# 字符串编程题精选

1. 编写程序，输入字符串（包含多个‘\*’），删除所有除前导和尾部以外的‘\*’：

#include<iostream>

#include<string.h>

using namespace std;

int main()

{char a[100],b[100];

int c,d,e,i,j,k,n,m;

cin>>a;

for(i=0;a[i]=='\*';i++)

{

b[i]=a[i];//前驱的星号赋给b一起输出

}

n=i;//前驱的星号数

for(j=strlen(a)-1;a[j]=='\*';j--)

m=j;//最后一个不是星号的坐标

for(j=n;j<=m;j++)//中间的 如果不是星号就赋给b m为最后一个不是星号的坐标

{if(a[j]!='\*')

{

b[i]=a[j];

i++;

}

}

for(j=m;a[j]=='\*';j++)//再把后面的星号赋给b；

{

b[i]=a[j];

i++;

}

b[i]=0;//截短字符串

cout<<b<<endl;

}

2.统计字符串里面的奇数和偶数

#include<iostream>

#include<string>

using namespace std;

void count(string s)

{

int i=0,a[100];

int dig,count1=0,count2=0;

while(s[i])

{

dig=s[i]-'0';

i++;

if(dig%2==0)

count1++;

else

count2++;

}

cout<<"偶数有"<<count1<<"个"<<endl;

cout<<"奇数有"<<count2<<"个"<<endl;

}

int main()

{

string s;

cout<<"input string(numbers):";

cin>>s;

count(s);

return 0;

}

3.字符串首尾互换后一起输出

#include<iostream>//首尾互换后一起输出

#include<string.h>

using namespace std;

int main()

{char aa[100],n,i;

cin>>aa;

n=strlen(aa);

for(i=0;i<n;i++)

aa[n+i]=aa[n-i-1];

aa[2\*n]='\0';

cout<<aa;

return 0;

}

1. 数字转换为字符串 （递归算法）

#include<iostream>

using namespace std;

void convert(int n)

{

int i;

char c;

if((i=n/10)!=0)//检查n是否为个位数

convert(i);

c=n%10+'0';

cout<<c<<" ";

}

int main()

{

int n;

cin>>n;

convert(n);

}

1. 汉诺塔递归

#include<iostream>

static int count=0;

using namespace std;

void move(char x,char y)

{

cout<<x<<"-->"<<y<<endl;

}

int hanoi(int n,char one,char two,char three)

{

if(n==1)

{

count++;

move(one,three);

}

else

{

count++;

hanoi(n-1,one,three,two);

move(one,three);

hanoi(n-1,two,one,three);

}

return count;

}

int main()

{

int m;

cout<<"input the number of disks:";

cin>>m;

cout<<"the steps are followed:"<<endl;

int a=hanoi(m,'A','B','C');

cout<<"it actived "<<a<<" times";

}

1. 递归计算数字和

#include<iostream>

using namespace std;

int tga(int \*a,int n)

{

if(n<1) return (0);

else return(a[n-1]+tga(a,n-1));}

int main()

{

int a[3]={1,2,3};

cout<<tga(a,3);

}

7.按照已有的顺序插入

#include<iostream>/

#define N 6

using namespace std;

int main()

{

int a[N]={1,12,13,14,16};

int i,j,x;

cin>>x;

//判断x的值是不是小于最小值

if(x<a[0]){

for(i=N-1;i>0;i--)

a[i]=a[i-1];

a[0]=x;

}

//判断x的值是不是大于最大值

if(x>a[N-1])

a[N-1]=x;

//x的值位于中间

for (i=0;i<N-1;i++){

if (x>a[i]&&x<=a[i+1]){

for (j=N-1;j>i+1;j--)

a[j]=a[j-1];

a[j]=x;

}

}

for (i=0;i<6;i++)

cout<<a[i]<<" ";

}

8.删除字符串中所有重复的字符，例如google，删除重复字符后为gole

#include <iostream>

using namespace std;

char \* deleteRepeatedChars(char \*s) {

if (s == NULL)

return s;

char \*fast = s;

char \*slow = s;

