

CHUYÊN ĐỀ XỬ LÝ SỐ LỚN

I. MỘT SỐ HÀM XỬ LÝ XÂU TRONG C++

Chú ý khi sử dụng các hàm này ta phải khai báo thư viện `#include "string.h"`

- **Hàm `strcpy`:**

Công dụng: sao chép chuỗi nguồn vào chuỗi đích.

Cấu trúc: `char*strcpy(char *dich, char *nguồn);`

Có nghĩa là khi ta nhập vào một dãy các kí tự ở chuỗi nguồn thì nó sẽ sao chép tất cả các kí tự vừa nhập vào cái chuỗi đích.

ví dụ như sau:

Code c	Code c++
<pre>#include<conio.h> #include<stdio.h> #include<string.h> int main() { char A[255],B[255]; printf("Nhập chuỗi: "); gets(A); strcpy(B,A); printf("Chuỗi đích: "); puts(B); getch(); return 0; }</pre>	<pre>#include<iostream>; #include<string.h>; using namespace std; int main() { char A[255],B[255]; cout<<"Nhập chuỗi: "; cin>>A; strcpy(B,A); cout<<"Chuỗi đích: "<<B; return 0; }</pre>

Chương trình trên khi ta nhập vào mảng A một dãy các kí tự là "abc" thì khi gọi hàm `strcpy(B,A);` thì nó sẽ copy 3 kí tự "abc" từ mảng A vào mảng B.

Nếu chúng ta muốn copy n kí tự từ chuỗi nguồn vào chuỗi đích ta dùng hàm sau:

- **Hàm `strncpy`:**

Công dụng: sao chép n kí tự đầu tiên của chuỗi nguồn vào chuỗi đích.

Cấu trúc: `char *strncpy(char *dich, char *nguồn,int n);`

ví dụ như sau:

```
#include<iostream>;
#include<string.h>;
using namespace std;
int main()
{
    char A[255],B[255];
    cout<<"Nhập chuỗi: ";
    cin>>A;
    strncpy(B,A,5);
    cout<<"chuỗi sao chép: "<<B;
    return 0;
}
```

}

1. Hàm strlen:

Công dụng: cho biết độ dài của chuỗi s

Cấu trúc:

intstrlen(char *s)

Ví dụ: Sử dụng hàm strlen xác định độ dài một chuỗi nhập từ bàn phím.

Code c	Code C++
<pre>#include<conio.h> #include<stdio.h> #include<string.h> int main() { char Chuoi[255]; int Dodai; printf("Nhap chuoi: "); gets(Chuoi); Dodai = strlen(Chuoi); printf("Chuoi vua nhap:"); puts(Chuoi); printf("Co do_dai %d",Dodai); getch(); return 0; }</pre>	<pre>#include<iostream> #include<stdio.h>; #include<string.h>; using namespace std; int main() { char Chuoi[255]; int Dodai; cout<<"Nhap chuoi: "; cin>>Chuoi; Dodai = strlen(Chuoi); cout<<"Chuoi vua nhap:"<<Chuoi<<endl; cout<<"Co do_dai: "<<Dodai; return 0; }</pre>

4. Hàm strcat:

Công dụng: ghép chuỗi nguồn vào sau chuỗi đích.

Cấu trúc: char *strcat(char *dich,char *nguồn)

Ví dụ: Nhập vào họ lót và tên của một người, sau đó in cả họ và tên của họ lên màn hình.

Code c	Code C++
<pre>#include<conio.h> #include<stdio.h> #include<string.h> int main() { char HoLot[30], Ten[12]; printf("Nhap Ho Lot: "); gets(HoLot); printf("Nhap Ten: "); gets(Ten); strcat(HoLot,Ten); /* Ghep Ten vao HoLot*/ printf("Ho ten la: "); puts(HoLot); getch(); return 0; }</pre>	<pre>#include<iostream>; #include<conio.h>; #include<stdio.h>; #include<string.h>; using namespace std; int main() { char HoLot[30], Ten[12]; cout<<"Nhap Ho Lot: "; cin>>HoLot; cout<<"Nhap Ten: "; cin>>Ten; strcat(HoLot," "); strcat(HoLot,Ten); /* Ghep Ten vao HoLot*/ cout<<"Ho ten la: "<<HoLot; return 0; }</pre>

