1.功能:输入一个正整数，按照从小到大的顺序输出它的所有质因子（如180的质因子为2 2 3 3 5 ）

最后一个数后面也要有空格

#include<stdio.h>

void get\_rt(int *num*)

{

   for(int i = 2; i <= num; i++)

   {

       while((num%i)==0)

       {

           printf("%d ",i);

           num/=i;

       }

   }

}

int main(void)

{

    int num;

    int index = 0;

    while(scanf("%d",&num)!=EOF)

    {

        get\_rt(num);

    }

    return 0;

}

2.输入一个int型整数，按照从右向左的阅读顺序，返回一个不含重复数字的新的整数。

#include "stdio.h"

#include "string.h"

int main()

{

    char    strin[100];

    int sz[1000]={0},i;

  scanf("%s",strin);

    for(i=strlen(strin)-1;;i--)

    {

        sz[strin[i]]++;

        if(sz[strin[i]]==1)

            printf("%c",strin[i]);

        if(i==0)

            break;

    }

*//    for(i=0;i<10;i++)*

*//  printf("\n%d ",sz[i+'0']);*

*//*

*////printf("\n%d",'5');*

}

人生苦短 我用phython

while True:

    try:

        s1=int(input())

        s2=int(input())

        print(s1+s2)

    except:

        break

输入描述:

输入一个整数

输出描述:

计算整数二进制中1的个数

#include<stdio.h>

int main()

{

    int count=0,num=0;

    while(scanf("%d",&num)!=EOF)

    {

        count=0;

        while(num/2!=0)

        {

            if(num%2==1)

                count++;

            num/=2;

        }

        if(num)

            count++;

        printf("%d\n",count);

    }

    return 0;

}

密码要求:

1.长度超过8位

2.包括大小写字母.数字.其它符号,以上四种至少三种

3.不能有相同长度超2的子串重复

说明:长度超过2的子串

#include "stdio.h"

#include "string.h"

int chongfu(char \**str*)

{

    char temp[4];

    temp[3]='\0';

    for(;\*(str+2)!='\0';str++)

    {

        memcpy(temp,str,3);

        if(strstr(str+3,temp)!=NULL)

            return 1;

    }

    return 0;

}

int main()

{

    char str[100];

    while(scanf("%s",str)!=EOF)

    {

        int cnt,i,dz=0,xz=0,sz=0,other=0,s=0;

        int len=strlen(str);

        if(len<=8)

            printf("NG\n");

        else

        {

            for(i=0;i<len-1;i++)

            {

                if(str[i]>='a'&&str[i]<='z')

                    xz=1;

                else if(str[i]>='A'&&str[i]<='Z')

                    dz=1;

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

                    sz=1;

                else other=1;

            }

            cnt=xz+dz+sz+other;

             if(cnt>=3)

            {

                   s++;

            }

            if(chongfu(str)==0)

            {

                s++;

            }

             if(s==2)

                 printf("OK\n");

              else    printf("NG\n");

        }

    }

}

字符串排序

#include <stdio.h>

#include <string.h>

int main(void)

{

    char str[1000]={0},temp[1000]={0};

    while(gets(str))

    {

        int i,j,k=0;

        char temp[1000]={0};

        for(i=0;i<26;i++)

        {

            for(j=0;j<strlen(str);j++)

            {

                if(str[j] == 'a'+i || str[j] == 'A'+i)

                    temp[k++] = str[j];

            }

        }

        k=0;

        for(i=0;i<strlen(str);i++)

        {

            if((str[i]>='a'&&str[i]<='z')||(str[i]>='A'&&str[i]<='Z'))

                str[i] = temp[k++];

        }

        printf("%s\n",str);

    }

    return 0;

}

回文

#include <stdio.h>

#include <string.h>

*函数名：palindrome\_passwd\_len*

*输入参数：类型char \*s ,表示一个密码字符串;类型int step=1表示判断偶数回文，step=2表示判断基数回文*

*输出参数：类型int，表示回文密码长度*

*返回值：返回回文密码最大有效长度*

int palindrome\_passwd\_len(char \**s*,int *step*)

{

    int len = strlen(s);

    int low = 0,high = 0;

    int max = 0;

    if(NULL == s){

        return 0;