//char为一个字节，8位即能代表0-255

//使用哈希表思想，将字符的acsⅡ码作为下标，值为bool值（0或1）来代表是否重复了

bool arr[256] = { 0 };

while (\*fast != '\0')

{

//当不重复的时候，记录下slow

if (arr[\*fast] == 0)

{

arr[\*fast]=1;

\*slow = \*fast;

slow++;

}

fast++;

}

\*slow = '\0';//关键一步，截止字符串s

return s;

}

int main() {

char s[100];

gets(s);

// char \*s = aa;

cout<< deleteRepeatedChars(s)<<endl;

}

1. 统计子字符串

#include<iostream>

#include<string>

using namespace std;

int str\_count(string substr, string str)

{

int times=0, num=0,i=0;

int a=substr.length();

int b=str.length();

if(a>b)

return 0;

else

{

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

{

if(str[i]==substr[0])//如果跟字字符串第一个相同的话就开始下面循环

for(int j=0;j<a;j++)

{

if(str[i+j]==substr[j])

{

num++;

continue;

}

else

break;

}

}

times=num/a;

return times;

}

}

int fun(char \*str,char\*substr)//第二种函数方法

{

int i, j, k , num=0;

for (i=0；str[i]；i++)

for (j=i, k=0; substr[k]==str[j]; k++, j++)

if (substr[k+1]=='\0')

{

num++;

break;

}

return num;

}

int main()

{

string str;

cout<<"plz input str:"<<endl;

cin>>str;

string substr;

cout<<"plz input substr:"<<endl;

cin>>substr;

cout<<"the repeated is "<<str\_count(substr,str)<<" times"<<endl;

while(1);

return 0;

}

1. 删除你想删除的字符

#include<iostream>

using namespace std;

void fun(char \*p,char a)

{

char s[100];

int n=0,i=0;

while(p[i]!='\0')

{

if(p[i]==a)

{

i++;

continue;

}

s[n]=p[i];

n++;

i++;

}

int main()

{

char a[100];

char b;

cout<<"输入你的字符串:";

cin.getline(a,100);

cout<<"现在输入你想删除的字符:";

cin>>b;

cout<<"新的字符串为:";

fun(a,b);

return 0;

}

1. 去除开头空格

#include<iostream>

#include<string.h>

using namespace std;

int main()

{

char str[100],copy[100];

int i=0,start;

cin.getline(str,100);

int len=strlen(str);

cout<<"original string:"<<"["<<str<<"]"<<endl;

while(str[i])

{

if(str[i]==' ')

i++;

else

{

start=i;

break;

}

}

for (i= start; i < len; i++)

{

copy[i-start] = str[i];

}

cout<<"new string:"<<"["<<copy<<"]"<<endl;

return 0;

}

1. 去除末尾空格

#include<iostream>

using namespace std;

int main()

{

char str[100];

int i,len=0;

cin.getline(str,60);

while(str[len]!='\0')

{

len++;

}

i=len-1;

cout<<"原字符串为:"<<"["<<str<<"]"<<endl;

while(str[i]==' ')

{

str[i]='\0';

i--;

}

cout<<"新字符串为:"<<"["<<str<<"]"<<endl;

return 0;

}

1. 去除所有空格

#include <iostream>

using namespace std;

int main()

{

char s[81];

int i,n;

cin.getline(s,80);

i=0;

n=0;

while(s[i])

{

if (s[i]==' ')//如果等于空格 往下面递增 且退出本次循环

{

i++;

continue;

}

s[n]=s[i];

n++;

i++;

}

s[n]='\0'; //关键一步 截短字符串

cout<<s<<endl;

return 0;

}

1. 字符串转化为整数

#include<iostream>

#include<string.h>

#include<string>

using namespace std;

int Convert(string str)

{

int num=0,digit;

int i=0;

bool flag=true;

if(str[i]=='-')

{

flag=false;

for(i=1;i<str.size(); i++)

{

digit =str[i]-'0';

num = num\*10 + digit;

}

num=-num;

return num;

}

if(flag)

{

for(i=0;i<str.size(); i++)

{

digit =str[i]-'0';

num = num\*10 + digit;

}

return num;

}

}

int main()

{

string s;

cin>>s;

int n = Convert(s);

cout << n << endl;

