plaintext giống nhau thì chúng sẽ được cách ly bằng một ký tự đại diện, chẳng hạn là x.
- ◆ Ví dụ, từ **balloon** sẽ được tách ra thành **ba lx lo on**.

Bài tập 1: Lập trình nhập 1 chuỗi ký tự từ bàn phím. Hãy loại bỏ các ký tự trùng lặp trong chuỗi. =>>>> Điền thêm các ký tự còn lại trong bảng chữ cái vào. =>>> Đặt chuỗi thu được vào ma trận 5x5.

```
string M = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
```

```
string loạiBoKyTuTrungLap(string &s) {
    for (int i = 0; i < s.size(); i++)
        for (int j = i + 1; j < s.size(); j++)
            if (s[j] == s[i]) { s.erase(j, 1); j--; } return s; }

string themChuoiConLaiVao(string &s) {
    for (int i = 0; i < s.size(); i++)
        for (int j = 0; j < M.size(); j++)
            if (M[j] == s[i]) { M.erase(j, 1); j--; } s = s + M; return s; }

int main() {
    string s; char A[5][5];
    cout << "Moi nhap chuoi khoa s:"; cin >> s;
    string khoa = loạiBoKyTuTrungLap(s); cout << "Chuoi la:" << khoa;
    string chuoiMH = themChuoiConLaiVao(s);
    cout << endl << "Chuoi moi la:" << chuoiMH; int t = 0;
    for(int i = 0; i < 5; i++)
        for (int j = 0; j < 5; j++) { A[i][j] = chuoiMH[t]; t++; }
    cout << "Ma tran khoa la:" << endl;
    for (int i = 0; i < 5; i++){
```

```

        for (int j = 0; j < 5; j++) cout << A[i][j] << " "; cout << endl;
    }
    return 0; }

```

4. MẬT MÃ HILL <Chú ý bài tập mã hóa bằng tay trong vở>>

◆ Định nghĩa: Mật mã Hill sẽ thay thế từng nhóm m kí tự trong plaintext bởi m kí tự ciphertext

$$\begin{pmatrix} c_1 \\ c_2 \\ c_3 \end{pmatrix} = \begin{pmatrix} k_{11} & k_{12} & k_{13} \\ k_{21} & k_{22} & k_{23} \\ k_{31} & k_{32} & k_{33} \end{pmatrix} \begin{pmatrix} p_1 \\ p_2 \\ p_3 \end{pmatrix} \bmod 26$$

$$\mathbf{P} = \mathbf{K}^{-1}\mathbf{C} \bmod 26$$

với:

- ◆ \mathbf{K}^{-1} là ma trận nghịch đảo của ma trận khoá \mathbf{K}
- ◆ tức là $\mathbf{K} \cdot \mathbf{K}^{-1} = \mathbf{K}^{-1} \cdot \mathbf{K} = \mathbf{I}$ (\mathbf{I} là ma trận đơn vị)

Ví dụ chọn

$$\mathbf{K} = \begin{pmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{pmatrix} \quad \mathbf{K}^{-1} = \begin{pmatrix} 4 & 9 & 15 \\ 15 & 17 & 6 \\ 24 & 0 & 17 \end{pmatrix}$$

Vì:

$$\begin{pmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{pmatrix} \begin{pmatrix} 4 & 9 & 15 \\ 15 & 17 & 6 \\ 24 & 0 & 17 \end{pmatrix} = \begin{pmatrix} 443 & 442 & 442 \\ 858 & 495 & 780 \\ 494 & 52 & 365 \end{pmatrix} \bmod 26 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

5. MẬT MÃ POLYALPHABETIC <<Chú ý mã hóa bằng tay tham khảo trong vở ghi>>.

- ◆ Mật mã Monoalphabetic chỉ sử dụng một bảng mã (mỗi kí tự plain text được thay thế bởi một kí tự cố định), nên không giấu được tần suất xuất hiện các kí tự
- ◆ Còn mật mã Polyalphabetic lại sử dụng nhiều bảng mã khác nhau (mỗi kí tự plain text có thể được thay thế bởi nhiều kí tự khác nhau, dựa trên các khoá thay thế khác nhau)

◆ Giả sử lấy $m=6$ và khoá K là CIPHER, plaintext là:

WEWILLMEETATMIDNIGHT

Hãy xác định ciphertext. |

◆ Do $m=6$, ta sẽ tách plaintext thành từng nhóm 6 kí tự:

WEWILL/MEETAT/MIDNIG/HT

Viết theo dạng số là:

22 4 22 8 11 11 / 12 4 4 19 0 19 / 12 8 3 13 8 6 / 7 19

◆ Từ khoá CIPHER tương ứng với:

$K = (2, 8, 15, 7, 4, 17)$

◆ Cộng từng nhóm 6 kí tự của plaintext với K ta có:

22	4	22	8	11	11	/	12	4	4	19	0	19	/	12	8	3	13	8	6	/	7	19
2	8	15	7	4	17	/	2	8	15	7	4	17	/	2	8	15	7	4	17	/	2	8
<hr/>																						
24	12	11	15	15	2	/	14	12	19	0	4	10	/	14	16	18	20	12	23	/	9	1

◆ Ciphertext tương ứng là:

YMLPPCOMTAEKOQSUMXJB

◆ **Bài tập 1:** Lập trình nhập chuỗi kí tự plaintext từ bàn phím, mã hoá chuỗi bằng thuật toán Vigenère với khoá K (là một chuỗi kí tự) nhập từ bàn phím. Hiện chuỗi mới ra màn hình.

Lập trình giải mã để khôi phục lại chuỗi ban đầu.

```
string M = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
```

```
int kyTu_So(char c) {
```

```
    for (int i = 0; i < 26; i++) {if (M[i] == c) {return i;} } }
```

```
char so_kyTu(int n) { return M[n]; }
```

```
char maHoa(char chuoi, char khoa) {
```

```
    int a = kyTu_So(chuoi), b = kyTu_So(khoa);
```

```
    a = (a + b + 26) % 26;
```

```
    return so_kyTu(a);
```

```
}
```

```

char giaiMa( char chuoi, char khoa) {
    int a = kyTu_So(chuoi), b = kyTu_So(khoa); a = (a - b + 26) % 26; return so_kyTu(a);
}

int main() {
    string s, k; cout << "Nhap plaintext:"; cin >> s;
    cout << "Nhap khoa k:"; cin >> k; int m = k.size();
    for (int i = 0; i < s.size(); i++){
        s[i] = maHoa(s[i], k[i%m]);}
    cout << "Chuoi ma hoa la:" << s;
    for (int i = 0; i < s.size(); i++) {
        s[i] = giaiMa(s[i], k[i%m]); }
    cout << endl<< "Chuoi giai ma la:" << s;    return 0; }

```

6. Mật mã VERNAM

- ◆ Plaintext được biểu diễn dưới dạng một chuỗi bit nhị phân
- ◆ Khoá K cũng được biểu diễn dưới dạng một chuỗi bit nhị phân (càng dài càng tốt, càng ngẫu nhiên càng tốt)
- ◆ Ciphertext được sinh ra bởi phép XOR giữa plaintext với khoá K

Ví dụ:

◆ Mã hoá:	Plaintext =	0001 0110 ...
	Khoá K =	0000 1101 ...
	Ciphertext =	0001 1011 ...
◆ Giải mã:	Ciphertext =	0001 1011 ...
	Khoá K =	0000 1101 ...
	Plaintext =	0001 0110 ...

Bảng mã ASGII:

100 0001	65	41	A
100 0010	66	42	B
100 0011	67	43	C
100 0100	68	44	D
100 0101	69	45	E
100 0110	70	46	F
100 0111	71	47	G
100 1000	72	48	H
100 1001	73	49	I
100 1010	74	4A	J
100 1011	75	4B	K
100 1100	76	4C	L
100 1101	77	4D	M
100 1110	78	4E	N

100 1111	79	4F	O
101 0000	80	50	P
101 0001	81	51	Q
101 0010	82	52	R
101 0011	83	53	S
101 0100	84	54	T
101 0101	85	55	U
101 0110	86	56	V
101 0111	87	57	W
101 1000	88	58	X
101 1001	89	59	Y
101 1010	90	5A	Z

110 0001	97	61	a
110 0010	98	62	b
110 0011	99	63	c
110 0100	100	64	d
110 0101	101	65	e
110 0110	102	66	f
110 0111	103	67	g
110 1000	104	68	h
110 1001	105	69	i
110 1010	106	6A	j
110 1011	107	6B	k
110 1100	108	6C	l
110 1101	109	6D	m
110 1110	110	6E	n

110 1111	111	6F	o
111 0000	112	70	p
111 0001	113	71	q
111 0010	114	72	r
111 0011	115	73	s
111 0100	116	74	t
111 0101	117	75	u
111 0110	118	76	v
111 0111	119	77	w
111 1000	120	78	x
111 1001	121	79	y
111 1010	122	7A	z

HILLCIPHER:

- Interesting multicipher cipher is the hillcipher
- The encryption algorithm takes m successive plaintext letters and substitutes for them m ciphertext letters.
- Ex:

$$C1 = (K11 P1 + K12 P2 + K13 P3) \text{ mod } 26.$$

$$C2 = (K21 P1 + K22 P2 + K23 P3) \text{ mod } 26.$$

$$C3 = (K31 P1 + K32 P2 + K33 P3) \text{ mod } 26.$$

This can be expressed in terms of column vectors and matrices

$$\begin{bmatrix} C1 \\ C2 \\ C3 \end{bmatrix} = \begin{bmatrix} P1 & P2 & P3 \end{bmatrix} * \begin{bmatrix} K11 & K12 & K13 \\ K21 & K22 & K23 \\ K31 & K32 & K33 \end{bmatrix} \text{ mod } 26$$

$$\mathbf{C} = \mathbf{PK} \text{ mod } 26$$

- C & P are the column vectors of length 3.
- K is a 3*3 matrix.

Example:

Plaintext = “paymoremoney”

$$C = \begin{bmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{bmatrix}$$

Encryption:

Plaintext: pay more money

3 * 3 matrix is our key. So split it 3 bit distance.

