Compilation of C Program Questions — **Innovation Section**

by Jaser Li (HDU) The Latest Updated Version, on Dec.18th, 2023

* 字符塔

这题要求你输入一个变量n，输出一个n层的塔。

input

一个变量n (0<n<=10)

output

由 \* 堆成的 n 层的塔（如样例所示）

Samples

输入数据

4

输出数据

\*

\*\*\*

\*\*\*\*\*

\*\*\*\*\*\*\*

严格的格式检查：

输出中不允许包含多余的空格与回车！

#include <stdio.h>

int **main**()

{

    int n,i,j;

**scanf**("%d",&n);

    for(i=0;i<n;i++)

    {

        for(j=0;j<n-i-1;j++)

**printf**(" ");

        for(j=0;j<(2\*i+1);j++)

**printf**("\*");

**printf**("\n");

    }

}

* 尝试一种不基于N的循环

在C语言的学习中，你会发现许多题目都会给出数据的数量N。而有没有一种方法，能打破N的垄断，为你打开新的格局呢？

题目不给给出数据的数量。要求编写程序循环读取数据，并将其打印出来。

Input

连续的一组数据，中间以空格隔开

Output

原样打印出来，中间以空格隔开

Samples

输入数据

1 2 3 4

输出数据

1 2 3 4

Tips. 不妨去了解一下EOF的用法！

注意：当你在运行自测以及在其他编译软件中，一般均以\n作为结束标志。但在部分Online Judge平台（如hdufhq）中，提交测评以EOF标志结束。

#include <stdio.h>

int **main**(){

    int n;

    while(**scanf**("%d",&n)==1) // “==1” 表示scanf得到非空输入

**printf**("%d ",n); // 此处勿漏空格

}

就是这么简单（前提是要想到把scanf作为while循环的判断条件）。这个写法还需好好理解并记住！

【拓展】EOF（End of File）

EOF是一个表示文件结尾的标志符号。在C语言中，它用于表示从输入流（例如标准输入）读取数据时遇到文件结尾的情况。

EOF 实际上是一个特殊的整数常量，在 stdio.h 头文件中定义为 -1。当从输入流读取数据时，一旦达到文件结尾，就会返回 EOF。

EOF 的使用通常与输入函数（如 scanf、getchar 和 fgets 等）配合使用，以判断是否已经读取到文件的末尾。例如，我们可以使用 scanf 函数读取整数，直到读取到文件结尾：

int num;

while(**scanf**("%d",&*num*)!=**EOF**) // 处理读取到的整数

**printf**("Read number:%d\n",num);

在上述代码中，scanf函数每次尝试从输入流中读取一个整数，并将其保存在num变量中。如果成功读取到整数，则返回读取到的项目数（在这里应该为1），否则返回EOF（-1）。

循环会一直执行，直到遇到文件结尾。当输入流到达文件结尾时，scanf函数无法再继续读取数据，此时返回EOF，循环结束。

需要注意的是，在不同的平台和操作系统上，触发文件结尾的条件可能会有所不同。例如，在Windows系统中，可以使用Ctrl+Z（在一行的开头按下Ctrl键同时按下字母Z）作为文件结尾的标志符；而在UNIX和Linux系统中，可以使用Ctrl+D来表示文件结尾。

除了scanf函数，还可以在其他输入函数（如getchar、fgets等）中使用EOF值来判断文件结尾的情况。每个输入函数都有自己的方式来指示文件结尾。详细的用法可以参考相应的函数文档和示例代码。

* 素数判断

给你N个正整数，请依次判断每个数字是否为素数，若为素数，请输出1，不为素数则输出0.

Input

共有两行，第一行有一个正整数N, 第二行有N个正整数，为需要判断的数据

Output

N 个数据，分别对应每个数据是否为素数

Samples

输入数据

5

3 7 23 4 2

输出数据

1 1 1 0 1

Limitation

1s, 1024KiB for each test case.