}

1. 整数转化为任意进制

#include<iostream>

using namespace std;

void convert(int n,int x)

{

int i=0,m=0,j=0,count=0;

char s[10];

char c;

while(n!=0){

m=n%x;

if(m>=0&&m<=9)

s[i]=m+'0';//把数字的ASCII码赋给这个数组

else if(m>=10&&m<=15)s[i]=m-10+'A';//如果大于 就把字母的ASCII码赋给数组

n=n/x;

count++;

i++;

}

i--;//因为只要倒i-1个数字

m=i/2;//除2是因为倒过来从中间开始

for(j=0;j<=m;j++,i--){

c=s[i];//倒过来赋值

s[i]=s[j];

s[j]=c;

}

for(j=0;j<=count;j++)

if((s[j]>='0'&&s[j]<='9')||(s[j]>='A'&&s[j]<='Z')||(s[j]>='a'&&s[j]<='z'))

cout<<s[j];

}

int main()

{

int n,x;

cout<<"请输入一个整数和要转化的进制:";

cin>>n>>x;

convert(n,x);

}

1. 字符串转为为任意进制

#include<iostream>

#include<string>

#include<math.h>

using namespace std;

void convert(string s,int x)

{

int i=0,count,sum;

while(s[i]!='\0')

{

int count=s.length();

sum=0;

for(i=count-1;i>=0;i--)//从十六进制个位开始，每位都转换成十进制

{

if(s[i]>='0'&&s[i]<='9')//数字字符的转换

{

sum+=(s[i]-48)\*pow(x,count-i-1); //count-i-1是因为要倒过来

}

else if(s[i]>='A'&&s[i]<='F')//字母字符的转换

{

sum+=(s[i]-55)\*pow(x,count-i-1);

}

}

cout<<sum;

}

}

int main()

{

string s;

int n;

cout<<"请输入字符串和要转化的进制:";

cin>>s>>n;

convert(s,n);

}

1. 逆序输出 正常输出 并且求位数

#include<stdio.h>

int main()

{

int numCount(int);

void printNumber(int);

void printResNumber(int);

int num;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

if(scanf("%d",&num)!=1||num<0)

{

printf("输入不合格!!\n");

return 0;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

printf("%d是个 %d位数\n打印各位数字: ",num,numCount(num));

printNumber(num);

printf("\n逆序各位数字: ");

printResNumber(num);

putchar('\n');

return 0;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int numCount(int num)

{

int count;

for(count=0;num>0;num/=10)count++;

return count;

}

void printNumber(int num)

{

if(num>0)

{

printNumber(num/10);//递归

printf("%3d",num%10);

}

}

void printResNumber(int num)

{

if(num>0)

{

printf("%3d",num%10);

printResNumber(num/10);

}

}

1. 递归将整数转化为相应的字符串输出

#include <iostream>

using namespace std;

void int2str(int number)

{

int m=number/10;

char c= number%10 + '0';//转化为数字存入 (ASCII码的形式）

if(m)

{ int2str(m); cout<<c;}//如果读取成功 递归存入

else

cout<<(c);

}

int main() {

int n;

cin>>n;

int2str(n);

return 0;

}

1. 递归判断回文数

#include<iostream>

#include<cmath>

using namespace std;

void circle(int n,int x)

{

// while(scanf("%d",&b)!=EOF)

// {

int i,j,c,m;

double sum;

int a[20];

m=n;

c=0;

sum=0;

while(n!=0)//算出转化成某进制的数

{

i=n%x;

a[c]=i;

c++;

n=n/x;

}

c--;

for(j=0;j<=c;j++)//二进制本就是倒着输出的，在这里可以直接进行回文数判断

{

sum=sum+a[j]\*pow(x,c-j);

}

if(sum-m==0)//判断回文数转换成十进制以后和原数比较

printf("Yes\n");

else

printf("No\n");

}

int main()

{

int n,x;

cin>>n>>x;

circle(n,x);

}

1. 字符串赋值

#include<iostream>

using namespace std;

char \*strcopy(char \*s1,char \*s2)

{

int i=0;

char \*p=s2;

while(\*(s2+i)!='\0')