Plaintext:

P	15
A	0
y	24

M	12
O	14
r	17

E	4
M	12
o	17

N	13
E	4
Y	24

Calculate the $C = KP \bmod 26$

$$\begin{aligned} C_{pay} &= \begin{bmatrix} 15 & 0 & 24 \end{bmatrix} * \begin{bmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{bmatrix} \\ &= \begin{bmatrix} 255 + 0 + 48 \\ 255 + 0 + 48 \\ 75 + 0 + 456 \end{bmatrix} = \begin{bmatrix} 303 \\ 303 \\ 531 \end{bmatrix} \text{ take mod 26.} \end{aligned}$$

$$C = (17 \quad 17 \quad 11) \rightarrow (R \quad R \quad L)$$

$$\begin{aligned} C_{mor} &= \begin{bmatrix} 12 & 14 & 17 \end{bmatrix} * \begin{bmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{bmatrix} \\ &= \begin{bmatrix} 204 + 294 + 34 \\ 204 + 252 + 34 \\ 60 + 294 + 323 \end{bmatrix} = \begin{bmatrix} 532 \\ 490 \\ 677 \end{bmatrix} \text{ take mod 26.} \end{aligned}$$

$$C = (12 \quad 22 \quad 1) \rightarrow (M \quad W \quad B)$$

$$\begin{aligned} C_{emo} &= \begin{bmatrix} 4 & 12 & 14 \end{bmatrix} * \begin{bmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{bmatrix} \\ &= \begin{bmatrix} 68 + 252 + 28 \\ 68 + 216 + 28 \\ 20 + 252 + 266 \end{bmatrix} = \begin{bmatrix} 348 \\ 312 \\ 538 \end{bmatrix} \text{ take mod 26.} \end{aligned}$$

$$C = (10 \quad 0 \quad 18) \rightarrow (K \quad A \quad S)$$

$$C_{ney} = \begin{bmatrix} 13 & 4 & 24 \end{bmatrix} * \begin{bmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{bmatrix}$$

$$= \begin{bmatrix} 221 + 84 + 48 \\ 221 + 72 + 48 \\ 65 + 84 + 456 \end{bmatrix} = \begin{bmatrix} 353 \\ 341 \\ 605 \end{bmatrix} \text{ take mod 26.}$$

$$C = (15 \quad 3 \quad 7) \rightarrow (P \quad D \quad H)$$

CIPHERTEXT : (RRL MWB KAS PDH)

DECRYPTION:

1. First calculate K^{-1}
2. $P = K^{-1} C \text{ mod } 26$
 $= K^{-1} (KP) = P \quad (C=KP \text{ mod } 26)$

Calculating K^{-1} :

Formula : $K^{-1} = \frac{1}{|K|} \text{adj}(K)$

$$K = \begin{bmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{bmatrix}$$

$$|K| = k_{11}[(k_{22} * k_{33}) - (k_{32} * k_{23})] - k_{12} [(k_{21} * k_{33}) - (k_{31} * k_{23})] +$$

$$k_{13} [(k_{21} * k_{32}) - (k_{31} * k_{22})]$$

$$|K| = 17 (342 - 42) - 17 (399 - 42) + 5 (42 - 36)$$

$$|K| = 5100 - 6069 + 30$$

$$|K| = -939.$$

$$|K| = -3 \text{ mod } 26 \quad \text{[to eliminate - value take "mod26"]}$$

$$|K| = -3 + 26 \equiv 23 \text{ mod } 26 \quad \rightarrow |K| \text{ mod } 26 = a$$

$$\frac{1}{|K|} = 23^{-1} = 17$$

$$\text{or } 1/x = 23 \bmod 26$$

$$23 \cdot x = 1 \pmod{26}$$

$$= \bmod 26 = 23 \cdot x$$

Using Euclid Theorem:

$$\begin{array}{rrrr} 26 & 1 & 0 & \rightarrow R1 \\ \hline 1 & 23 & 0 & 1 \rightarrow R2 \\ 7 & 3 & +1 & -1 \\ 1 & 2 & -7 & 8 \\ 1 & 8 & -9 & \end{array}$$

$$26 * 8 + 23 (-9) = 1$$

$$23 (-9) \equiv 1 \bmod 26. \quad (\text{remove the -ve symbol})$$

$$\text{So } x \equiv -9 \bmod 26$$

$$X = 17.$$

$$(-9 + 26 = 17)$$

So

$$\frac{1}{|K|} = 23^{-1} = 17$$

$$\text{Adj}(k) = (\text{cofactor})^T$$

$$K_{11} = \begin{bmatrix} 18 & 21 \\ 2 & 19 \end{bmatrix} = (342 - 42) = 300$$

$$K_{12} = \begin{bmatrix} 21 & 21 \\ 2 & 19 \end{bmatrix} = (399 - 42) = 357$$

$$K_{13} = \begin{bmatrix} 21 & 18 \\ 2 & 2 \end{bmatrix} = (42 - 36) = 6$$

$$K_{21} = \begin{bmatrix} 17 & 5 \\ 2 & 19 \end{bmatrix} = (323 - 10) = 313$$

$$K_{22} = \begin{bmatrix} 17 & 5 \\ 2 & 19 \end{bmatrix} = (323 - 10) = 313$$

$$K_{23} = \begin{bmatrix} 17 & 17 \\ 2 & 2 \end{bmatrix} = (34 - 34) = 0$$

$$K_{31} = \begin{bmatrix} 17 & 5 \\ 18 & 21 \end{bmatrix} = (357 - 90) = 267$$

$$K_{32} = \begin{bmatrix} 17 & 5 \\ 21 & 21 \end{bmatrix} = (357 - 105) = 252$$

$$K_{33} = \begin{bmatrix} 17 & 17 \\ 21 & 18 \end{bmatrix} = (306 - 357) = -51$$

$$(\text{co factor}) = \begin{bmatrix} 300 & -357 & 6 \\ -313 & 313 & 0 \\ 267 & -252 & -51 \end{bmatrix}$$

$$(\text{co factor})^T = \begin{bmatrix} 300 & -313 & 267 \\ -357 & 313 & -252 \\ 6 & 0 & -51 \end{bmatrix}$$

$$\text{Adj} (K) = (\text{cofactor})^T$$

$$\text{adj} (K) = \begin{bmatrix} 300 & -313 & 267 \\ -357 & 313 & -252 \\ 6 & 0 & -51 \end{bmatrix}$$

$$K^{-1} = \frac{1}{|K|} \text{adj} (K)$$

$$\begin{aligned}
K^{-1} &= 17 \begin{bmatrix} 300 & -313 & 267 \\ -357 & 313 & -252 \\ 6 & 0 & -51 \end{bmatrix} \\
&= \begin{bmatrix} 300 * 17 & -313 * 17 & 267 * 17 \\ -357 * 17 & 313 * 17 & -252 * 17 \\ 6 * 17 & 0 * 17 & -51 * 17 \end{bmatrix} \\
&= \begin{bmatrix} 5100 & -5321 & 4539 \\ -6069 & 5321 & -4284 \\ 102 & 0 & -867 \end{bmatrix} \\
K^{-1} &= \begin{bmatrix} 4 & -17 & 15 \\ -11 & 17 & -20 \\ 24 & 0 & -9 \end{bmatrix}
\end{aligned}$$

To remove negative (-) value take “mod 26” for each value.

$$K^{-1} = \begin{bmatrix} 4 & 9 & 15 \\ 15 & 17 & 6 \\ 24 & 0 & 17 \end{bmatrix}$$

This is demonstrated as follows:

$$\begin{aligned}
K * K^{-1} &= \begin{bmatrix} 17 & 17 & 15 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{bmatrix} \begin{bmatrix} 4 & 9 & 15 \\ 15 & 17 & 6 \\ 24 & 0 & 17 \end{bmatrix} \\
&= \begin{bmatrix} 68 + 255 + 120 & 84 + 270 + 504 & 8 + 30 + 456 \\ 153 + 289 + 0 & 189 + 306 + 0 & 18 + 34 + 0 \\ 255 + 102 + 85 & 315 + 108 + 357 & 30 + 12 + 323 \end{bmatrix} \\
&= \begin{bmatrix} 443 & 858 & 494 \\ 442 & 495 & 52 \\ 442 & 780 & 365 \end{bmatrix} \text{mod } 26 \\
&= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (\text{unit matrix})
\end{aligned}$$

- The strength of the Hillcipher is that it completely hides single- letter frequencies.
- The use of larger matrix hides more frequency information.

- Thus a 3*3 Hillcipher hides not only single-letter but two-letter frequency information.
- The Hill cipher is strong against a cipher-text only attack.

$$P = K^{-1} C \bmod 26$$

$$P = \begin{bmatrix} 4 & 9 & 15 \\ 15 & 17 & 6 \\ 24 & 0 & 17 \end{bmatrix} \begin{pmatrix} 11 \\ 13 \\ 18 \end{pmatrix}$$

$$P = \begin{bmatrix} 44 + 117 + 270 \\ 165 + 221 + 108 \\ 264 + 0 + 306 \end{bmatrix} = \begin{bmatrix} 431 \\ 494 \\ 570 \end{bmatrix} \bmod 26$$

$$P = (15 \ 0 \ 24) \Rightarrow \text{pay}$$

$$P = \begin{bmatrix} 4 & 9 & 15 \\ 15 & 17 & 6 \\ 24 & 0 & 17 \end{bmatrix} \begin{pmatrix} 7 \\ 3 \\ 11 \end{pmatrix}$$

$$P = \begin{bmatrix} 28 + 27 + 165 \\ 105 + 51 + 66 \\ 168 + 0 + 187 \end{bmatrix} = \begin{bmatrix} 220 \\ 222 \\ 355 \end{bmatrix} \bmod 26$$

$$P = (12 \ 14 \ 17) \Rightarrow \text{mor}$$

$$P = \begin{bmatrix} 4 & 9 & 15 \\ 15 & 17 & 6 \\ 24 & 0 & 17 \end{bmatrix} \begin{pmatrix} 4 \\ 22 \\ 12 \end{pmatrix}$$

$$P = \begin{bmatrix} 16 + 198 + 180 \\ 60 + 374 + 72 \\ 96 + 0 + 204 \end{bmatrix} = \begin{bmatrix} 394 \\ 506 \\ 300 \end{bmatrix} \bmod 26$$