对于100%的数据，有 1<=N<=10000

给出的数据将不包含1

#include <stdio.h>

int **sushu**(int *num*)

{

    if(num<=1)

        return 0;

    for(int i=2;i\*i<=num;i++){ // 注意：<=不要漏等号！！！

        if(num%i==0)

            return 0;

    }

    return 1;

}

int **main**()

{

    int N;

    int n;

**scanf**("%d",&N);

    for(int j=0;j<N;j++){

**scanf**("%d",&n);

**printf**("%d ",**sushu**(n));

    }

}

这一段sushu（英文懒得查，所以就汉语拼音啦hhh）函数的写法务必记牢！！！

* 文字压缩包

根据二进制数据长度N及对应二进制数，生成对应压缩数据：第一个数表示连续有几个 0，第二个数表示接着连续有几个 1，第三个数接着连续几个 0，第四个数接着连续几个 1，以此类推……

Input

第一行为二进制数据长度N，第二行为给定的二进制数据（给定的二进制数只会以0开头）

Output

压缩后的数据，数据间以空格隔开

Samples

输入数据

8

00001100

输出数据

4 2 2

#include <stdio.h>

int **main**()

{

    int N,count=0;

    char flag='0',s; //变量的初始类型应该分别声明为？

**scanf**("%d",&N);

    for(int i=0;i<N;i++)

    {

**scanf**(" %c",&s); //此处%c前面的空格有何作用？

        if(s==flag){

            count+=1;

        }else{

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

            if(flag=='1')flag='0';

            else flag='1';

            count=1;

        }

    }

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

    return 0;

}

当在使用 scanf 函数读取字符时，在格式字符串中使用 %c 时，它会自动读取输入缓冲区中的下一个字符，并将其赋值给变量 s。然而，%c格式并不会自动跳过之前的空白字符（如空格、制表符、换行符等）。

这意味着，如果前面还有一个或多个空白字符，scanf 函数将尝试读取并存储这些字符。这可能导致问题，尤其是当前一个 scanf 函数读取了其他非字符类型的输入时，留下了未处理的换行符或空格在输入缓冲区中。

为了解决这个问题，可在%c格式之前添加一个空格字符" "，告诉scanf函数跳过所有前导空白字符，直到遇到非空白字符为止。这样就有效地消耗掉了之前的换行符或空格，使%c格式只读取真正的字符输入。

例如，假设输入缓冲区中有一个换行符 \n，那么在使用 scanf(" %c", &s); 时，它会跳过这个换行符，并读取下一个非空白字符作为实际的字符输入。

这种技巧经常用于读取字符之前，确保跳过之前可能残留的空白字符，以避免潜在的错误或不一致性。

也可以将scanf("%d",&N);改为scanf("%d\n",&N);则不需要再加空格。

* 十进制转二进制

计算机的世界里只认2进制，请将我们熟知的十进制“语言”翻译为二进制“语言”。给一个0-65535的整形变量，输出对应的二进制数。

Input

0-65535的整形变量

Output

二进制数

Samples

输入数据

123

输出数据

1111011

#include <stdio.h>

int **main**()

{

    int num,i=0;

    int b[16]; //预留充足的位置，(65535)D即为(1111111111111111)B

**scanf**("%d",&num);

if(num==0) **printf**("%d",0);

    while(num!=0)

    {

        b[i]=num%2;

        num/=2;

        i+=1;

    }

    for(i--;i>=0;i--)

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

}

为什么试了好多遍都没对捏？逻辑上错了？？尊嘟假嘟？

倒不如仔细看看题干！！注意题干“给一个0-65535的整形变量”，还要考虑0噢！

而if(num==0) printf("%d",0);一步到位，能够直接解决这个问题。

另外，这个进制转换的代码一定要熟记！

* 移花接木

题目内容：

输入两个浮点数，将第二个数的小数部分拼接到第一个数的整数部分，

将第一个数的小数部分拼接到第二个数的整数部分。然后输出新生成的浮点数。

输入格式:

输入两个浮点数。（注意可能有负数）

输出格式：

输出新生成的两个浮点数。(输出保留6位小数)

输入样例1：

135.79 24.068

输出样例1：

135.068000 24.790000

输入样例2：

-1.23 4.56

输出样例2：

-1.560000 4.230000

#include <stdio.h>

int **main**(){

    double a,b,A,B;