{

\*s1=\*s2;

s1++;

s2++;

}

s1='\0';

return p;

}

int main()

{

char s1[100];

char s2[100];

gets(s1);

gets(s2);

strcopy(s1,s2);

cout<<s1;

}

1. 字符串比较

#include<iostream>

using namespace std;

int strcmp(char \*s1,char \*s2)

{

int i=0;

while(\*(s1+i)==\*(s2+i))

if(\*(s1+i++)=='\0') return 0;//如果全部相等 返回0

if(\*s1>\*s2)

return 1;

else return (\*(s1+i)-\*(s2+i));

}

int main()

{

char a[100],b[100];

gets(a);

gets(b);

cout<<strcmp(a,b);

}

1. 字符串连接

#include<iostream>//字符串连接

#include<cmath>

#include<iomanip>

using namespace std;

char \*mycat(char \*s1, char \*s2)

{

//数组型

int i = 0;

while(s1[i] != '\0') {

i++;

}

int j = 0;

while(s2[j] != '\0') {

s1[i] = s2[j];

i++;

j++;

}

s1[i] = '\0';

return s1;

// 指针型

char \*p = s1; //定义字符型指针p指向s1

while(\*s1 != '\0') { //让s1指向'\0'

s1++;

}

while(\*s2 != '\0') { //让s2连在s1后

\*s1 = \*s2;

s1++;

s2++;

}

\*s1 = '\0'; //让s1以'\0'结尾

return p;

}

int main()

{

char s1[20],s2[20],\*p1,\*p2;

p1=s1,p2=s2;

cout<<"请输入两个字符串:";

gets(s1);

gets(s2);

cout<<mycat(s1,s2);

}

1. 删除除数字以外的东西

#include<iostream>

using namespace std;

int main()

{

char t[100];

char \*p=t,\*s=t;

cout<<"请输入字符串:"<<endl;

gets(t);

while(\*p)

{

if(\*p>='0'&&\*p<='9')

\*s++=\*p;

p++;

}

\*s='\0';//关键一步 截短字符串

puts(t);

}

1. 算单词数

#include<iostream>

using namespace std;

int main()

{

const int len=80;

char sentence[len+1],prev=' ';

int i,num=0;

cin.getline(sentence,len+1);

for(i=0;sentence[i]!='\0';++i)

{

if(prev==' '&&sentence[i]!=' ')

++num;

prev=sentence[i];

}

cout<<"单词个数为："<<num<<endl;

return 0;

}

1. 字符串碎片问题

#include<iostream>//字符串碎片问题

#include<iomanip>

#include<string>

using namespace std;

void component(string s,int n)

{

int t=1;

for(int i=0;i<n-1;i++)

{

if(s[i]!=s[i+1])

{

t++;

}

}

cout<<fixed<<setprecision(2)<<(float)n/t;

}

int main()

{

string s;

cin>>s;

component(s,s.size());

return 0;

}

1. 删除两端的空格

#include<iostream>//删除两端的空格

#include<string.h>

using namespace std;

char\* AfTrim(char\* src)

{

int len = strlen(src);

int start = 0; //copy字符串的开头引号

int end = 0; // copy字符串的结尾引号

int i = 0; //从头遍历的引号

int k = len-1; // 从尾遍历的引号

char\* copy = new char(len + 1); //申请一片相同大小的内存

while (src[i]) //从头遍历，直至找到不为空格终止

{

if (src[i] == ' ')

i++;

else

{

start = i;

break;

}

}

while (src[k]) //从尾遍历，直至找到不为空格终止

{

if (src[k] == ' ')

{

k--;

}

else

{

end = k+1;

break;

}

}

//进行copy

for (int j = start; j < end; j++)

{

copy[j-start] = src[j];

cout<<src[j];

}

return copy;

delete[]copy;

}

int main()

{

// char a[] = " i am fine ";

// AfTrim(a);

char t[100];

gets(t);

AfTrim(t);

return 0;

}

1. 把中文中的数字转化为中文数字

#include<iostream>

#include<string.h>

using namespace std;

char\* Change(char\* src)