$$P = (4 \ 12 \ 14) \Rightarrow \text{emo}$$

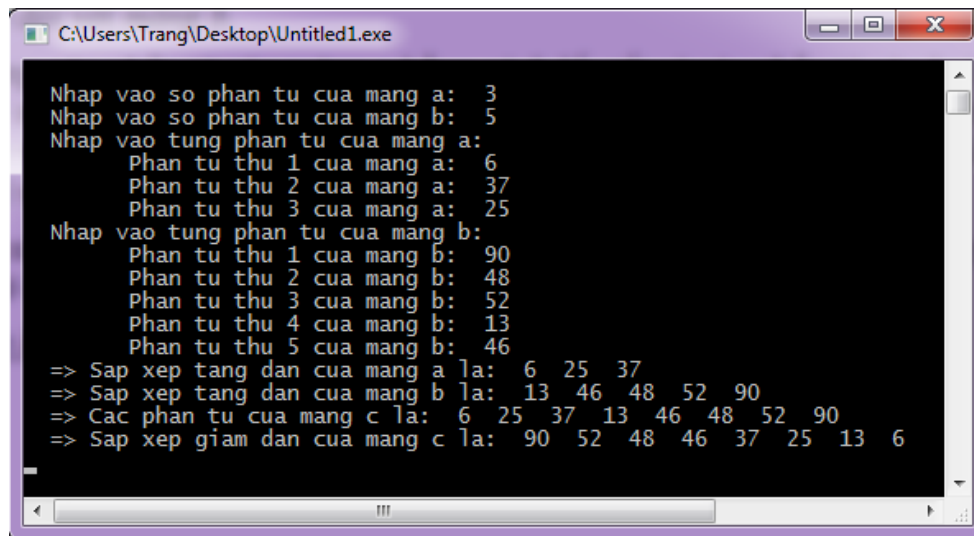
$$P = \begin{bmatrix} 4 & 9 & 15 \\ 15 & 17 & 6 \\ 24 & 0 & 17 \end{bmatrix} \begin{pmatrix} 19 \\ 17 \\ 22 \end{pmatrix}$$

$$P = \begin{bmatrix} 76 + 153 + 330 \\ 285 + 289 + 132 \\ 456 + 0 + 374 \end{bmatrix} = \begin{bmatrix} 559 \\ 706 \\ 830 \end{bmatrix} \bmod 26$$

$$P = (13 \ 4 \ 24) = > \text{ney}$$

$$P = \begin{pmatrix} 15 & 0 & 24 \\ 12 & 14 & 17 \\ 4 & 12 & 14 \\ 13 & 4 & 24 \end{pmatrix} = \text{pay mor emo ney}$$

Finally the hillcipher isproved.



```

C:\Users\Trang\Desktop\Untitled1.exe
Nhap vao so phan tu cua mang a: 3
Nhap vao so phan tu cua mang b: 5
Nhap vao tung phan tu cua mang a:
  Phan tu thu 1 cua mang a: 6
  Phan tu thu 2 cua mang a: 37
  Phan tu thu 3 cua mang a: 25
Nhap vao tung phan tu cua mang b:
  Phan tu thu 1 cua mang b: 90
  Phan tu thu 2 cua mang b: 48
  Phan tu thu 3 cua mang b: 52
  Phan tu thu 4 cua mang b: 13
  Phan tu thu 5 cua mang b: 46
=> Sap xep tang dan cua mang a la: 6 25 37
=> Sap xep tang dan cua mang b la: 13 46 48 52 90
=> Cac phan tu cua mang c la: 6 25 37 13 46 48 52 90
=> Sap xep giam dan cua mang c la: 90 52 48 46 37 25 13 6

```

Bài 97: Nhập vào một mảng 2 chiều $m \times n$ ($m, n < 5$) gồm các số thực:

- Nhập vào một số nguyên dương k ($0 \leq k < m$; $0 \leq k < n$). Tính và đưa ra màn hình:

+ Tích các phần tử ở cột thứ k .

+ Số lượng các phần tử là số nguyên tố trên dòng thứ k .

- Tìm phần tử có giá trị nhỏ nhất trong mảng vừa nhập. Có bao nhiêu phần tử có giá trị bằng giá trị nhỏ nhất đó.

(Đề thi cuối kỳ lớp THCS 3, ĐHKHTN – ĐHQGHN, Thứ 6 ngày 21/12/2012, Kỳ I năm học 2012 – 2013, thời gian làm bài: 40 phút)

CHƯƠNG VI. XÂU KÍ TỰ

Bài 97: Viết chương trình nhập vào từ bàn phím một chuỗi ký tự. Thực hiện:

- Tính xem có bao nhiêu ký tự trong chuỗi đó.

- Đọc một ký tự từ bàn phím, tìm xem chuỗi đó có bao nhiêu ký tự giống với ký tự vừa đọc. Đưa kết quả ra màn hình? (X)

```
#include <stdio.h>
```

```
#include <conio.h>
```

```
main()
```

```

{
    char XauKiTu[1000],c;
    int i, Dem=0, SoKyTuGiong=0;
    printf("\n Nhap vao mot xau ki tu: ");
    gets(XauKiTu);
    while (XauKiTu[Dem]!='\0')Dem++;
    printf(" => So ki tu trong xau do la: %d\n",Dem);
    printf(" Nhap vao mot ki tu tu ban phim: ");
    scanf("%c",&c);
    for(i=0;i<Dem;i++)
        if(XauKiTu[i]==c)SoKyTuGiong++;
    printf(" => So ki tu cua xau giong voi ki tu %c vua nhap la:
%d\n",c,SoKyTuGiong);
    fflush(stdin);
    getch();
    return main();
}

```

** Giải thích:*

a) *Xâu ký tự có n ký tự thực chất chỉ là một mảng 1 chiều gồm n phần tử. Khai báo kiểu ký tự cho xâu và giới hạn số ký tự tối đa của xâu là 1000. Ở đây phải dùng hàm đọc từ bàn phím **gets()** chứ không được dùng hàm **scanf()**. Truy nhập tới một phần tử của xâu với tên biến và chỉ số đặt trong ngoặc vuông [] tương tự như truy nhập tới một phần tử của mảng.*

b) *Tính xem có bao nhiêu ký tự trong xâu vừa nhập (hay xác định độ dài của xâu): đếm từ ký tự đầu tiên của xâu cho tới khi gặp ký tự '\0' thì dừng lại, dùng vòng lặp while.*

c) *Tìm xem trong xâu có bao nhiêu ký tự giống với ký tự được nhập vào từ bàn phím: dùng vòng for với biến i chạy từ 0 đến n – 1 (n là số ký tự của xâu). Nếu phần tử nào trong xâu giống với ký tự được nhập từ bàn phím thì tăng biến **SoKiTuGiong** lên 1 đơn vị.*

d) *Lệnh **fflush(stdin)** có tác dụng xóa sạch bộ nhớ đệm bàn phím để đi vào lần nhập xâu ký tự khác. Nếu không dùng lệnh **return main()** thì cũng không cần dùng lệnh này.*

** Màn hình kết quả như sau:*

```

C:\Users\Anh Ho\Deskto\Untitled1.exe
Nhap vao mot xau ki tu: Hoang Van Trong
=> So ki tu trong xau do la: 15
Nhap vao mot ki tu tu ban phim: n
=> So ki tu cua xau giống với ki tu n vừa nhập là: 3

Nhap vao mot xau ki tu: Đại học khoa học tự nhiên, Đại học quốc gia Hà Nội
=> So ki tu trong xau do la: 50
Nhap vao mot ki tu tu ban phim: o
=> So ki tu cua xau giống với ki tu o vừa nhập là: 6

```

Bài 98: Nhập vào từ bàn phím một chuỗi ký tự S. Đếm số lần xuất hiện của ký tự a trong S (a được nhập từ bàn phím) và kiểm tra xem S có phải là chuỗi đối xứng hay không?

```

#include <stdio.h>
#include <conio.h>
#include <string.h>
int DoiXung(char a[],int n);
main()
{
    char S[100],a;
    int i,GiongNhau=0;
    printf("\n Nhập vào một chuỗi ký tự S: ");
    gets(S);
    printf(" Nhập vào một ký tự nào đó từ bàn phím: ");
    scanf("%c",&a);
    for(i=0;i<strlen(S);i++)
        if(S[i]==a)GiongNhau++;
    printf(" => Số lần xuất hiện của ký tự %c trong chuỗi S là:
%d\n",a,GiongNhau);
    if(DoiXung(S,strlen(S))==1)printf(" => Chuỗi vừa nhập là một chuỗi đối
xung.\n");
    else printf(" => Chuỗi vừa nhập không là một chuỗi đối xứng.\n");
    fflush(stdin);
    getch();
}

```

```

return main();
}
int DoiXung(char a[],int n)
{
    int k=1,i;
    for(i=0;i<n;i++)
        if(a[i]!=a[n-i-1])k=0;
    return (k);
}

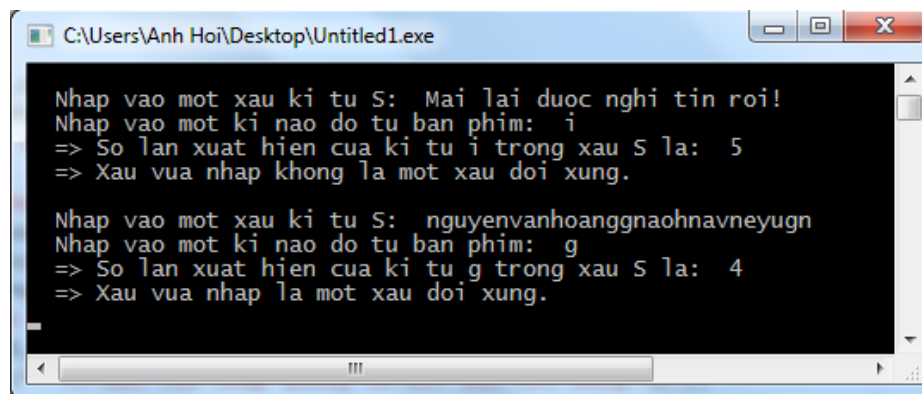
```

* Giải thích:

a) Đếm số lần xuất hiện của kí tự *a* trong chuỗi *S*: Dùng vòng lặp *for* duyệt qua từng kí tự của chuỗi *S*, nếu kí tự nào của chuỗi *S* mà giống với kí tự được nhập vào từ bàn phím thì tăng biến **GiongNhu** lên 1 đơn vị với khởi tạo bằng 0. Giá trị của biến *GiongNhu* sau vòng lặp *for* cuối cùng chính là số lần xuất hiện của kí tự trong chuỗi và đưa ra màn hình giá trị của biến *GiongNhu*.

b) Kiểm tra xem chuỗi *S* nhập vào có phải là chuỗi đối xứng hay không: Chuỗi đối xứng là chuỗi mà các kí tự cách đều 2 đầu mút của chuỗi thì giống hệt nhau (tương tự dãy đối xứng ở phần mảng giá trị một chiều). Ban đầu coi chuỗi *S* có tính đối xứng (gán *k* = 1). Vòng lặp *for* kiểm tra từng kí tự của chuỗi với biến *i* chạy từ 0 đến *n* - 1 (*n* là số kí tự của chuỗi *S* và được tính qua hàm *strlen(S)*), nếu *S[i] ≠ S[n-i-1]* thì lập tức kết luận chuỗi trên không có tính đối xứng (gán *k*=0). In ra màn hình những nội dung tương ứng với kết quả trả về của hàm *DoiXung()*.