**scanf**("%lf %lf",&a,&b);

    int a1=(int)a,b1=(int)b;

    double a2,b2;

    if((int)a>=0) a2=a-a1;

    else a2=a1-a;

    if((int)b>=0) b2=b-b1;

    else b2=b1-b;

if(a1>=0) A=a1+b2;

else A=a1-b2;

if(b1>=0) B=b1+a2;

else B=b1-a2;

**printf**("%lf %lf",A,B);

}

本题关键点即为如何对正负不同情况进行正确的分类。

在C语言中，**int类型的强制类型转换**可以通过使用类型转换运算符(int)或者使用函数int()来实现。对于正数，强制类型转换的结果通常是向下取整；而对于负数，强制类型转换的结果则是向上取整。

例如，以下代码演示了正数和负数的强制类型转换：

double d = 3.5; int i = (int)d;*//* **i 的值为 3**

double e = -3.5;int j = (int)e;*//* **j 的值为 -3**

double f = -3.2;int k = (int)f;*//* **k 的值为 -3**

double g = -3.8;int l = (int)g;*//* **l 的值为 -4**

需要注意的是，强制类型转换可能会导致精度损失或者数值溢出等问题，需要特别注意。

* 找出最大字符串

熟练使用库函数也是非常重要的技能哦^\_^

给你N个字符串，请通过比较找出N个字符串中值最大的那个。

Input

第一行，字符串个数N。 接下来每行为一个字符串。

Output

(唯一)最大字符串。

Samples

输入数据

2

haha

he

输出数据

he

#include <stdio.h>

#include <string.h> //必须引入，以进行字符串比较和替换操作！

#define **maxl** 100

int **main**()

{

    int N;

**scanf**("%d",&N);

    char s[**maxl**][**maxl**];

    char maxs[**maxl**]="";

    for(int i=0;i<N;i++)

    {

**scanf**("%s",s[i]);

        if(**strcmp**(s[i],maxs)>0)

**strcpy**(maxs,s[i]);

    }

**printf**("%s",maxs);

}

可以使用循环和比较变量的方法来解决这个问题。

上述代码中，使用了 strcpy()和strcmp()函数 来进行字符串拷贝和比较。

首先，我们用 scanf() 函数读取字符串的个数 n。然后，我们使用一个循环逐行读取输入的字符串，并将当前最大的字符串与输入的字符串进行比较。如果输入的字符串比当前最大字符串更大，就将其拷贝到 maxs 变量中。最后，我们将 maxs输出即可。

* 查找给定区间内的水仙花数

本题要求编写程序，输出给定正整数M和N区间内的所有三位水仙花数。三位水仙花数，即其个位、十位、百位数字的立方和等于该数本身。

输入格式:

在一行中给出两个正整数M和N（100≤M≤N≤999）。

输出格式:

顺序输出M和N区间内所有三位水仙花数，每一行输出一个数。

若该区间内没有三位水仙花数，则无输出。

如果M或者N不符合题目的要求，则输出Invalid Value.。

输入样例1:

100 400

输出样例1:

153

370

371

输入样例2:

500 600

输出样例2:

（输出为空）

输入样例3：

990 101

输出样例3：

Invalid Value.

#include <stdio.h>

#include <math.h>

int **main**(){

    int M,N;

**scanf**("%d %d",&M,&N);

    if(M>N || M<100 || N<100 || M>999 || N>999){

**printf**("Invalid Value.");

        return 0; // 在此处进行return有何作用？

    }

    for(int i=M;i<=N;i++){

        int a=i/100;

        int b=i/10%10;

        int c=i%10;

        if(**pow**(a,3)+**pow**(b,3)+**pow**(c,3)==i)

**printf**("%d\n",i);

    }

}

在本题中，当输入的M和N不符合要求时，我们需要输出"Invalid Value."并结束程序，此时使用return 0;可以让程序立即结束并返回数字0，告诉操作系统程序正常退出。这也是一个良好编程习惯，因为它可以帮助我们将程序的不同部分清晰地划分开来。