{

int len = strlen(src);

int count = 0;

char\* copy = new char(len + 1);

char\* chinese[10] = { "零","一","二","三","四","五","六","七","八","九"}; //定义一个中文数字数组

for (int i = 0; i < len; i++)

{

if (src[i] == '\0') //终止符就结束

break;

if (src[i]>='0'&&src[i]<='9') //搜索到数字字符时

{

int num = src[i] - '0'; //得到数字对应的序号

char\* hanzi = chinese[num];//去除该序号对应的数组元素

copy[count] = hanzi[0];//因为一个中文字符占两个字节，所以要分开赋值

copy[count + 1] = hanzi[1];

count+=2;

}

else

{

copy[count] = src[i];//其他的都不管，照常copy

count++;

}

}

for (int j = 0; j < count; j++)

{

cout<<copy[j];

}

cout<<endl;

return copy;

delete[]copy;

}

int main()

{

char a[] = "我爱12你好34";

Change(a);

return 0;

}

1. 把一句话中的逗号转化为空格输出

#include<iostream>

#include<string.h>

using namespace std;

void split(char \*s)

{

int len=strlen(s);

char \*copy=new char[len+1];

for(int i=0;i<len;i++)

{

if(s[i]=='\0')

break;

if(s[i]==',')

{

copy[i]='\0';

}

else

{

copy[i]=s[i];

}

}

for(int j=0;j<len;j++)

{

if(copy[j]=='\0')

{

cout<<" ";

}

else

{

cout<<copy[j];

}

}

delete []copy;

}

int main()

{

char \*s="hello,everyone,good,morning";

split(s);

return 0;

}

1. 统计一个指针字符数组里面的数字并且输出

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include<iostream>

using namespace std;

int parse(const char\* input, int numbers[])

{

int count = 0;

//char buf[128];

int start, end;

int flag = 0; //字符是否有效，为1则有效，为0则无效

int len = strlen(input);

for (int i = 0; i < len; i++ )

{

if (input[i] == '\0')

break;

else if (input[i]>='0'&&input[i]<='9')

{

if (flag == 0)

{

start = i;

flag = 1;

}

}

else

{

if (flag ==1)

{

end = i;

flag = 0;

count++;

for (int j = start; j < end; j++)

{

cout<<input[j];

}

printf("\n");

}

}

}

cout<<"count="<<count;

return count;

}

int main()

{

int number[128];

char\* str = { "12 15 98 " };

parse(str, number);

return 0;

}

1. 得到字串

#include<iostream>//得到字串

#include<string.h>

#include<string>

using namespace std;

int substr(string src, int start, int num)

{

char buf[128];

for (int i = start; i < start+num; i++)

{

buf[i - start] = src[i];

cout<<src[i];

}

buf[num] = 0;

return 0;

}

int main()

{

string s;

int start,length;

cout<<"please enter a string:";

cin>>s;

cout<<"please enter the start position and length:";

cin>>start>>length;

while((length+start)>s.size())

{

cout<<"error! please input again!:";

cin>>start>>length;

}

substr(s,start, length);

return 0;

}

1. 回文数最简单算法

#include<iostream>

using namespace std;

int main()

{

int a,b,c,n,m,i;

cin>>n;

a=n;

c=0;

for(;a!=0;)

{

b=a%10;

a=a/10;

c=c\*10+b;

}

if(c==n)

cout<<"Yes"<<endl;

else

cout<<"No"<<endl;

return 1;

}

1. 插入排序

#include<iostream>

using namespace std;

int main()

{

int a[10];

int i,j,b;

for(i=0;i<10;i++)

cin>>a[i];

for(i=1;i<10;i++)

{

b=a[i];

for(j=i-1;j>=0 && b<a[j];j--)//往前面推

a[j+1]=a[j];后一个赋给前面一个（小的放后面）

a[j+1]=b;再把b给后面一个

}

for(i=0;i<10;i++)

cout<<a[i]<<" ";

}

1. 删除一个数

#include<iostream>

using namespace std;

int main()

{

int i,n,j,flag=0,pos,t;

int a[10];

for(i=0;i<10;i++)

cin>>a[i];

cout<<"input the number u wanna delete:";

cin>>n;

for(i=0;i<10;i++)

{

if(a[i]==n)