* Màn hình kết quả như sau:



```

C:\Users\Anh Hoi\Desktop\Untitled1.exe
Nhap vao mot xau ki tu S: Mai lai duoc nghi tin roi!
Nhap vao mot ki nao do tu ban phim: i
=> So lan xuat hien cua ki tu i trong xau S la: 5
=> Xau vua nhap khong la mot xau doi xung.

Nhap vao mot xau ki tu S: nguyenvanhoangnaohnavneyugn
Nhap vao mot ki nao do tu ban phim: g
=> So lan xuat hien cua ki tu g trong xau S la: 4
=> Xau vua nhap la mot xau doi xung.

```

Bài 99: Viết chương trình nhập vào hai chuỗi kí tự, ghép hai chuỗi này thành một chuỗi. Chuyển chuỗi kí tự đã ghép thành chữ hoa, in ra màn hình để kiểm tra kết quả chuyển đổi? (X)

```
#include <stdio.h>
```

```

#include <conio.h>
#include <ctype.h>
main()
{
    char
    XauKiTu1[1000],XauKiTu2[1000],XauKiTu3[3000],Dem1=0,Dem2=0,i,j;

    printf("\n Nhap vao xau ki tu thu nhât: ");
    gets(XauKiTu1);
    printf(" Nhap vao xau ki tu thu hai: ");
    gets(XauKiTu2);
    while (XauKiTu1[Dem1]!='\0')Dem1++;
    while (XauKiTu2[Dem2]!='\0')Dem2++;
    printf(" => Xau ki tu sau khi ghep la:\n\t");
    for(i=0;i<Dem1;i++)
        XauKiTu3[i]=XauKiTu1[i];
    for(j=0;j<Dem2;j++)
        XauKiTu3[Dem1+j]=XauKiTu2[j];
    XauKiTu3[Dem1+Dem2]='\0';
    printf("%s\n",XauKiTu3);
    i=0;
    while(XauKiTu3[i]!='\0')
        {XauKiTu3[i]=toupper(XauKiTu3[i]);i++;}
    printf(" => Chuyen doi xau da ghep thanh chu hoa:\n\t%s\n",XauKiTu3);
    getch();
    return main();
}

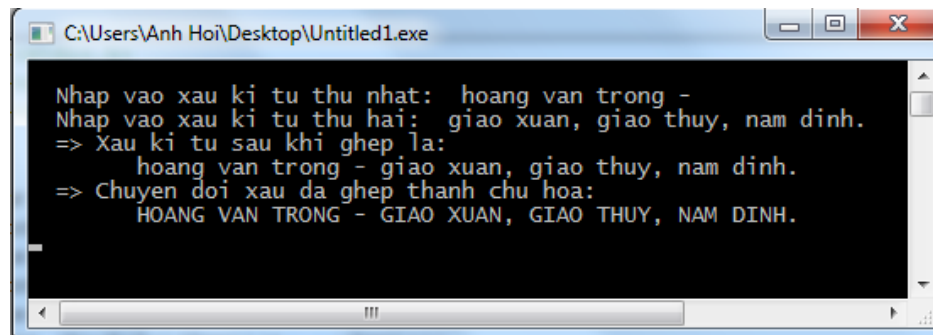
```

** Giải thích:*

a) Ghép hai xâu đã cho thành một xâu thứ 3. Trước tiên, đếm xem xâu 1 và xâu 2 có bao nhiêu kí tự, sử dụng biến **Dem1** và biến **Dem2**. Do đó, xâu thứ 3 sẽ có Dem1 + Dem 2 kí tự. Các phần tử của xâu thứ 3 được lấy lần lượt của xâu 1 và xâu 2, lấy hết xâu 1 được Dem1 phần tử, sau đó lấy sang xâu 2 được Dem1 + Dem2 phần tử. Sử dụng vòng lặp for để gán từng phần tử của xâu 1 và xâu 2 cho xâu 3.

b) Chuyển xâu kí tự thành chữ hoa: dùng vòng lặp `while` với `i` bắt đầu từ giá trị 0 (nếu dùng vòng `for` thì biến `i` chạy từ 0 đến `Dem1 + Dem2 - 1`) nếu `XauKiTu3[i]` khác kí tự kết thúc `'\0'` thì thực hiện chuyển đổi thành kí tự hoa với hàm `toupper()` được lấy trong thư viện `ctype.h`. Cuối cùng, in xâu đã chuyển đổi chữ hoa ra màn hình.

* Màn hình kết quả như sau:



* Ta có thể ghép xâu 2 vào xâu 1 nhờ hàm `strcat(xâu 1, xâu 2)`. Hàm này sẽ thêm vào sau xâu 1 các phần tử của xâu 2 (với điều kiện bộ nhớ dành cho xâu 1 đủ để chứa các phần tử của xâu 2). Sử dụng cách này sẽ làm cho số câu lệnh ít hơn rất nhiều. Mã cho trường hợp này như sau:

```
#include <stdio.h>
#include <conio.h>
#include <ctype.h>
main()
{
    char XauKiTu1[1000],XauKiTu2[1000],i=0;
    printf("\n Nhap vao xau ki tu thu nhât: ");
    gets(XauKiTu1);
    printf(" Nhap vao xau ki tu thu hai: ");
    gets(XauKiTu2);
    strcat(XauKiTu1,XauKiTu2);
    printf(" => Xau ki tu sau khi ghép là:\n\t%s\n",XauKiTu1);
    while(XauKiTu1[i]!='\0')
        {XauKiTu1[i]=toupper(XauKiTu1[i]);i++;}
    printf(" => Chuyen doi xau đã ghép thành chu hoa:\n\t%s\n",XauKiTu1);
    getch();
    return main();
}
```

}

* Màn hình kết quả như sau:

```

C:\Users\Anh Hoi\Desktop\Untitled1.exe

Nhap vao xau ki tu thu nhât: Chua Ngoa Van -
Nhap vao xau ki tu thu hai: xa Binh Khe, huyen Dong Trieu, tinh Quang Ninh.
=> Xau ki tu sau khi ghép là:
    Chua Ngoa Van - xa Binh Khe, huyen Dong Trieu, tinh Quang Ninh.
=> Chuyen doi xau da ghép thành chu hoa:
    CHUA NGOA VAN - XA BINH KHE, HUYEN DONG TRIEU, TINH QUANG NINH.

Nhap vao xau ki tu thu nhât: Trinh phuc vao thu 6
Nhap vao xau ki tu thu hai: ngay 09/11/2012.
=> Xau ki tu sau khi ghép là:
    Trinh phuc vao thu 6 ngay 09/11/2012.
=> Chuyen doi xau da ghép thành chu hoa:
    TRINH PHUC VAO THU 6 NGÀY 09/11/2012.

```

Bài 100: Nhập vào một xâu kí tự:

- Xét xem trong xâu có k kí tự kề nhau và giống nhau hay không?
- Thực hiện loại bỏ tất cả các kí tự kề nhau mà giống nhau, chỉ để lại một?

```

#include <stdio.h>
#include <conio.h>
#include <string.h>
main()
{
    char Xau[100];
    int k,h=0,i,j,Dem;
    printf("\n Nhập vào một xâu kí tự: ");
    gets(Xau);
    printf(" Nhập số k để xem có k kí tự kề nhau và giống nhau hay không:
");
    nhaplai:scanf("%d",&k);
    if(k>1)
    {
        for(i=0;i<strlen(Xau)-1;i++)
            {Dem=1;

```



```

        for(j=i+1;j<strlen(Xau);j++)
        {
            if(Xau[i]==Xau[j])Dem++;
            else break;
        }
        if(Dem>=k)h=1;
    }
    if(h==1)printf(" => Xau co %d ki tu ke nhau va giống nhau.\n",k);
    else printf(" => Xau không có %d ki tu ke nhau va giống nhau.\n",k);
}
else {printf(" Nhập lại k cho anh mau: ");goto nhaplai;}
for(i=0;i<strlen(Xau)-1;i++)
    if(Xau[i]==Xau[i+1])
    { strcpy(Xau+i,Xau+i+1);
      i--;
    }
printf(" => Xau ki tu sau khi chỉnh sửa là: %s\n",Xau);
fflush(stdin);
getch();
return main();
}

```

** Giải thích:*

- Thực hiện nhập chuỗi ký tự bất kỳ và số k từ bàn phím. Nếu $k > 1$ thì mới làm tiếp bước kiểm tra trong chuỗi có k ký tự kế nhau và giống nhau hay không. Ngược lại, nếu $k \leq 1$ thì tiến hành nhập lại giá trị của k bằng lệnh nhảy vô điều kiện **goto** tới chỗ gán nhãn **nhaplai**. Mọi người tìm hiểu thêm lệnh nhảy goto ở giáo trình **Ngôn ngữ lập trình C – Quách Tuấn Ngọc**, trang 151; sử dụng lệnh này có rất nhiều tiện ích hay.