    }

    for(int i = 0;i < len;i++){

        low=i;

        high = low + step;

        while(low >= 0 && high < len && (s[low] == s[high])){

            low--;

            high++;

        }

        if((high-low-1) > max){

            max = high-low-1;

        }

    }

    return max;

}

int main(void)

{

    char str[10000];

    int max= 0,len=0;

    while(gets(str))

    {

        len = palindrome\_passwd\_len(str,1);

        max = palindrome\_passwd\_len(str,2);

        if(len >= max){

            printf("%d\n",len);

        }else{

            printf("%d\n",max);

        }

    }

    return 0;}

蛇形矩阵

#include <stdio.h>

#include<string.h>

#include <stdlib.h>

int main()

{

    int arr[100][100];

    int N;

    int i,j;

    arr[0][0] = 1;

    while(scanf("%d",&N)!=EOF)

    {

*// printf("%d",arr[0][0]);*

        for(i=0; i<N; i++)

        {

            if(i>0)

            {

                arr[i][0] = arr[i-1][0]+i;

            }

            printf("%d",arr[i][0]);                  *//注意这里的空格*

            for(j=1; j<N-i; j++)

            {

                arr[i][j] = arr[i][j-1]+j+1+i;

                printf(" %d",arr[i][j]);            *//注意这里的空格，若是%d后面空就不对了*

            }

           printf("\n");

        }

    }

}

加密TRAILBLAZERS

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

void encrypt(char \* *key*, char \* *data*, char \* *out*){

    int cnt = 0, i=0, j=0;

    char book[26] = {0};

    for(i=0; key[i] != '\0'; i++){

        if(key[i]>='a')

            key[i] -= 'a'-'A';

        for(j=0; j<cnt; j++){

            if(key[i]==book[j])

                break;

        }

        if(j==cnt){

            book[cnt] = key[i];

            cnt++;

        }

    }

    for(i='A'; i <= 'Z'; i++){

        for(j=0; j<cnt; j++){

            if(i==book[j])

                break;

        }

        if(j==cnt){

            book[cnt] = i;

            cnt++;

        }

    }

*// 加密*

    for(i=0; data[i]!='\0'; i++){

        if(data[i]>='a'){

            data[i] -= 'a' - 'A';

            out[i] = book[data[i]-'A'] + 'a' - 'A';

        }else{

            out[i] = book[data[i]-'A'];

        }

    }

    out[i] = '\0';

}

int main(void){

    char key[1000];

    while(scanf("%s", &key) != EOF){

        char data[3000]={0}, out[3000]={0};

        scanf("%s", &data);

        encrypt(&key, &data, &out);

        printf("%s\n", out);

        bzero(key, 1000);

    }

    return 0;

}

*// 有一只兔子，从出生后第3个月起每个月都生一只兔子，*

*// 小兔子长到第三个月后每个月又生一只兔子，*

*// 假如兔子都不死，问每个月的兔子总数为多少？*

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

main()

{

    int mon,a=1,b=0,c=0,sum=0;

    while(scanf("%d", &mon)!=EOF)

    {

   int fb=0,fm=1,sm=0,total=0;

        int i;

        for(i=0;i<mon;i++)

        {

            total=fb+fm+sm;

            fb=sm+fb;

            sm=fm;

            fm=fb;

        }

        printf("%d\n",total);

    }

}

编写一个截取字符串的函数，输入为一个字符串和字节数，输出为按字节截取的字符串。

但是要保证汉字不被截半个，如"我ABC"4，应该截为"我AB"，输入"我ABC汉DEF"6，

应该输出为"我ABC"而不是"我ABC+汉的半个"。

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

int main()

{

    char str[1000];

    while(scanf("%s",str)!=EOF)

    {

        int i,j,sum=0,num;

        scanf("%d",&num);

        for(i=0;i<num;i++)

        printf("%c",str[i]);

        printf("\n");

    }

}

四则运算

print input()

任意一个偶数（大于2）都可以由2个素数组成，组成偶数的2个素数有很多种情况

，本题目要求输出组成指定偶数的两个素数差值最小的素数对

#include<stdio.h>

int isprime(int *x*){   *//判断是否为素数*

    int flag=1;

    for(int i=2;i<x;i++){

        if(x%i==0){

            flag=0;

            break;