{

pos=i;

flag=1;

}

}

if(flag==1)

{

for(i=pos;i<10;i++)

a[i]=a[i+1];

for(i=0;i<8;i++)

cout<<a[i]<<" ";

cout<<a[8]<<endl;

}

else

cout<<"Not found"<<endl;

}

1. 字符串反转问题

#include <iostream>

using namespace std;

//字符串反转

char \*strReverse(char \*str){

// 判断字符是否为null或是空字符串

if (str == NULL || str == '\0') {

return str;

}

// 定义char数组指针

char \*start = str;

char \*end = str;

while (\*end != '\0'){//指向最后一个结点

end++;

}

// 交换头和尾的值

char temp;

--end;// 去除\0的位置

while (start < end){

temp = \*start;

\*start = \*end;

\*end = temp;

start++;

end--;

}

return str;

}

// 扩展版反转

char \* strReverseEx(char \*str){

// 判断字符是否为null或是空字符串

if (str == NULL || str == '\0') {

return str;

}

// 先整体反转

strReverse(str);

// 定义char数组指针

char \*start = str;

char \*end = str;

while (true)

{ // 然后根据空格分单词反转

if (\*end == ' ')

{

\*end = '\0';

strReverse(start);

\*end = ' ';

start = end+1;

}

// \*end=='\0'，说明到了字符串结尾

if (\*end=='\0')

{

strReverse(start);

break;

}

end++;

}

return str;

}

int main(void){

char str[] = "Welcome to Cangege Blog";

cout << strReverseEx(str) << endl;// 输出Blog Cangege to Welcome

system("pause");

return 0;

}

1. 输出字符串出现的次数以及出现的首位置

#include<string.h>

#include<iostream>

#define M 81

char temp\_str[30]; // 临时子串

using namespace std;

//fun()函数，ss代表母串，s代表子串.

int fun(char \*ss,char \*s){

int count=0;

char \*p,\*q;

while(\*ss!='\0'){

p=ss; //p指向ss首地址

q=s; //q指向s首地址

while(\*s!='\0'){

if(\*p==\*q){

p++;

q++;

}

else

{

ss++; //此处需注意，当字符比较不相等时，则需要将ss的地址向后移动一个。

break;

}

}

if(\*q=='\0')//如果全部等于 就+1

{

count++;

}

}

return count;

}

void ReadStrUnit(char \* str,char \*temp\_str,int idx,int len) // 从母串中获取与子串长度相等的临时子串

{

int index = 0;

for(index;index < len;index++)

{

temp\_str[index] = str[idx+index];

}

temp\_str[index] = '\0';

}

int GetSubStrPos(char \*str1,char \*str2)

{

int idx = 0;

int len1 = strlen(str1);

int len2 = strlen(str2);

if( len1 < len2)

{

cout<<"error!"; // 子串比母串长

return -1;

}

while(1)

{

ReadStrUnit(str1,temp\_str,idx,len2); // 不断获取的从 母串的 idx 位置处更新临时子串

if(strcmp(str2,temp\_str)==0)

break; // 若临时子串和子串一致，结束循环

idx++; // 改变从母串中取临时子串的位置

if(idx>=len1)return -1; // 若 idx 已经超出母串长度，说明母串不包含该子串

}

return idx; // 返回子串第一个字符在母串中的位置

}

int main()

{

char a[M],s[M];

gets(a);

gets(s);

int n=fun(a,s);

if(n<=0)

cout<<"not found!;";

else

cout<<n<<" "<<GetSubStrPos(a,s)<<endl;

return 0;

}

1. 返回子字符串的首位置以及个数改良版

#include<iostream>

using namespace std;

int substr\_find(char \*s,char \*substr)

{

int i=0,j=0;

int count=0;

while(s[i]!='\0')

{

while(substr[j]!='\0')

if(substr[j]==s[i])

{

i++;

j++;

}

else

{

i-=j;

j=0;

break;

}

if(j)

return i-j;

i++;

}

return -1;

}

int substr\_count(char \*s,char \*substr)

{

char \*p,\*q;

int count=0;

while(\*s!='\0')

{

p=s;

q=substr;

while(\*substr!='\0')