a) Thuật toán xét xem chuỗi có k ký tự kế nhau và giống nhau hay không: Trước tiên giả sử rằng chuỗi đã cho không có k ký tự kế nhau và giống nhau (khởi tạo $h = 0$). Thực hiện duyệt từng ký tự của chuỗi; với mỗi ký tự thứ i ta kiểm tra xem từ ký tự thứ i+1 đến ký tự cuối cùng có bao nhiêu ký tự kế ngay sau ký tự thứ i và giống với nó rồi trả kết quả về cho biến **Dem**, trong trường hợp mà thấy ký tự thứ i và ký tự thứ j khác nhau (j chạy từ i+1 đến $\text{strlen}(\text{Xau}) - 1$) thì lập tức thoát khỏi vòng lặp **for(j=i+1;j<strlen(Xau);j++)** bằng lệnh nhảy vô điều kiện **break**. Khi ra khỏi vòng **for(j=i+1;j<strlen(Xau);j++)** thì thực hiện phép so sánh Dem với k, nếu $\text{Dem} \geq k$ thì có nghĩa là trong chuỗi đã xuất hiện nhiều hơn hoặc bằng k ký tự kế nhau và giống nhau (gán $h = 1$). Tiếp

theo, dựa vào giá trị của biến h để kết luận chuỗi đã cho có k ký tự kề nhau và giống nhau hay không.

b) Nếu thấy có nhiều ký tự kề nhau và giống nhau thì chỉ giữ lại một: Dùng vòng for duyệt từng ký tự của chuỗi từ ký tự đầu tiên đến ký tự sát với ký tự cuối cùng (trong trường hợp này thì chỉ cần kiểm tra từ $Xau[0]$ đến $Xau[n-2]$ với n là tổng số ký tự). Nếu ký tự thứ i bằng ký tự thứ $i+1$ thì ghi đè địa chỉ của ký tự thứ $i+1$ lên địa chỉ của ký tự thứ i , lúc này ký tự thứ $i+1$ sẽ trở thành ký tự thứ i của chuỗi mới và tổng số ký tự của chuỗi mới bị giảm đi 1 vì vậy ta phải giảm i đi 1 đơn vị để tiếp tục duyệt các ký tự khác của vòng for thì mới cho kết quả chính xác.

Dùng hàm `strcpy (Xau+i, Xau+i+1)` để ghi đè địa chỉ của ký tự thứ $i+1$ lên địa chỉ của ký tự thứ i .

* Màn hình kết quả như sau:

```

G:\TRONG\Trong.exe
Nhap vao mot xau ki tu: Diii hhooccccccc lllla di ttuuuuu,
Nhap so k de xem co k ki tu ke nhau va giong nhau hay khong: 7
=> Xau co 7 ki tu ke nhau va giong nhau.
=> Xau ki tu sau khi chinh sua la: Di hoc la di tu,

Nhap vao mot xau ki tu: ngoooi hoooc la nnngoi tu!
Nhap so k de xem co k ki tu ke nhau va giong nhau hay khong: 10
=> Xau co 10 ki tu ke nhau va giong nhau.
=> Xau ki tu sau khi chinh sua la: ngoi hoc la ngoi tu!

Nhap vao mot xau ki tu: Tai nang co han nhung khoon nann vo cung.
Nhap so k de xem co k ki tu ke nhau va giong nhau hay khong: 7
=> Xau khong co 7 ki tu ke nhau va giong nhau.
=> Xau ki tu sau khi chinh sua la: Tai nang co han nhung khon nan vo cung.
  
```

Bài 101: **Xâu ký tự chuẩn** là chuỗi không có dấu cách ở đầu và cuối câu, đồng thời không có hai dấu cách liên tiếp nhau trong chuỗi. Viết chương trình:

- Nhập một chuỗi ký tự từ bàn phím.
- Đếm xem có bao nhiêu dấu cách trong chuỗi.
- Nếu xuất hiện 2 dấu cách liên tiếp nhau thì bỏ đi một. (X)

```

#include <stdio.h>
#include <conio.h>
#include <string.h>
main()
{
    char Xau[1000];
    int i=0,SoKyTuTrang=0;
  
```

```

printf("\n Nhập vào một chuỗi ký tự: ");
gets(Xau);
while(Xau[i]!='\0')
    {if(Xau[i]==' ')SoKyTuTrang++;i++;}
printf(" => Số ký tự trong chuỗi là: %d\n",SoKyTuTrang);
i=0;
while(Xau[i]!='\0')
    {
        if(Xau[i]==' ' && Xau[i]==Xau[i+1]){strcpy(Xau+i,Xau+i+1);i--;}
        i++;
    }
printf(" => Chuỗi đã chỉnh sửa dấu cách là: %s\n",Xau);
getch();
return main();
}

```

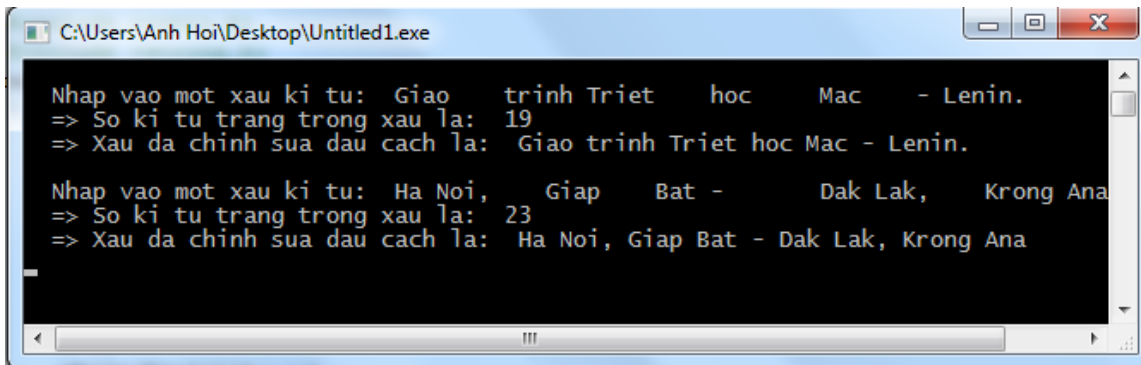
** Giải thích:*

- Sử dụng biến **SoKyTuTrang** để đếm số ký tự trắng với giá trị khởi tạo bằng 0. Dùng vòng lặp *while* duyệt qua từng ký tự của chuỗi, nếu ký tự nào bằng ký tự trắng thì tăng biến *SoKyTuTrang* lên 1 đơn vị.

- Bỏ đi một dấu cách nếu có 2 dấu cách liên tiếp nhau: vòng lặp *while* chạy từ ký tự đầu tiên cho đến khi gặp ký tự '\0' thì dừng lại. Nếu ký tự thứ *i* bằng ký tự thứ *i + 1* và bằng ký tự trắng thì copy địa chỉ của ký tự thứ *i + 1* lên địa chỉ của ký tự thứ *i* (lúc này chuỗi sẽ giảm đi một ký tự) bằng hàm *strcpy()* và khi copy xong như vậy thì ký tự thứ *i + 1* sẽ trở thành ký tự thứ *i* cho vòng lặp sau, chính vì vậy phải giảm giá trị của *i* xuống 1 đơn vị. Cuối cùng in ra chuỗi đã chỉnh sửa dấu cách.

* Lưu ý: Nếu dãy ban đầu nhập vào có chứa nhiều dấu cách ở đầu câu hoặc cuối câu thì cuối cùng vẫn còn chứa 1 dấu cách ở đầu hoặc cuối câu. Vì đề bài không yêu cầu nên mình không viết lệnh xóa dấu cách ở đầu câu và cuối câu.

** Màn hình kết quả như sau:*



Bài 102: Viết chương trình nhập một chuỗi ký tự bất kỳ từ bàn phím. Chuẩn hóa chuỗi ký tự:

- Nếu có dấu cách ở đầu câu và cuối câu thì bỏ đi.
- Nếu trong chuỗi ký tự có hai dấu cách liên tiếp nhau thì bỏ đi một dấu cách.
- Đếm xem trong chuỗi ký tự vừa chuẩn hóa có bao nhiêu từ. (X)

```
#include <stdio.h>
#include <conio.h>
main()
{
    char Xau[1000];
    int i=0,SoDauCach=0,SoKiTu;
    printf("\n Nhập vào một chuỗi ký tự: ");
    gets(Xau);
    while(Xau[i]!=' ')
        strcpy(Xau+i,Xau+i+1);
    SoKiTu=strlen(Xau);
    i=SoKiTu-1;
    while(Xau[i]!=' ')
        {Xau[i]='\0';i--;}
    i=0;
    while(Xau[i]!='\0')
    {
        if(Xau[i]==' '&&Xau[i]==Xau[i+1]){strcpy(Xau+i,Xau+i+1);i--;}
        i++;
    }
}
```

```

    }
    printf(" => Xau da chuan hoa la: %s\n",Xau);
    i=0;
    while(Xau[i]!='\0'){if(Xau[i]==' ')SoDauCach++;i++;}
    printf(" => So tu trong xau la: %d\n",SoDauCach+1);
    getch();
    return main();
}

```

* Giải thích:

a) Bỏ đi các dấu cách ở đầu câu: Vòng lặp while duyệt phần tử đầu tiên mà nếu phần tử này bằng kí tự trắng thì copy đề địa chỉ của phần tử thứ hai lên phần tử thứ nhất và phần tử thứ hai này lại trở thành phần tử thứ nhất trong vòng lặp sau. Cứ thế cho tới khi gặp kí tự khác kí tự trắng thì dừng lại.

b) Bỏ đi các dấu cách ở cuối câu: Sau khi bỏ đi các kí tự trắng ở đầu câu thì xâu ban đầu trở thành xâu mới, số kí tự của xâu mới này được gán cho biến **SoKiTu** và gán $i = \text{SoKiTu} - 1$. Vòng lặp while duyệt từ phần tử cuối cùng của xâu mới, nếu phần tử này bằng kí tự trắng thì gán nó bằng kí tự kết thúc '\0', sau đó giảm biến i xuống 1 đơn vị để đi vào vòng lặp sau.

c) Sau khi bỏ đi các kí tự trắng ở cuối câu thì xâu lại trở thành xâu mới. Gán lại giá trị khởi tạo cho $i = 0$ và thực hiện bỏ đi một dấu cách nếu có hai dấu cách đứng cạnh nhau trong xâu. Nếu kí tự thứ i bằng kí tự thứ $i + 1$ và bằng kí tự trắng thì copy đề địa chỉ của phần tử thứ $i + 1$ lên địa chỉ của phần tử thứ i . Hết bước này thì xâu ban đầu nhập vào đã được chuẩn hóa và ta chỉ cần viết lệnh in ra màn hình là ok.

d) Số từ trong xâu: Muốn biết trong xâu có bao nhiêu từ thì chỉ cần lấy số kí tự trắng cộng thêm 1 đơn vị, số kí tự trắng được xác định nhờ biến **SoDauCach**. Nhưng số kí tự trắng ở đây phải được đếm sau khi xâu đã được chuẩn hóa. Nếu không số từ hiện ra sẽ không chính xác.