需要注意的是，main函数可以省略return语句，此时默认返回数字0。

* 大笨钟的心情

本题请你替大笨钟写一个程序，根据心情自动输出回答。

输入格式：

输入在一行中给出24个[0,100]区间内的整数，依次代表大笨钟在一天24小时中，每个小时的心情指数。

随后若干行，每行给出一个[0,23]之间的整数，代表网友询问笨钟这个问题的时间点。当出现非法的时间点时，表示输入结束，这个非法输入不要处理。题目保证至少有1次询问。

输出格式：

对每一次提问，如果当时笨钟的心情指数大于50，就在一行中输出 心情指数 Yes，否则输出 心情指数 No。

输入样例：

80 75 60 50 20 20 20 20 55 62 66 51 42 33 47 58 67 52 41 20 35 49 50 63

17

7

3

15

-1

输出样例：

52 Yes

20 No

50 No

58 Yes

#include <stdio.h>

int **main**(){

    int mood[24],hour;

    for (int i=0;i<24;i++){

**scanf**("%d",&mood[i]);

    }

    while(**scanf**("%d", &hour) == 1) {

        if(hour>=0 && hour<=23){

            if (mood[hour]>50) **printf**("%d Yes\n",mood[hour]);

            else **printf**("%d No\n",mood[hour]);

        }else break;

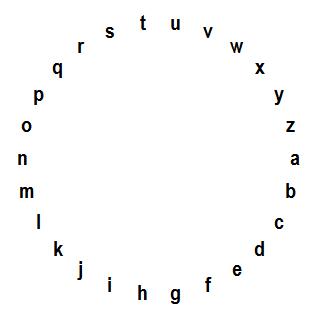
    }

}

这段代码首先会使用一个数组mood来存储每个小时的心情指数。然后会使用循环读取24个整数作为心情指数。接下来，使用另一个循环读取询问时间点，直到遇到非法的时间点（小于0或大于23）停止。

在每次循环中，根据给定的时间点，在数组mood中找到对应的心情指数。然后判断该指数是否大于50，并输出相应的回答。

注意，这里假设输入是符合格式要求的，即输入的时间点不会超出范围。在实际使用中，应该对输入进行适当的错误处理，以确保程序的健壮性。

* 环形加密（凯撒加密）

将26个英文字母按顺时针方向排成一个圆环。密钥是一个整数。

具体加密方法如下：

若密钥 key≥0，则从明文字母开始按顺时针方向走∣key∣步得到密文字母；

若密钥 key<0，则从明文字母开始按逆时针方向走∣key∣步得到密文字母。

请编写程序输入明文和密钥，输出密文。

输入格式

明文(小写字母) 密钥(任意整数)

输出格式

密文(小写字母)

输入样例1

a 5

输出样例1

f

输入样例2

b -3

输出样例2

y

#include <stdio.h>

int **main**()

{

    char ming;

    int key;

**scanf**("%c %d",&ming,&key);

    char mi=(ming+key-97+26)%26+97;

**printf**("%c",mi);

}

问：该代码存在什么错误？

k处理不当。GPT修改如下：

#include <stdio.h>

int **main**()

{

    char ming;

    int key;

**scanf**("%c %d",&ming,&key);

    int k=key%26;

    if(k<0)k+=26; //将k的值限定在0~25

    char mi='a'+(ming-'a'+k)%26; //此处写'a'或97等效

**printf**("%c",mi);

}

* 求e的近似值

自然常数e可以用级数1+1/1!+1/2!+...+1/n!+...来近似计算。本题要求对给定的非负整数n，求该级数的前n+1项和。

输入格式:

输入第一行中给出非负整数n（≤1000）。

输出格式:

在一行中输出部分和的值，保留小数点后八位。

输入样例: 输出样例:

10 2.71828180

根据题干要求，我们轻而易举写出了以下代码：

#include <stdio.h>

int **fac**(int *num*){if(*num*==0 || *num*==1)return 1;else return *num*\***fac**(*num*-1);}

int **main**(){

    int n;double m,sum=0.0;**scanf**("%d",&n);

    for(int i=0;i<=n;i++) sum+=1.0/**fac**(i);