{

if(\*p==\*q)

{

p++;

q++;

}

else

{

s++;

break;

}

}

if(\*q=='\0')

{

count++;

}

}

return count;

}

int main()

{

char s1[100],s2[100];

gets(s1);

gets(s2);

cout<<substr\_find(s1,s2)<<" "<<substr\_count(s1,s2);

}

1. 删除一个字符简便方法

#include<iostream>

using namespace std;

void deleteC(char \*s,char x)

{

int i=0,j=0;

for(i=0;s[i];i++)

if(s[i]!=x)

s[j++]=s[i];

s[j]='\0';

}

int main()

{

char s[100],x;

gets(s);

cin>>x;

deleteC(s,x);

cout<<s;

}

1. 把字符串反转后连接之前的一起输出

#include<iostream>

#include<string.h>

using namespace std;

int main()

{

char str1[100],i;

int n;

cin>>str1;

n=strlen(str1);

for(i=0;i<n;i++)

{

str1[n+i]=str1[n-i-1];

}

for(i=0;i<n\*2;i++)

cout<<str1[i];

return 0;

}

1. 找出第一个不重复的字符

#include <iostream>

**using** **namespace** std;

**void**  f(**char**\* s)

{

**int** cnt,i,l;

**char**\* s1;

 l=**strlen**(s);

**for**(i=0;i<l;i++)

 {

  s1=s;

  cnt=0;

**while**(\*s1)

  {

**if**(\*s1==s[i])

   {

    cnt++;

**if** (cnt>1)

**break**;

   }

   s1++;

  }

**if**(cnt==1)

  {

     cout<<s<<": "<<s[i]<<endl;

**return**;

  }

 }

 cout<<s<<": "<<"没找到"<<endl;

}

**int** main()

{

 f("addbccadfeg");

 f("addbccabfeg");

 f("aabbccdd");

**return** 0;

}

1. 找出不重复的连续子列数目

#include<iostream>

#include<string>

using namespace std;

int lengthOfLongestSubstring(string s) {

int size,i=0,j,k,max=0;

size = s.size();

for(j = 0;j<size;j++){

for(k = i;k<j;k++)

if(s[k]==s[j]){

i = k+1;//如果相等就往下面递归一次 然后退出本层循环

break;

}

if(j-i+1 > max)//如果j的坐标减去子列的坐标大于0 那么存在

max = j-i+1;

}

return max;

}

int main()

{

string s;

cin>>s;

cout<<lengthOfLongestSubstring(s);

}

1. 删除s1中在s2中出现过的字符

#include<iostream>

#include<string>

using namespace std;

string RejectString(string s1,string s2)

{

string newstring = "";

int i, j;

for (i = 0; i < s1.size(); i++)

{

for (j = 0; j < s2.size(); j++)

{

if (s1[i] != s2[j])//如果不相等 就退出本次循环

{

continue;

}

else

{

j = 0;//如果相等 那么就初始化j为0 退出本层循环

break;

}

}

if (j == s2.size())//如果j等于s2的长度 说明全部相等

{

newstring += s1[i];//赋值

}

}

return newstring;

}

int main()

{

string s1,s2;

cin>>s1>>s2;

cout<<RejectString(s1,s2);

}

1. 删除子字符串

#include<iostream>

using namespace std;

char \*del\_substr(char \*str, char \*delstr)

{

char \*p, \*q;

char \*src, \*dst;

dst = src = str;

while(\*src != '\0')

{

p = src;

q = delstr;

while(\*p == \*q && \*q != '\0')

{

p++;

q++;

}

if (\*q == '\0')

{

src = p;//重新定位

}

else

{

\*dst++ = \*src++;//dst是用来备份的

}

}

\*dst = '\0';

return str;

}

int main()

{

char c1[100],c2[100];

gets(c1);

gets(c2);

cout<<del\_substr(c1,c2);

}

1. 往前面移动一位

#include<iostream>

#include<string.h>

using namespace std;

void fun(char \*s)

{

int i,len;

char tmp;

len=strlen(s);

if(len==0)

return;

tmp=s[0];

for(i=0;i<len-1;i++)

s[i]=s[i+1];

s[len-1]=tmp;