* Màn hình kết quả như sau:

```

C:\Users\Anh Hoi\Desktop\Untitled1.exe
Nhap vao mot xau ki tu: "Bong ma ben cua so" cua Nguyen Ngoc Ngan.
=> Xau da chuan hoa la: "Bong ma ben cua so" cua Nguyen Ngoc Ngan.
=> So tu trong xau la: 9

Nhap vao mot xau ki tu: Kinh te hoc vi mo.
=> Xau da chuan hoa la: Kinh te hoc vi mo.
=> So tu trong xau la: 5

Nhap vao mot xau ki tu: Dai hoc tong hop Ha Noi.
=> Xau da chuan hoa la: Dai hoc tong hop Ha Noi.
=> So tu trong xau la: 6

```

Bài 103: Viết chương trình nhập vào từ bàn phím một xâu kí tự. Tính xem có bao nhiêu loại chữ cái có trong xâu, mỗi loại chữ cái xuất hiện bao nhiêu lần, in kết quả thông báo ra màn hình? (X)

```
#include <stdio.h>
#include <conio.h>
main()
{
    char Xau[1000];
    int i,j,k,n,SoLoai,Dem=0,Giong=0;
    printf("\n Nhập vào mot xau ki tu: ");
    gets(Xau);
    SoLoai=strlen(Xau);
    n=strlen(Xau);
    for(i=0;i<n;i++)
        for(j=i-1;j>=0;j--)
            if(Xau[i]==Xau[j]){ SoLoai--;break;}
    for(i=0;i<n;i++)
        if(Xau[i]==' '){ SoLoai--;break;}
    printf(" => So loai chu cai co trong xau la: %d\n",SoLoai);
    printf(" => So lan xuat hien cua cac chu cai co trong xau:\n");
    for(i=0;i<n;i++)
    {
        for(j=i-1;j>=0;j--)
            if(Xau[i]==Xau[j])Giong=1;
        if(Giong==0&&Xau[i]!=' ')
        {
            for(k=0;k<n;k++)
                if(Xau[i]==Xau[k])Dem++;
            printf("\tChu cai '%c' xuat hien %d lan.\n",Xau[i],Dem);
        }
        Dem=0;Giong=0;
    }
}
```

```

    getch();
    return main();
}

```

* Giải thích:

a) Tính xem có bao nhiêu loại chữ cái xuất hiện trong xâu: được xác định bằng biến **SoLoai**. Ban đầu gán biến SoLoai bằng tổng số kí tự có trong xâu (kể cả kí tự trắng và kí tự đặc biệt như: ; , \ “ ” ? - _ ! @ # % ^ & *) và tổng số kí tự này được xác định bằng hàm **strlen(Xau)**, sau đó duyệt từng kí tự từ kí tự đầu tiên đến kí tự cuối cùng. Nếu kí tự nào trùng với một trong các kí tự đứng trước nó thì giảm SoLoai xuống 1 đơn vị. Cuối cùng, nếu thấy trong xâu có kí tự trắng thì lại giảm SoLoai xuống tiếp 1 đơn vị (giả sử coi như các kí tự đặc biệt cũng là các chữ cái).

b) Đếm số lần xuất hiện của các chữ cái: Duyệt từng kí tự của xâu bằng vòng lặp for. Nếu kí tự nào không giống với ít nhất một kí tự trước nó và kí tự đó phải khác kí tự trắng thì thực hiện đếm các kí tự giống với nó trong xâu, khởi tạo biến **Dem** bằng 0 và cũng duyệt qua tất cả kí tự của xâu nếu thấy trùng nhau thì tăng biến Dem lên 1 đơn vị. Sau cùng, in ra chữ cái và số lần xuất hiện tương ứng của chữ cái đó. Mọi người lưu ý là có sự kết hợp khá phức tạp giữa các vòng for lồng nhau ở đoạn chương trình trên.

c) Nếu không cho các kí tự đặc biệt là chữ cái thì phải viết thêm 1 số câu lệnh nữa để không đếm các kí tự đặc biệt đó. Ở bài này mình chỉ cho kí tự trắng không phải chữ cái mà thôi! Một điều nữa là có sự phân biệt chữ thường và chữ hoa, chương trình sẽ đếm chữ hoa thành một loại chữ cái riêng và chữ thường thành một loại chữ cái riêng.

* Màn hình kết quả như sau:

```

C:\Users\DAT\Desktop\Untitled1.exe
Nhap vao mot xau ki tu: duong loi cach mang cua dang cong san viet nam
=> So loai chu cai co trong xau la: 15
=> So lan xuất hiện của các chu cai có trong xau:
Chu cai 'd' xuất hiện 2 lan.
Chu cai 'u' xuất hiện 2 lan.
Chu cai 'o' xuất hiện 3 lan.
Chu cai 'n' xuất hiện 6 lan.
Chu cai 'g' xuất hiện 4 lan.
Chu cai 'l' xuất hiện 1 lan.
Chu cai 'i' xuất hiện 2 lan.
Chu cai 'c' xuất hiện 4 lan.
Chu cai 'a' xuất hiện 6 lan.
Chu cai 'h' xuất hiện 1 lan.
Chu cai 'm' xuất hiện 2 lan.
Chu cai 's' xuất hiện 1 lan.
Chu cai 'v' xuất hiện 1 lan.
Chu cai 'e' xuất hiện 1 lan.
Chu cai 't' xuất hiện 1 lan.

```

Bài 104: Viết chương trình nhập vào từ bàn phím một xâu kí tự bất kì:

- Chuẩn hóa xâu kí tự vừa nhập.
- Chuyển đổi từ thứ k thành chữ hoa, in ra màn hình chuỗi kí tự để xem kết quả.
- Tách kí tự vừa chuyển thành chữ hoa đó ra khỏi xâu kí tự, in chuỗi kí tự này ra màn hình để xem kết quả. (X)

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
#include <ctype.h>
main()
{
    char Xau[1000];
    int i=0,j=0,k,z,SoKiTu,SoDauCach=0,DoDaiTu=0,Dem=1;
    printf("\n Nhap vao mot xau ki tu: ");
    gets(Xau);
    while(Xau[i]!=' ')
        strcpy(Xau+i,Xau+i+1);
    SoKiTu=strlen(Xau);
    i=SoKiTu-1;
    while(Xau[i]!=' ')
        {Xau[i]='\0';i--;}
    i=0;
    while(Xau[i]!='\0')
    {
        if(Xau[i]==' '&&Xau[i]==Xau[i+1]){strcpy(Xau+i,Xau+i+1);i--;}
        i++;
    }
    printf(" => Xau da chuan hoa la: %s\n",Xau);
    i=0;
    while(Xau[i]!='\0'){if(Xau[i]==' ')SoDauCach++;i++;}
    printf(" Nhap vao so k de chuyen tu thu k thanh chu hoa: ");
```



```

nhaplaidiku:scanf("%d",&k);
if(k==1)
{
    z=0;
    for(i=0;i<strlen(Xau);i++)
    {
        if(Xau[i]!=' ')DoDaiTu++;
        if(Xau[i]==' ')break;
    }
    for(i=0;i<strlen(Xau);i++)
    {
        if(Xau[i]!=' ')Xau[i]=toupper(Xau[i]);
        if(Xau[i]==' ')break;
    }
    printf(" => Chuoi ki tu sau khi chuyen tu thu %d thanh chu hoa
la:\n\t%s\n",k,Xau);
}
else if(k>1&&k<=SoDauCach+1)
{
    for(i=0;i<strlen(Xau);i++)
    {
        if(Xau[i]==' ')Dem++;
        if(Dem==k){j=i+1;break;}
    }
    z=j;
    for(i=j;i<strlen(Xau);i++)
    {
        if(Xau[i]!=' '){Xau[i]=toupper(Xau[i]);DoDaiTu++;}
        if(Xau[i]==' ')break;
    }
    printf(" => Chuoi ki tu sau khi chuyen tu thu %d thanh chu hoa
la:\n\t%s\n",k,Xau);
}

```