**printf**("%.8f",sum);}

然而，当输入较大的数值时，阶乘函数的计算可能会导致结果溢出（例如1000→inf），并且在浮点数中存储结果也可能会超过表示范围，从而导致输出结果为inf（无穷大）。因此，在计算阶乘和级数之前，需要进行一些优化。一种常见的优化方法是使用对数运算来避免阶乘的直接计算。通过使用对数来计算阶乘的自然对数值，可以减小计算过程中的数值溢出风险。

#include <stdio.h>

#include <math.h>

double **facSum**(int *n*){

    double sum=0.0,term=1.0;

    for(int i=0;i<=*n*;i++){if(i>0) term/=i;sum+=term;}

    return sum;

}

int **main**(){

    int n;**scanf**("%d",&n);

    double sum=**facSum**(n);

**printf**("%.8f\n", sum);

}

【变式】幂级数展开式：IMG_256（完整题干略，保留四位小数）。

输入样例: 输出样例:

1.2 10 3.3201

#include <stdio.h>

#include <math.h>

int **main**(){

    double x;int n;

**scanf**("%lf %d",&x,&n);

    double sum=1.0,term=1.0;

    for(int i=1;i<=n;i++){term=term\*x/i;sum+=term;}

**printf**("%.4lf",sum);

}

* 倒着输出两数的乘积

做作业的时候，邻座的小盆友问你：“五乘以七等于多少？”你应该不失礼貌地微笑着告诉他：“五十三。”

本题要求，对任何一对给定的正整数，倒着输出它们的乘积。

输入格式：

在第一行给出两个不超过1000的正整数A和B，以空格分隔。

输出格式：

在一行中倒着输出A和B的乘积。

输入样例：

5 7

输出样例：

53

#include <stdio.h>

int **main**(){

    int A,B;

**scanf**("%d %d",&A,&B);

    int s=A\*B;

    while(s>0){

**printf**("%d",s%10);

        s/=10;

}

return 0;

}

修改如下：

#include <stdio.h>

int **reverse**(int *s*){

    int s1=0;

    while(*s*>0){

        s1=s1\*10+(*s*%10);

*s*/=10;

    }

    return s1;

}

int **main**(){

    int A,B;

**scanf**("%d %d",&A,&B);

    int s=A\*B;

    int s1=**reverse**(s);

**printf**("%d\n",s1);

    return 0;

}

* 数据归类

你将会得到一组无序的数据，将这些杂乱的数据进行归类，统计其中的字符个数、数字个数和行数。

Input

一组数据，保证其中只含有字符与数字

Output

字符个数、数字个数、行数，中间以空格隔开

Samples

输入数据

wel0come

to23fhq

输出数据

12 3 2

题目不算复杂，根据题意我们可以大致写出以下代码：

#include <stdio.h>

int **main**(){

    char ch[1000];

**fgets**(ch,sizeof(ch),**stdin**);

    int zi=0,shu=0,line=1; //此处line赋值为1

    for(int i=0;ch[i]!='\0';i++){

        if(ch[i]>='a'&&ch[i]<='z') zi+=1;

        else if(ch[i]>='0'&&ch[i]<='9') shu+=1;

        else if(ch[i]=='\n') line+=1;

    }

**printf**("%d %d %d",zi,shu,line);

    return 0;

}

然而，经过运行发现，该代码存在一个问题：题中样例为分行输入，而非由'\n'连接的一次性输入。该代码无法对多次输入的数据进行处理。

于是加入一个判断是否有新行输入的while循环结构，对代码修改如下：

#include <stdio.h>

int **main**(){

    char lst[1000];

    int zi=0,shu=0,line=0; //此处line赋值为0

    while(**fgets**(lst,sizeof(lst),**stdin**)!=**NULL** && lst[0]!='\n'){

        line+=1; //每有一行输入计数就+1, 故初值赋0

        for(int i=0;lst[i]!='\0';i++){

            if(lst[i]>='a'&&lst[i]<='z') zi+=1;

            else if(lst[i]>='0'&&lst[i]<='9') shu+=1;

        }

    }

**printf**("%d %d %d",zi,shu,line);

return 0;