        }

    }

    return flag;

}

int main(){

    int n;

    while(scanf("%d",&n)!=EOF){

          for(int i=n/2;i>=2;i--){    *//从中间开始寻找*

              if(isprime(i)&&isprime(n-i)){

                 printf("%d\n",i);

                 printf("%d\n",n-i);

                 break;

              }

          }

    }

    return 0;

}

## **输入描述:**

输入一个string型基因序列，和int型子串的长度

## **输出描述:**

找出GC比例最高的子串,如果有多个输出第一个的子串

#include <stdio.h>

#include <string.h>

int main(void)

{

    char str[500]={0};

    int num;

    int bst[500]={0};

    while(gets(str))

    {

        scanf("%d",&num);

        int len;

        len=strlen(str);

        int i,j;

        int count=0;

        int k=0;

        for(i=0;i<=len-num;i++)

        {

            for(j=i;j<i+num;j++)

            {

                if(str[j]=='C'||str[j]=='G')

                {

                    count++;

*//printf("%d",count);*

                }

            }

            bst[i]=count;

            count=0;

        }                       *//接下来比较bst里的大小*

*/\*   for(i=0;i<len;i++)*

*{*

*printf("%d",bst[i]);*

*} \*/*

         int max=0;

         int x=0;

        for(i=0;i<len;i++)    *//获得最大子列*

        {

            if(bst[i]>max)

            {

                max=bst[i];

                x=i;

            }

          }

*//  printf("%d",max);*

*//  printf("%d",x);*

        for(i=x;i<x+num;i++)

        {

            printf("%c",str[i]);

         }

    }

    return 0;

}

配置文件恢复

#include <stdio.h>

#include <string.h>

int main()

{ char a[100];

 int i,flag,cnt,count,index;

     char \*buf1[] = {"reset", "board",

                     "board", "add",

                      "board", "delet",

                      "reboot", "backplane",

                      "backplane", "abort"};

    char \*buf2[]= {"board fault",

                      "where to add",

                      "no board at all",

                      "impossible",

                      "install first"};

 while(gets(a))

 {flag=0;

  count=0;

  index=0;

  cnt=0;

  for(i=0;a[i]!='\0';i++)

  {

      if(a[i]==' ')

      {

          flag=1;

          cnt=i;

          break;

      }

  }

  if(flag)

  {

    for(i=0;i<5;i++)

    {

        if(strncmp(buf1[2\*i],a,cnt)==0&&strncmp(buf1[2\*i+1],a+cnt+1,strlen(a)-cnt-1)==0)

        {

            count++;

            index =i;

        }

    }

     if(count==1)

            {

                printf("%s\n", buf2[index]);

            }

            else

            {

                printf("unkown command\n");

            }

  }

          else if(strncmp("reset",a,strlen(a))==0)

            {

                printf("reset what\n");

            }

            else

            {

                printf("unkown command\n");

            }

  }

return 0;}

题目描述

查找两个字符串a,b中的最长公共子串。若有多个，输出在较短串中最先出现的那个。

#include<stdio.h>

#include<string.h>

int main()

{

    char a[1000];

    char b[1000];

    char temp[1000];

    while(~scanf("%s",&a))

    {

        scanf("%s",&b);

        int len1=strlen(a);

        int len2=strlen(b);

        int i,j,k,m,z,length,max=0,end;

        if(len1>len2)

        {

            length=len1;

            len1=len2;

            len2=length;

            strcpy(temp,a);

            strcpy(a,b);

            strcpy(b,temp);

        }

      for(i=0;i<len1;i++)

       {

          for(j=0;j<len2;j++)

            {

                if(a[i]==b[j])

                {

                    k=0;

                    m=i;

                    z=j;

                    while(a[m]==b[z])

                    {

                        k++,m++,z++;

                        if(m>=len1||z>=len2)

                            break;

                    }

                    if(k>max)

                    {

                        max=k;

                        end=m;

                    }

                }

            }

       }

        for(i=end-max;i<end;i++)

            printf("%c",a[i]);

        printf("\n");

    }

    return 0;

}

矩阵乘法

#include<stdio.h>

int main()

{

    int x,y,z;

    while(scanf("%d%d%d",&x,&y,&z)!=EOF)