```

else
    {printf(" Nhập lại k đi may: ");goto nhaplaidiku;}
strcpy(Xau+z,Xau+z+DoDaiTu);
printf(" => Chuoi ki tu sau khi da tach chu hoa ra la:\n\t%s\n",Xau);
fflush(stdin);
getch();
return main();
}

```

** Giải thích:*

Bài này mình viết code để xử lý tất cả các trường hợp đặc biệt có thể xảy ra nên phải sử dụng nhiều biến do đó nếu nhìn qua thì khá rắc rối, mọi người chịu khó nhé!

a) Phần chuẩn hóa xâu kí tự nhập vào: tức là xóa hết kí tự trắng ở đầu xâu và cuối xâu, nếu trong xâu mà có 2 kí tự trắng liên tiếp nhau thì bỏ đi một kí tự trắng. Thuật toán này đã có trong bài trước đó về chuẩn hóa xâu kí tự. Để xóa bỏ kí tự trắng ở đầu xâu thì duyệt các phần tử từ phần tử đầu tiên với điều kiện kí tự đó phải bằng kí tự trắng thì mới cho thực hiện vòng lặp, sau đó copy đề địa chỉ của phần tử thứ $i + 1$ lên địa chỉ của phần tử thứ i và xâu đã bị giảm đi một kí tự, cứ như thế cho tới khi gặp kí tự khác kí tự trắng thì thôi. Còn xóa kí tự trắng ở cuối xâu thì lại duyệt ngược lại bắt đầu từ kí tự cuối cùng, nếu nó bằng kí tự trắng thì gán nó bằng kí tự kết thúc '\0'. Xóa 1 kí tự trắng nếu thấy xuất hiện 2 kí tự trắng đứng cạnh nhau thì làm tương tự, nếu thấy phần tử thứ i bằng phần tử thứ $i + 1$ và bằng kí tự trắng thì copy đề địa chỉ của kí tự thứ $i + 1$ lên địa chỉ của phần tử thứ i .

b) Chuyển đổi từ thứ k thành chữ hoa với k được nhập vào từ bàn phím: Xâu sau khi đã được chuẩn hóa thì số dấu cách của xâu mới này được xác định bằng biến **SoDauCach** và số từ trong xâu sẽ là **SoDauCach + 1**. Chính vì vậy, muốn nhập số k vào để chuyển đổi từ thứ k thành chữ hoa thì k phải nằm trong đoạn $[1, \text{SoDauCach} + 1]$, nếu k không thỏa mãn điều kiện này thì tiến hành nhập lại có sử dụng tới lệnh nhảy vô điều kiện **goto**.

Nếu $k = 1$ thì ta chỉ cần chuyển từ thứ nhất thành chữ hoa bắt đầu từ kí tự **Xau[0]** đến khi gặp kí tự trắng thì dừng lại và thoát khỏi vòng lặp **for**. Biến z để xác định vị trí bắt đầu của từ cần chuyển thành chữ hoa (trường hợp này $z = 0$), dùng biến này để thuận tiện cho việc cắt từ ở những câu lệnh sau đó, biến **DoDaiTu** để xác định độ dài của từ cần chuyển thành chữ hoa. Tiếp theo, hiện ra xâu kí tự đã chuyển từ thứ k thành chữ hoa.

Nếu $1 < k \leq \text{SoDauCach} + 1$ thì công việc lại phức tạp hơn, ta phải xác định được vị trí bắt đầu của từ thứ k . Biến **Dem** được dùng để đếm số dấu cách cho tới khi **Dem = k** (với **Dem** được khởi tạo bằng 1) thì vị trí bắt đầu của từ thứ k chính là $j = i + 1$ và thoát khỏi vòng lặp **for**. Khi đã xác định được vị trí bắt đầu của từ thứ k rồi thì dùng hàm **toupper(Xau[i])** để chuyển thành chữ hoa từ kí tự $i = j$ đến khi gặp kí tự trắng thì thoát khỏi vòng lặp. Trong trường hợp này của k cũng sử dụng biến z để xác định vị trí bắt đầu của từ thứ k (chính là $z = j$) và biến **DoDaiTu** để xác định độ dài của từ thứ k . Tiếp theo, hiện ra xâu kí tự đã chuyển từ thứ k thành chữ hoa.

c) Tách từ vừa in thành chữ hoa ra khỏi chuỗi: Ở bước trên ta đã xác định được vị trí bắt đầu và độ dài của từ thứ k thì đến phần này ta chỉ cần copy đề địa chỉ của phần tử thứ $z + \text{DoDaiTu}$ lên địa chỉ của phần tử thứ z thì lập tức xâu sẽ mất một từ thứ k . Chỗ mà có từ bị cắt này sẽ xuất

hiện 2 kí tự trắng (mình cố tình để như thế cho dễ nhìn thấy chỗ bị cắt), còn nếu muốn chỉ xuất hiện 1 kí tự trắng thì thay vì copy đề địa chỉ của phần tử thứ $z + DoDaiTu$ thì ta copy đề địa chỉ của phần tử $z + DoDaiTu + 1$ lên địa chỉ của phần tử thứ z .

* Màn hình kết quả như sau:

```

C:\Users\Anh Ho\Desktop\Untitled1.exe
Nhap vao mot xau ki tu:   Ngang mat   len troi,   han   doi vo doi.
=> Xau da chuan hoa la: Ngang mat len troi, han doi vo doi.
Nhap vao so k de chuyen tu thu k thanh chu hoa: 5
=> Chuoi ki tu sau khi chuyen tu thu 5 thanh chu hoa la:
    Ngang mat len troi, HAN doi vo doi.
=> Chuoi ki tu sau khi da tach chu hoa ra la:
    Ngang mat len troi, doi vo doi.

Nhap vao mot xau ki tu: Cui mat           xuong dat, vo doi           la ta!
=> Xau da chuan hoa la: Cui mat xuong dat, vo doi la ta!
Nhap vao so k de chuyen tu thu k thanh chu hoa: 4
=> Chuoi ki tu sau khi chuyen tu thu 4 thanh chu hoa la:
    Cui mat xuong DAT, vo doi la ta!
=> Chuoi ki tu sau khi da tach chu hoa ra la:
    Cui mat xuong vo doi la ta!
    
```

Bài 105: Viết chương trình nhập vào từ bàn phím một xâu kí tự bất kì:

- Chuẩn hóa xâu kí tự vừa nhập và chuyển kí tự đầu tiên của xâu thành chữ hoa.
- Thay tất cả các kí tự của từ cuối cùng trong xâu thành các dấu *. In kết quả.
- In mỗi từ trong xâu ra một dòng. (X)

```

#include <stdio.h>
#include <conio.h>
#include <string.h>
#include <ctype.h>
main()
{
    char Xau[1000];
    int i=0, SoKiTu;
    printf("\n Nhap vao mot xau ki tu: ");
    gets(Xau);
    while(Xau[i]!=' ')
        strcpy(Xau+i, Xau+i+1);
    
```

```

SoKiTu=strlen(Xau);
i=SoKiTu-1;
while(Xau[i]==' ')
    {Xau[i]='\0';i--;}
i=0;
while(Xau[i]!='\0')
    {
        if(Xau[i]==' '&&Xau[i]==Xau[i+1]){strcpy(Xau+i,Xau+i+1);i--;}
        i++;
    }
printf(" => Xau da chuan hoa la: %s\n",Xau);
Xau[0]=toupper(Xau[0]);
printf(" => Xau ki tu sau khi bien doi ki tu dau tien thanh ki tu hoa
la:\n\t%s\n",Xau);
for(i=strlen(Xau)-1;i>=0;i--)
    {
        if(Xau[i]!=' ')Xau[i]='*';
        else break;
    }
printf(" => Xau ki tu sau khi bien doi tu cuoi cung la:\n\t%s\n",Xau);
printf(" => In moi tu trong xau ra mot dong:\n\t");
for(i=0;i<strlen(Xau);i++)
    {
        if(Xau[i]!=' ')printf("%c",Xau[i]);
        else printf("\n\t");
    }
printf("\n");
getch();
return main();
}

```

* Giải thích:

a) Chuẩn hóa xâu kí tự cũng giống như các bài ở trên, cũng phải làm các công việc để xóa bỏ kí tự trắng ở đầu và cuối xâu, nếu trong xâu có 2 kí tự trắng cạnh nhau thì bỏ đi một kí tự trắng. Sau khi chuẩn hóa xong thì vừa biến đổi kí tự đầu tiên thành kí tự hoa vừa gán kí tự hoa đó cho kí tự đầu tiên.

b) Thay tất cả các kí tự của từ cuối cùng trong xâu thành kí tự * thì dùng vòng lặp for duyệt từ kí tự cuối cùng cho tới khi gặp kí tự trắng thì thoát vòng lặp. Duyệt đến đâu thì gán kí tự * cho phần tử thứ i của xâu.

c) Sau khi thay tất cả các kí tự của từ cuối cùng thành kí tự * thì tiến hành in ra các từ trong xâu, in mỗi từ trên một dòng. Vòng lặp for duyệt từ phần tử đầu tiên đến phần tử cuối cùng. Nếu phần tử thứ i khác kí tự trắng thì cho in ra màn hình, còn nếu phần tử thứ i bằng kí tự trắng thì thực hiện lệnh xuống dòng và lùi vào một khoảng cách bằng tab.

* Màn hình kết quả như sau:

```

C:\Users\DAT\Desktop\Untitled1.exe
Nhập vào một xâu kí tự:   bien      hoc menh mong, quay      dau la bo
=> Xâu đã chuẩn hóa là:  bien hoc menh mong, quay dau la bo
=> Xâu kí tự sau khi biến đổi kí tự đầu tiên thành kí tự hoa là:
    Bien hoc menh mong, quay dau la bo
=> Xâu kí tự sau khi biến đổi từ cuối cùng là:
    Bien hoc menh mong, quay dau la **
=> In mỗi từ trong xâu ra một dòng:
    Bien
    hoc
    menh
    mong,
    quay
    dau
    la
    **
  
```

Bài 106: Viết chương trình nhập nhiều tên người vào từ bàn phím. Hãy sắp xếp lại theo thứ tự alphabet (thứ tự abc...z) và in ra màn hình kết quả đã sắp xếp theo thứ tự đó?

```

#include <stdio.h>
#include <conio.h>
#include <string.h>
void SapXep(char Xau[][100],int n);
main()
{
    char Xau[100][100],n,i;
    printf("\n Nhập vào số lượng tên người: ");
    scanf("%d",&n);
  
```

```

printf(" Nhập vào từng tên người:\n");
for(i=1;i<=n;i++)
    {printf("\tTên thứ %d là: ",i);scanf("%s",Xau[i]);}
SapXep(Xau,n);
printf(" => Tên người đã sắp xếp theo thứ tự alphabet là:\n");
for(i=1;i<=n;i++)
    printf("\t\t%s\n",Xau[i]);
getch();
return main();
}
void SapXep(char Xau[][100],int n)
{
    int i,j;
    char XauTrungGian[100];
    for(i=1;i<n;i++)
        for(j=i+1;j<=n;j++)
            if(strcmp(Xau[i],Xau[j])>0)
                {
                    strcpy(XauTrungGian,Xau[i]);
                    strcpy(Xau[i],Xau[j]);
                    strcpy(Xau[j],XauTrungGian);
                }
}

```

** Giải thích:*

a) Khai báo một mảng các chuỗi ký tự và coi mỗi chuỗi ký tự là một phần tử của mảng, cung cấp sẵn một bộ nhớ chứa tối đa 100 chuỗi và mỗi chuỗi chứa tối đa 100 ký tự. Sau đó nhập vào số lượng chuỗi cần thiết và nhập vào từng chuỗi.

b) Định nghĩa thêm hàm **SapXep()** để sắp xếp các chuỗi theo thứ tự alphabet (thứ tự abc), tiếp theo truyền mảng các chuỗi vào cho hàm. Hàm **strcmp()** là hàm so sánh 2 chuỗi dựa theo chỉ số của từng ký tự trong bảng mã ASCII. Nếu chuỗi thứ nhất lớn hơn chuỗi thứ hai (đứng sau chuỗi 2 theo thứ tự alphabet) thì thực hiện đổi vị trí hai chuỗi này cho nhau bằng hàm **strcpy()** và có sử dụng **XauTrungGian**. Qua hàm **SapXep()** thì không cần trả về giá trị cho hàm, sau đó in ra các chuỗi theo thứ tự đã sắp xếp.

** Màn hình kết quả như sau:*

```

C:\Users\DAT\Desktop\Untitled1.exe
Nhap vao so luong ten nguoi: 9
Nhap vao tung ten nguoi:
Ten thu 1 la: Loan
Ten thu 2 la: Trang
Ten thu 3 la: Dung
Ten thu 4 la: Anh
Ten thu 5 la: Xuyen
Ten thu 6 la: Trong
Ten thu 7 la: Thai
Ten thu 8 la: Dung
Ten thu 9 la: Phong
=> Ten nguoi da sap xep theo thu tu alphabet la:
    Anh
    Dung
    Dung
    Loan
    Phong
    Thai
    Trang
    Trong
    Xuyen

```

Bài 107: Viết chương trình nhập vào một xâu có tối đa 80 kí tự. Thống kê xâu đó theo các thông tin sau:

- Số lượng kí tự trong xâu.
- Số lượng kí tự là nguyên âm và tỷ lệ kí tự nguyên âm tính theo %.
- Số từ trong xâu biết các từ cách nhau bởi một hoặc một nhóm kí tự trắng.
- Tiến hành chuẩn hóa xâu kí tự (loại bỏ các kí tự trắng thừa). Viết hoa kí tự đầu tiên?