5. Hàm strncat:

Công dụng: ghép n kí tự đầu tiên của chuỗi vào sau chuỗi đích

Cấu trúc: char *strncat(char *dich,char *nguồn,int n);

```
#include<iostream>;
#include<string.h>;
using namespace std;
int main()
{
    char xau1[30], xau2[12];
    cout<<"Nhập xau 1: ";
    cin>>xau1;
    cout<<"Nhập nhập xau 2: ";
    cin>>xau2;
    strcat(xau1,xau2,5); /* Ghep Ten vao HoLot*/
    cout<<"xau sau khi ghep: "<<xau1;
    return 0;
}
```

6. Hàm strcmp:

Công dụng: so sánh 2 chuỗi s1 và s2

Cấu trúc: `int strcmp(char *s1, char *s2);`

Hàm sẽ trả về 1 trong các giá trị sau:

- Giá trị âm nếu chuỗi s1 nhỏ hơn chuỗi s2
- Giá trị 0 nếu hai chuỗi bằng nhau
- Giá trị dương nếu chuỗi s1 lớn hơn chuỗi s2

Ví dụ:

```
char *chu1 = "aaa", *chu2= "bbb", *chu3 = "aaa";
strcmp(chu1, chu2); //kết quả trả về - 1
strcmp(chu1, chu3); //kết quả trả về 0
strcmp(chu2, chu3); //kết quả trả về 1
ví dụ minh họa đây:
/*Nhập danh sách tên và sắp xếp theo thu tu tăng dần*/
```

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
#define MAXNUM 5
#define MAXLEN 10
int main(void)
{
    char ten[MAXNUM][MAXLEN]; //mang chuỗi
    char *c[MAXNUM]; //mang con trỏ trỏ đến chuỗi
    char *ct;
    int i, j, n = 0;
    //nhập danh sách tên
    while (n < MAXNUM)
    {
        printf("Nhập vào tên người thu %d: ", n + 1);
        gets(ten[n]);
        c[n++] = ten[n]; //con trỏ đến tên
    }
    //sắp xếp danh sách theo thu tu tăng dần
    for (i = 0; i < n - 1; i++)
    for (j = i + 1; j < n; j++)
```

```
if (strcmp(c[i], c[j]) > 0)
{
    ct = c[i];
    c[i] = c[j];
    c[j] = ct;
}
//In danh sach da sap xep
printf("Danh sach sau khi sap xep:\n");
for (i = 0; i < n; i++)
printf("Ten nguoi thu %d : %s\n", i + 1, c[i]);
getch();
}
```

7. Hàm strlwr:

Công dụng: chuyển tất cả các kí tự chuỗi về chữ thường

Cấu trúc: `char *strlwr(char *s);`

8. Hàmstrupr :

Công dụng: chuyển tất cả các kí tự chuỗi thường về chữ hoa

Cấu trúc: `char *strupr(char *s)`

Ví dụ: Viết chương trình nhập vào một chuỗi ký tự từ bàn phím. Sau đó sử dụng hàmstrupr(); để chuyển đổi chúng thành chuỗi chữ hoa.

```
#include<conio.h>
#include<stdio.h>
#include<string.h>
intmain()
{
    char Chuoi[255],*s;
    printf("Nhap chuoi: ");
    gets(Chuoi);
    s=strupr(Chuoi) ;
    printf("Chuoi chu hoa:");
    puts(s);
    getch();
    return0;
}
```

9. Hàm strrev :

Công dụng: đảo ngược chuỗi kí tự

Cấu trúc: `char *strrev(char *s);`

10. Hàm strchr:

Công dụng: trả về địa chỉ vị trí xuất hiện đầu tiên của kí tự ch trong chữ s và sẽ trả về giá trị NULL trong trường hợp không tìm thấy.

Cấu trúc: `char *strchr(char *s,int ch);`

11. Hàm strrchr:

Cấu trúc: `char *strrchr(char *s,char ch);`

Công dụng: trả về địa chỉ vị trí xuất hiện cuối cùng của kí tự ch trong chuỗi s. Nếu không tìm thấy hàm sẽ trả về giá trị NULL

12. Hàm strstr:

Công dụng: trả về địa chỉ vị trí xuất hiện đầu tiên của chuỗi s1 trong chuỗi s và sẽ trả về giá trị NULL trong trường hợp không tìm thấy.

Cấu trúc: **char *strstr(char *s, char *s1);**

Ví dụ: Viết chương trình sử dụng hàm strstr() để lấy ra một phần của chuỗi gốc bắt đầu từ chuỗi "hoc".

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

int main()
{
    char Chuoi[255],*s;
    printf("Nhap chuoi: ");
    gets(Chuoi);
    s=strstr(Chuoi,"hoc");
    printf("Chuoi trich ra:");
    puts(s);
    getch();
    return 0;
}
```