    {

        int loop,kkk;

        int a[x][y],b[y][z],c[x][z];

        int temp=0;

        for(loop=0;loop<x;loop++)

            for(kkk=0;kkk<y;kkk++)

                scanf("%d",&a[loop][kkk]);

        for(loop=0;loop<y;loop++)

            for(kkk=0;kkk<z;kkk++)

                scanf("%d",&b[loop][kkk]);

        for(loop=0;loop<x;loop++)

        {

            for(kkk=0;kkk<z;kkk++)

            {

                temp=0;

                for(int i=0;i<y;i++)

                    temp+=a[loop][i]\*b[i][kkk];

                printf("%d ",temp);

            }

            printf("\n");

        }

    }

    return 0;

}

通配符

#include <stdio.h>

#include <string.h>

int match(const char \**pattern*, const char \**str*)

{

    if (\*pattern == '\0' && \*str == '\0')

        return 1;

    else if (\*pattern == '\0' || \*str == '\0')

        return 0;

    if ((\*pattern == '?')|| (\*pattern == \*str))

    {

        return match(pattern+1, str+1); */\* 使用递归，若碰到'？'，则同时下移一位 \*/*

    }

    else if (\*pattern == '\*')

    {

*// 匹配0个1个或多个*

        return match(pattern+1, str) || match(pattern+1, str+1) || match(pattern, str+1);

    }

    return 0;

}

int main(void)

{

    char pattern[1000];

    char str[1000];

    while (scanf("%s %s", pattern, str)!=EOF)

    {

        int ret = match(pattern, str);

        printf("%s", ret?"true\n":"false\n");

    }

    return 0;

}

对于不同的字符串，我们希望能有办法判断相似程度，

我们定义了一套操作方法来把两个不相同的字符串变得相同，具体的操作方法如下：

1 修改一个字符，如把“a”替换为“b”。

2 增加一个字符，如把“abdd”变为“aebdd”。

3 删除一个字符，如把“travelling”变为“traveling”。

比如，对于“abcdefg”和“abcdef”两个字符串来说，

我们认为可以通过增加和减少一个“g”的方式来达到目的。上面的两种方案，都只需要一次操作。

把这个操作所需要的次数定义为两个字符串的距离，而相似度等于“距离＋1”的倒数。

也就是说，“abcdefg”和“abcdef”的距离为1，相似度为1/2=0.5.

给定任意两个字符串，你是否能写出一个算法来计算出它们的相似度呢？

#include "stdio.h"

int min(int *a*,int *b*,int *c*)

{

    int data = a;

    if ( data > b )

        data = b;

    if ( data > c )

        data = c;

    return data;

}

int main(void)

{

    int i,j;

    int len1;

    int len2;

    char str1[1000];

    char str2[1000];

    while ( gets(str1) )

    {

        gets(str2);

        len1 = strlen(str1);

        len2 = strlen(str2);

        char temp[len1+1][len2+1];

        temp[0][0] = 0;

        for ( i = 0;i < len1+1;i++ )

            temp[i][0] = i;

        for ( j = 0;j < len2+1;j++ )

            temp[0][j] = j;

        for ( i = 1;i < len1+1;i++ )

        {

            for ( j = 1;j < len2+1;j++ )

            {

                if ( str1[i-1] == str2[j-1] )

                    temp[i][j] = temp[i-1][j-1];

                else

                    temp[i][j] = min( temp[i-1][j-1]+1,temp[i-1][j]+1,temp[i][j-1]+1 );

            }

        }

        printf("1/%d\n",temp[len1][len2]+1);

    }

    return 0;

}

功能: 求一个byte数字对应的二进制数字中1的最大连续数，例如3的二进制为00000011，最大连续2个1

输入: 一个byte型的数字

输出: 无

返回: 对应的二进制数字中1的最大连续数

#include <stdio.h>

int main()

{

    int a;

    while (scanf("%d", &a) != EOF) {

        int num\_1 = 0;

        int max = 0;

        while (a)

        {

            if (a % 2 == 1)

            {

                num\_1++;

                if (num\_1 > max)

                    max = num\_1;

            }

            else

            {

                num\_1 = 0;

            }

            a = a / 2;

        }

        printf("%d\n", max);

    }

    return 0;

}