```

#include <stdio.h>
#include <conio.h>
#include <string.h>
#include <ctype.h>
main()
{
    char Xau[81];
    int i, SoKiTu, KiTuNguyenAm=0, SoTu, ToanTrang=1;
    printf("\n Nhap vao mot xau ki tu:\n\t");
    gets(Xau);
    printf(" => So luong ki tu trong xau la: %d\n", strlen(Xau));

```

```

for(i=0;i<strlen(Xau);i++)
    if(Xau[i]=='a'||Xau[i]=='e'||Xau[i]=='i'||Xau[i]=='o'||Xau[i]=='u')
        KiTuNguyenAm++;
printf(" => So luong ki tu nguyen am la %d va chiem %0.2f (phan
tram)\n",KiTuNguyenAm,(float)KiTuNguyenAm/strlen(Xau)*100);
for(i=0;i<strlen(Xau)-1;i++)
    if(Xau[i]!=' ')ToanTrang=0;
if(ToanTrang==1)SoTu=0;
else if(ToanTrang==0&&Xau[0]!=' ')SoTu=1;
else SoTu=0;
for(i=0;i<strlen(Xau)-1;i++)
    if(Xau[i]==' '&&Xau[i+1]!=' ')SoTu++;
printf(" => So tu co trong xau la: %d\n",SoTu);
i=0;
while(Xau[i]!=' ')
    strcpy(Xau+i,Xau+i+1);
SoKiTu=strlen(Xau);
i=SoKiTu-1;
while(Xau[i]!=' ')
    {Xau[i]='\0';i--;}
i=0;
while(Xau[i]!='\0')
    {
        if(Xau[i]==' '&&Xau[i]==Xau[i+1]){strcpy(Xau+i,Xau+i+1);i--;}
        i++;
    }
Xau[0]=toupper(Xau[0]);
printf(" => Xau da chuan hoa va viet hoa ki tu dau tien la:\n\t%s\n",Xau);
getch();
return main();
}

```

* Giải thích:

a) Nhập vào xâu có tối đa 80 kí tự thì phải dành ra 81 ô nhớ vì còn một ô nhớ để chứa kí tự kết thúc '\0'.

Tổng số lượng kí tự có trong xâu được tính bằng hàm **strlen** (**strlen** = **string length** = độ dài xâu kí tự). Số lượng kí tự là nguyên âm thì sử dụng biến **KiTuNguyenAm** để đếm với giá trị khởi tạo bằng 0, dùng vòng lặp **for** với **i** chạy từ 0 đến $n - 1$ (n là tổng số kí tự của xâu). Nếu kí tự thứ **i** bằng 1 trong 5 kí tự nguyên âm (a, e, i, o, u) thì tăng biến **KiTuNguyenAm** lên một đơn vị.

Muốn tính tỷ lệ % của các kí tự nguyên âm thì lấy giá trị của biến **KiTuNguyenAm** chia cho tổng số kí tự của xâu rồi nhân với 100. Phải ép kiểu số thực (float) cho giá trị của tỷ lệ. Dấu **%0.2f** có ý nghĩa là lấy đến 2 chữ số sau dấu phẩy ngăn cách phần nguyên và phần thập phân.

b) Tính số từ có trong xâu (trong trường hợp xâu chưa chuẩn hóa): sử dụng biến **SoTu** nhưng tùy theo từng trường hợp mà gán giá trị ban đầu cho biến **SoTu**:

+ Nếu xâu nhập vào mà toàn kí tự trắng thì gán **SoTu** = 0.

+ Nếu xâu nhập vào mà chứa kí tự khác trắng và kí tự đầu tiên khác trắng thì **SoTu** = 1.

+ Nếu xâu nhập vào mà chứa kí tự khác trắng và kí tự đầu tiên bằng kí tự trắng thì gán **SoTu** = 0.

Khi đã biết giá trị khởi tạo ban đầu của biến **SoTu** thì thực hiện vòng lặp để duyệt qua các kí tự. Nếu kí tự thứ **i** bằng kí tự trắng và kí tự thứ **i + 1** khác kí tự trắng thì tăng biến **SoTu** lên 1 đơn vị. Cuối cùng in ra màn hình số từ có trong xâu.

c) Tiến hành chuẩn hóa xâu kí tự: tức là cắt các kí tự trắng xuất hiện ở đầu xâu và cuối xâu, nếu bên trong xâu mà có 2 kí tự trắng cạnh nhau thì bỏ đi 1 kí tự trắng (phần này mọi người xem ở các bài tập trên, nội dung chuẩn hóa xâu kí tự nhé!). Nếu muốn kí tự đầu ưu tiên thành kí tự hoa thì vừa dùng hàm chuyển đổi chữ hoa vừa gán kí tự đã chuyển đổi cho phần tử đầu tiên của xâu. Đưa ra màn hình chuỗi đã chuẩn hóa và viết hoa kí tự đầu tiên.

Ta có thể xác định số từ của xâu bằng cách sau khi chuẩn hóa xong ta mới đếm số kí tự trắng. Lúc này, số từ của xâu bằng số kí tự trắng cộng thêm 1.

* Màn hình kết quả như sau:

```

C:\Users\DAT\Desktop\Untitled1.exe
Nhập vào một xâu kí tự:
khong may do thay day ai!
=> Số lượng kí tự trong xâu là: 58
=> Số lượng kí tự nguyên âm là 7 và chiếm 12.07 (phần trăm)
=> Số từ có trong xâu là: 6
=> Xâu đã chuẩn hóa và viết hoa kí tự đầu tiên là:
    Không may do thay day ai!

Nhập vào một xâu kí tự:
truc xinh truc moc dau dinh, em xinh em hut thuoc lao cung xinh!
=> Số lượng kí tự trong xâu là: 82
=> Số lượng kí tự nguyên âm là 17 và chiếm 20.73 (phần trăm)
=> Số từ có trong xâu là: 14
=> Xâu đã chuẩn hóa và viết hoa kí tự đầu tiên là:
    Truc xinh truc moc dau dinh, em xinh em hut thuoc lao cung xinh!
    
```

Bài 108: Nhập vào một chuỗi S chỉ gồm các chữ cái thường từ a -> z. Đưa ra độ dài chuỗi S và kiểm tra xem chuỗi S có phải là chuỗi đối xứng hay không. Hãy cho biết số lần xuất hiện của mỗi chữ cái có trong chuỗi?

(Đề thi cuối kỳ lớp THCS 3, ĐHKHTN – ĐHQGHN, Thứ 4 ngày 19/12/2012, Kỳ I năm học 2012 – 2013, thời gian làm bài: 30 phút)

Bài 108*: Nhập vào một chuỗi S có n phần tử ($n \leq 100$) :

- Tính số lượng ký tự có trong chuỗi và đưa chuỗi S ra màn hình.
- Viết chuỗi S theo thứ tự ngược lại.
- Chuẩn hóa chuỗi S (Viết hoa ký tự đầu tiên và xóa ký tự trắng thừa).

(Đề thi cuối kỳ lớp THCS 3, ĐHKHTN – ĐHQGHN, Thứ 6 ngày 21/12/2012, Kỳ I năm học 2012 – 2013, thời gian làm bài: 40 phút)

CHƯƠNG VII. KIỂU DỮ LIỆU CẤU TRÚC

Bài 108: Nhập vào 2 phân số. In ra tổng, hiệu, tích, thương của 2 phân số đó?

```
#include <stdio.h>
#include <conio.h>
struct phanso{int TuSo;int MauSo;};
main()
{
    float Tong,Hieu,Tich,Thuong;
    struct phanso P1,P2;
    printf("\n Nhập vào tu so va mau so cua phan so 1 (mau khác 0): ");
    nhaplai1:scanf("%d%d",&P1.TuSo,&P1.MauSo);
    if(P1.MauSo==0){printf(" Nhập lại phan so 1: ");goto nhaplai1;}
    printf(" Nhập vào tu so va mau so cua phan so 2 (mau khác 0): ");
    nhaplai2:scanf("%d%d",&P2.TuSo,&P2.MauSo);
```

CÁCH GIẢI MỘT SỐ BÀI TẬP VỀ KIỂU CHUỖI

Nguồn: <http://viettruong92.blogspot.com> | Góc học tập

Để dowload vui lòng chọn Tập-->Tải xuống dưới dạng-->chọn định dạng tài

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 chứ 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 strInsert 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.**

```
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 khoang trang dau chuoì
    int n=strlen(s);
    if(s[n-1]==' ')
        s[n-1]='\0'; //xoa khoang trang cuoi chuoì

    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 chuoi 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--;\n\n    }\n\n    cout<<a;\n}\n
```

19

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

20

```
void daothutu(char *s)\n{\n    char c[255];\n    c[0]='\\0';//ham strcat noi chuoi tai vi tri NULL\n\n    for(int i=strlen(s)-1;i>=0;i--)\n\n        if(s[i]==' ')\n        {\n            strcat(strcat(c,s+i+1)," ");\n            s[i]='\\0';\n\n        }\n}\n
```

```
    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)))//truong hop tu giua va tu cuoi
            strcpy(s+t-1,s+t+d); // s+t-1 la khoang trang
        if(s[t+d]==' ' && t==0)//truong hop xoa tu dau tien
            strcpy(s,s+t+d+1);
    }
    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";  
}
```