13. Hàm memset

Công dụng: Set num byte nhớ từ vị trí được trỏ tới bằng giá trị value

Cấu trúc: **void *memset (void *ptr, int value, size_t num);**

14. Hàm memcpy

Công dụng: Chép num byte từ vị trí mà source trỏ tới đến vị trí mà destination trỏ tới

Cấu trúc: **void *memcpy (void *destination, const void *source, size_t num);**

15. Hàm memcmp

Công dụng: So sánh giá trị các vùng nhớ mà ptr1 và ptr2 trỏ tới theo từng byte, sẽ dừng lại khi so sánh đủ num byte. Trả về -1 khi byte đầu tiên mà không trùng nhau của 2 vùng so sánh của ptr1 nhỏ hơn ptr2, trả về 0 khi 2 vùng nhớ bằng nhau, trả về 1 khi byte đầu tiên mà không trùng nhau của 2 vùng so sánh của ptr1 lớn hơn ptr2

Cấu trúc: **cint memcmp(const void *ptr1, const void *ptr2, size_t num);**

16. Hàm stricmp:

Công dụng: So sánh 2 chuỗi không phân biệt chữ hoa chữ thường, hàm trả về tương tự strcmp.

Cấu trúc: **cint stricmp (const char * string1, const char * string2);**

Ví dụ:

Ví dụ này sử dụng stricmp () để so sánh hai chuỗi.

```
#include <stdio.h>
#include <string.h>
int main (void)
{
    /* So sánh hai chuỗi như là chữ thường */
    if (0 == stricmp ("hello", "Hello"))
    if (0 == stricmp("hello", "HELLO"))
    printf("The strings are equivalent.\n");
    else
    printf("The strings are not equivalent.\n");
    return 0;
}
```

}

II. Xử lý số nguyên lớn

1. Cộng 2 số nguyên lớn

Phân tích thuật toán

- Bước 1: Chuẩn hóa hai xâu a, b để có độ dài bằng nhau. Nếu xâu nào có độ dài ngắn hơn thì thêm các '0' vào đầu xâu đó.

- Bước 2: Duyệt từ cuối hai xâu về đầu xâu:

+ Tạo xâu kết quả c=a;

+ Tách từng phần tử của hai xâu chuyển sang kiểu số;

+ Tính tổng:

$tổng = số\ 1 + số\ 2 + nhớ\ (ban\ đầu\ nhớ\ bằng\ 0);$

$nhớ = tổng / 10;$

$tổng = tổng \% 10;$

+ Chuyển đổi giá trị **tổng** tính được sang ký tự rồi gán vào xâu kết quả.

+ Lưu ý cộng thêm giá trị nhớ lần cuối nếu **nhớ khác '0'**.

Chương trình tham khảo

```
string Congxau(string a, string b)
{
    string c;
    long n1=a.length(),n2=b.length(),i,nho=0,Tong;
    if(n1>n2) b.insert(0,n1-n2,'0');
    if(n1<n2) a.insert(0,n2-n1,'0');
    c=a;
    for(i=a.length()-1;i>=0;i--)
    {
        Tong=(a[i]-48)+(b[i]-48)+nho;
        nho=Tong/10;
        Tong=Tong%10;
        c[i]=char(Tong+48);
    }
    if(nho>0) c=char(nho+48)+c;
```

```
    return c;  
}
```

2. Trừ 2 số nguyên lớn (Trừ số lớn cho số bé)

Phân tích thuật toán

- Bước 1: Chuẩn hóa hai chuỗi a, b để có độ dài bằng nhau. Nếu chuỗi nào có độ dài ngắn hơn thì thêm các '0' vào đầu chuỗi đó.

- Bước 2: Duyệt từ cuối hai chuỗi về đầu chuỗi:

+ Tạo chuỗi kết quả c=a;

+ Tách từng phần tử của hai chuỗi chuyển sang kiểu số;

+ Tính hiệu:

hiệu = số 1 - số 2 - mượn (ban đầu mượn bằng 0);

Nếu hiệu < 0 thì {hiệu = hiệu + 10; mượn = 1;}

Nếu hiệu > 0 thì mượn = 0;

+ Chuyển đổi giá trị **hiệu** tính được sang ký tự rồi gán vào chuỗi kết quả.

+ Xử lý chuỗi kết quả nếu chuỗi có độ dài lớn hơn 1 mà phần tử đầu tiên của mảng chuỗi là '0'.