}

【拓展】头文件 <ctype.h> 的用法

<ctype.h> 是C语言标准库中的头文件，提供了一组用于字符处理的函数。

以下是 <ctype.h> 头文件中常用的函数和它们的作用：

|  |  |
| --- | --- |
| isalpha(int c) | 判断给定的字符是否为字母（a-z、A-Z） |
| isdigit(int c) | 判断给定的字符是否为数字（0-9） |
| isalnum(int c) | 判断给定的字符是否为字母或数字 |
| islower(int c) | 判断给定的字符是否为小写字母（a-z） |
| isupper(int c) | 判断给定的字符是否为大写字母（A-Z） |
| isspace(int c) | 判断给定的字符是否为空白字符（空格、制表符、换行符等） |
| tolower(int c) | 将给定的字符转换为小写字母 |
| toupper(int c) | 将给定的字符转换为大写字母 |

这些函数可以用于对字符进行简单的判断、分类和转换操作，从而简化字符处理的任务。它们通常以字符的ASCII值作为参数，并返回一个非零值（真）或零值（假），表示字符是否满足特定的条件。

于是，我们对上述代码中的if-else if语句修改如下：

#include <ctype.h>

//......（略）

if(**isalpha**(lst[i])) zi+=1;

else if(**isdigit**(lst[i])) shu+=1;

【拓展】fgets的用法

fgets()是C语言标准库中的函数，用于从指定的流（文件、键盘等）读取一行字符数据。

fgets()函数的声明如下：

char \***fgets**(char \**s*, int *size*, FILE \**stream*);

其中，参数s是用于存储读取数据的字符数组地址；参数size是最多读取的字符数（包括终止符'\0'）；参数stream是要读取的流对象（文件、键盘等）。该函数返回值是指向字符串s的指针，或者是一个空指针（如果读取失败或者到达文件结尾）。

在上述代码中，我们在一个while循环中使用fgets( )函数逐行读取输入数据。具体而言，我们使用以下代码读取一行输入数据并将其存储在字符数组line中：

**fgets**(line, sizeof(line), stdin)

其中，line是用于存储读取数据的字符数组地址，sizeof(line)表示最多读取的字符数，即字符数组line的大小，而stdin表示从标准输入流（也就是键盘）中读取数据。

当读取操作成功完成时，fgets()函数将在字符数组line的末尾自动添加一个终止符'\0'，以便我们可以将其作为C语言中的字符串进行处理。如果读取失败或者到达文件结尾，fgets()函数将返回一个空指针，此时我们可以通过判断是否为NULL来处理这种情况。

* 多组坐标

假设校园门口为坐标原点(0,0)。学长念出w表示向上移动一格到(0,1)，s表示向下移动一格到(0,-1)，a表示向左移动一格到(-1,0)，d表示向右移动一格到(1,0)，以此类推。

随后学长会念出几个数字。第一个数字(假设为a)表示前a个字符连在一起构成第一个同学的宿舍坐标，第二个数字(假设为b)表示从a+1个字符开始往后的b个字符构成第二个同学的坐标。以此类推

Input

第一行是一串仅含w,s,a,d的字符串，可重复出现，顺序不定

第二行是n个数字，以空格间隔（后面没有换行符，n的大小随机）

Output

输出n个坐标(x,y)。括号用英文输入法打入，括号之间不用间隔

Samples

输入数据1 输出数据1

ddddd (2,0)(1,0)(2,0)

2 1 2

输入数据2 输出数据2

wdass (1,1)(-1,-2)

2 3

#include <stdio.h>

int **main**(){

    char dir[100];

    char ch;

    int i=0,j=0,step,x=0,y=0;

    while(**scanf**("%c",&ch) && ch!='\n'){ //后者ch已在前者赋值，可直接比较

        dir[i]=ch;

        i+=1;

    }

    i=0; // i赋0，给下一个while循环利用

    while(**scanf**("%d",&step)>0 && step!='\n'){ //后者step已在前者赋值，可直接比较

        for(j=0;j<step;j++){

            if(dir[i]=='a')x-=1;

            else if(dir[i]=='d')x+=1;

            else if(dir[i]=='s')y-=1;

            else if(dir[i]=='w')y+=1;

            i+=1;