}

int main()

{

char s1[100];

gets(s1);

fun(s1);

puts(s1);

}

1. 右边循环n位置

void str\_exchange(char \*s, int n)

{

int len = strlen(s);

char \*ss = new char[len + 1];

for (int i = 0; i < n; i++)

ss[i] = s[len - n + i];//先后面n个赋值给前面的

for (int i = n; i < len; i++)

ss[i] = s[i - n];//再把后面len-n个赋值给ss

ss[len] = '\0';//给s's加结束符

for (int i = 0; i <= len; i++)

s[i] = ss[i];最后赋值给s

delete []ss;

}

1. 判断是否出现全部字符

#include<iostream>

#include<string.h>

using namespace std;

//判断字符串2中的字符是否全部在字符串1中出现

bool same(char\*str1, char\*str2)

{

int hashtable\_str1[256] = { 0 };

int hashtable\_str2[256] = { 0 };

for (int i = 0; i < strlen(str1 - 1); i++) //用表1记录字符串1的字符

{

hashtable\_str1[str1[i]] = 1; //出现则为1，没出现则为0

}

for (int i = 0; i < strlen(str2 - 1); i++) //表2记录字符串2的字符

{

hashtable\_str2[str2[i]] = 1;

}

for (int i = 0; i < 256; i++) //比较表1和表2

{

if (hashtable\_str1[i] < hashtable\_str2[i]) return false;

}

return true;

}

int main()

{

char string1[50];

char string2[50];

cout << "string1:";

cin >> string1;

cout << "string2:";

cin >> string2;

int result=same(string1, string2);

if (result == 0)

cout << "no";

else

cout << "yes";

system("pause");

}

#include<iostream>

#include<string.h>

using namespace std;

void stringContain(char \*s,char \*sub)

{

int count=0;

int alen=strlen(s),blen=strlen(sub);

int i;

for(i=0;i<blen;i++)

{

int j=0;

for(j=0;j<alen;j++)

if(sub[i]==s[j])

count++;

}

if(count==blen)

cout<<"yes";

else

cout<<"no";

}

int main()

{

char s1[10],sub[10];

gets(s1);

gets(sub);

stringContain(s1,sub);

}

1. 删除公共字符

#include<iostream>

#include<string>

using namespace std;

int main() {

// 注意这里不能使用cin接收，因为cin遇到空格就结束了。

// oj中IO输入字符串最好使用getline。

string str1, str2;

//cin>>str1;

//cin>>str2;

getline(cin, str1);

getline(cin, str2);

// 使用哈希映射思想先str2统计字符出现的次数

int hashtable[256] = {0};

for (size\_t i = 0; i < str2.size(); ++i) {

hashtable[str2[i]]++;

}

// 遍历str1，str1[i]映射hashtable对应位置为0，则表示这个字符在

// str2中没有出现过，则将他+=到ret。注意这里最好不要str1.erases(i) // 因为边遍历，边erase，容易出错。

string ret;

for (size\_t i = 0; i < str1.size(); ++i)

{

if (hashtable[str1[i]] == 0)

ret += str1[i];

}

cout << ret << endl;

return 0;

}

1. A和B的最长公共字串

#include<iostream>

#include<string.h>

using namespace std;

char \*strcmp(char \*s1,char \*s2,char \*dest)

{

char\*scmp2=s2;

int count = 0, temp = 0;

char \*s3 =NULL, \*s4=NULL;

while (\*s1 != '\0'&&\*scmp2 != '\0')

{

s3 = s1;

while (\*s1 == \*scmp2)

{

temp++;//计数

s1++;

scmp2++;

}

if (temp > count)//若大于就把temp的值转给count

{

count = temp;

s4 = s3;//再赋值指针

}

temp = 0;//重新归零

s1++;//往下面继续找

scmp2 = s2;//把第二个字串再重新赋值

}

for (int i = 0; i < count; i++)

{

\*dest++ = \*s4++;/

}

\*dest = '\0';

return dest;

}

int main()

{

char a[10],b[10],c[10];

gets(a);

gets(b);

strcmp(a, b,c);

cout <<c<< endl;

system("pause");

return 0;

}