Chương trình tham khảo

```
string Truxau(string a, string b)  
{  
    string c="";  
    long n1=a.length(),n2=b.length(),i,Muon=0,Hieu;  
    if(n1>n2) b.insert(0,n1-n2,'0');  
    for(i=a.length()-1;i>=0;i--)  
    {  
        Hieu=(a[i]-48)-(b[i]-48)-Muon;  
        if(Hieu<0){Hieu+=10;Muon=1;}else Muon=0;  
        c=char(Hieu+48)+c;  
    }  
    while(c.length()>1&&c[0]=='0') c.erase(0,1);  
}
```

```
return c;  
}
```

3. Nhân một số nguyên lớn với một nguyên số nhỏ

Phân tích thuật toán

- Bước 1: Duyệt từ cuối xâu số lớn về đầu xâu
- Bước 2: + Tách từng phần tử của xâu chuyển sang kiểu số và tính tích:
$$tích = số\ nhỏ * tg + nhớ\ (tg\ là\ số\ được\ tách\ từ\ xâu\ số\ lớn);$$
$$nhớ = tích / 10;$$
$$Tích = tích \% 10;$$
 - + Chuyển đổi giá trị **tích** tính được sang ký tự rồi gán vào xâu kết quả.
 - + Lưu ý cộng thêm giá trị nhớ lần cuối nếu **nhớ khác '0'**.

Chương trình tham khảo

```
string Nhanlso(string a, int k)  
{  
    string b;  
    long i, Nho=0, Tich;  
    for(i=a.length()-1; i>=0; i--)  
    {  
        Tich=Nho+(a[i]-48)*k;  
        Nho=Tich/10;  
        Tich=Tich%10;  
        b=b+char(Tich+48);  
    }  
    if(Nho!=0) b=char(Nho+48)+b;  
    while(b.length()>1&&b[0]=='0') b.erase(0,1);  
    return b;  
}
```

4. Nhân 2 số nguyên lớn

Phân tích thuật toán

- Duyệt từ cuối xâu a về đầu xâu.

- Tách từng phần tử của xâu a nhân với xâu b (Thuật toán nhân với số nhỏ).
- Cộng liên tiếp các kết quả thu được (lưu ý trước khi cộng 2 xâu thêm ký tự “0” vào sau xâu thứ 2).
- Xử lý các ký tự “0” trước xâu sau khi cộng.

Chương trình tham khảo

```
string Nhanxau(string a, string b)
{
    string x,Tg1="0",Tg2,c;
    long i,j=0;
    for(i=b.length()-1;i>=0;i--)
    {
        Tg2=Nhan1so(a,(b[i]-48));
        Tg2.insert(Tg2.length(),j,'0');
        j++;
        c=Congxau(Tg1,Tg2);
        Tg1=c;
    }
    return c;
}
```

5. Chia số nguyên lớn cho số nguyên nhỏ

Phân tích thuật toán

- Bước 1: Duyệt từ đầu xâu số nguyên lớn
- Bước 2:
- + Tách từng phần tử của xâu đem chia cho số nguyên nhỏ:

$$\text{chia} = \text{số} + \text{dur} * 10 \text{ (dur ban đầu bằng 0);}$$

$$\text{thương} = \text{chia} / \text{số nhỏ};$$

$$\text{dur} = \text{chia} \% 10;$$

- + Cộng liên tiếp các **thương** được phần nguyên;
- + Lưu lại giá trị **dur** cuối cùng được phần dư;
- + Lưu ý: xóa các “0” ở đầu mảng xâu kết quả.

```
void chia_so(char a[], long b, char div[], char mod[])
{
    long i, n = strlen(a), du = 0, so, chia, thuong;
    char tg[10], luu[100000] = "";
    for (i = 0; i < n; i++)
    {
        strncpy(tg, a + i, 1); tg[1] = '\0'; so = atoi(tg);
        chia = du * 10 + so; du = chia % b; thuong = chia / b;
        itoa(thuong, luu, 10);
        strcat(div, luu);
    }
    itoa(du, luu, 10); strcpy(mod, luu);
    i = 0;
    while (i < strlen(div) - 1 && div[i] == '0') i++;
    strcpy(luu, "");
    strncpy(luu, div + i, strlen(div) - i);
    luu[strlen(div) - i] = '\0';
    strcpy(div, luu);
}
```

5.6. Chia hai số nguyên lớn

Phân tích thuật toán

- Lấy số ký tự của xâu a bằng số ký tự của xâu b lưu vào xâu chia.
- Chừng nào số xâu chia còn lớn hơn xâu b thì tiến hành trừ liên tiếp xâu chia cho xâu b, ghi lại số lần trừ có thể là thương tìm được.
- Hạ từng ký tự của xâu a xuống xâu chia.
- Lưu ý: xử lý các ký tự “0” vô nghĩa của xâu a và xâu b.

Chương trình tham khảo

```
string Chiaxau(string a, string b, string &mod)
{
```

```
string div="",Tg="",Tg1="";

char luu[3];

long i,n1,n2,dem;

n1=a.length();n2=b.length();

if((n1<n2)|| (n1==n2)&&(a<b)){mod=a;return "0";}

else

{

    Tg=a.substr(0,n2);

    for(i=n2-1;i<a.length();i++)

    {

        dem=0;Tg=Tg+a[i];

        while(Tg.length()>1&&Tg[0]=='0')

            Tg.erase(0,1);

        while((Tg.length()>n2)|| (Tg.length()==n2)&&(Tg>=b))

            {dem++;Tg=Truxau(Tg,b);}

        itoa(dem,luu,10);

        div=div+luu;

    }

    mod=Tg;while((mod.length()>1)&&(mod[0]=='0'))

mod.erase(0,1);

}

return div;

}
```

III. Một số bài tập áp dụng

1. Dãy số Fibonacci ($n \leq 500$)

Dãy số Fibonacci được xác định bởi các công thức sau:

$$\begin{cases} F_0 = 0 \\ F_1 = 1 \\ F_n = F_{n-1} + F_{n-2} \text{ với } n \geq 2 \end{cases}$$

Một số phần tử đầu tiên của dãy số Fibonacci:

n	0	1	2	3	4	5	6	...
$Fibonacci_n$	0	1	1	2	3	5	8	...

Số Fibonacci là đáp án của các bài toán:

a) Bài toán cổ về sự sinh sản của các cặp thỏ với các giả thiết như sau:

- Các con thỏ không bao giờ chết;
- Hai tháng sau khi ra đời, mỗi cặp thỏ mới sẽ sinh ra một cặp thỏ con (một đực, một cái);
- Khi đã sinh con rồi thì cứ mỗi tháng tiếp theo chúng lại sinh được một cặp con mới.

Giả sử từ đầu tháng 1 có một cặp mới thì đến giữa tháng thứ n sẽ có bao nhiêu cặp?

b) Đếm số cách xếp $n-1$ quân domino kích thước 2×1 phủ kín bảng có kích thước $2 \times (n-1)$.

Hàm tính số Fibonacci thứ n bằng phương pháp lặp sử dụng công thức

$$F_n = F_{n-1} + F_{n-2} \text{ với } n \geq 2 \text{ và } F_0 = 0, F_1 = 1.$$

```
int fibo (int n)
{
    int f1, f2, fi;
    if (n<=1) return n;
    f2=0; f1=1;
    for (int i=2; i<=n; i++)
    {
        fi=f1+f2;      f2=f1;      f1=fi;
    }
}
```

Sử dụng phương pháp đệ quy:

```
int fibonacci(int n)
{
    if (n==0 || n==1)
        return 1;
    else
        return (fibonacci(n-2)+fibonacci(n-1));
}
```

2. Dãy số CATALAN ($n \leq 100$)

Số Catalan được xác định bởi công thức sau:

$$Catalan_n = \frac{1}{n+1} C_{2n}^n = \frac{(2n)!}{(n+1)!n!} \text{ với } n \geq 0.$$

Một số phần tử đầu tiên của dãy số Catalan là:

n	0	1	2	3	4	5	6	...
$Catalan_n$	1	1	2	5	14	42	132	...

Số Catalan là đáp án của các bài toán:

a) Có bao nhiêu cách khác nhau đặt n dấu ngoặc mở và n dấu ngoặc đóng đúng đắn?

Ví dụ: $n=3$ ta có 5 cách sau:

$((()))$, $((()))$, $(())()$, $(())()$, $(())()$

b) Có bao nhiêu cây nhị phân khác nhau có đúng $n+1$ lá?

Ví dụ: $n=3$



c) Cho một đa giác lồi $(n+2)$ đỉnh, ta chia thành các tam giác bằng nhau vẽ các đường chéo không cắt nhau trong đa giác. Hỏi có bao nhiêu cách chia như vậy?

Ví dụ: $n=4$