        }

**printf**("(%d,%d)",x,y);

        x=0;y=0; // x,y赋0，给下一组数据使用

    }

}

* 连续自然数和

对一个给定的正整数M，求出所有的连续的正整数段（每一段至少有两个数），这些连续的自然数段中的全部数之和为 M。

例：1998+1999+2000+2001+2002=10000所以1998~2002的自然数段为M=10000的一个解。

Input

包含一个整数的单独一行给出M的值（10≤M≤2,000,000）。

Output

每行两个正整数，给出一个满足条件的连续正整数段中的第一个数和最后一个数，两数之间用一个空格隔开，所有输出行的第一个按从小到大的升序排列，对于给定的输入数据，保证至少有一个解。

Samples

输入数据

10000

输出数据

18 142

297 328

388 412

1998 2002

#include <stdio.h>

int **main**(){

    int M;

**scanf**("%d",&M);

    int start=1,end=2; // 初始化头、尾数值

int sum=start+end;

    while(1){ // while(1)循环一直进行，直到执行break语句

        if(sum==M)

**printf**("%d %d\n",start,end);

        if(sum<=M){ // sum小，右边界拓展

            end++;

            sum+=end;

        }else{ // sum大，左边界缩进

            sum-=start;

            start++;

        }

        if(end==M) break; // 当end==M，已经没有符合题意的组合，结束while循环

}

}

* 学生信息存储与获取

Input

第一行为整数N,M

接下来N行每一行给出学生的姓名（互不相同）、性别、年龄、学号

后M行每一行为一个学生姓名，数据间均用空格隔开。

Output

M行数据，每一行输出对应的学生性别、年龄、学号

Samples

输入数据 1

3 2

小冯 男 19 22046666

小李 男 18 22047777

小王 女 20 22048888

小李

小王

输出数据 1

男 18 22047777

女 20 22048888

#include <stdio.h>

#include <string.h>

struct Student {

    char name[100],gender[100];

    int age,id;

};

int **main**() {

    int N, M;

**scanf**("%d %d", &N, &M);

**getchar**();

    struct Student stu[N];

    for (int i = 0; i < N; i++) {

        char line[100];

**fgets**(line, sizeof(line), **stdin**);

**sscanf**(line, "%s %s %d %d", stu[i].name, stu[i].gender, &stu[i].age, &stu[i].id);

    }

    for (int i = 0; i < M; i++) {

        char name[100];

**fgets**(name, sizeof(name), **stdin**);

        name[**strcspn**(name, "\n")] = '\0';

        for (int j = 0; j < N; j++) {

            if (**strcmp**(name, stu[j].name) == 0) {

**printf**("%s %d %d\n", stu[j].gender, stu[j].age, stu[j].id);

                break;

            }

        }

    }

    return 0;

}

* 换硬币

将一笔零钱换成5分、2分和1分的硬币，要求每种硬币至少有一枚，有几种不同的换法？

输入格式:

输入在一行中给出待换的零钱数额x∈(8,100)。

输出格式:

要求按5分、2分和1分硬币的数量依次从大到小的顺序，输出各种换法。每行输出一种换法，格式为：“fen5:5分硬币数量, fen2:2分硬币数量, fen1:1分硬币数量, total:硬币总数量”。最后一行输出“count = 换法个数”。

输入样例:

13

输出样例:

fen5:2, fen2:1, fen1:1, total:4

fen5:1, fen2:3, fen1:2, total:6

fen5:1, fen2:2, fen1:4, total:7

fen5:1, fen2:1, fen1:6, total:8

count = 4

#include <stdio.h>

int **main**(){

    int x,a,b,c,count=0;

**scanf**("%d",&x);

    a=x/5;

    for(int i=a;i>0;i--){

        b=(x-5\*i)/2;

        for(int j=b;j>0;j--){

            c=(x-5\*i-2\*j)/1;

            for(int k=c;k>0;k--)

                if(i&&j&&k&&(i\*5+j\*2+k\*1==x)){

**printf**("fen5:%d, fen2:%d, fen1:%d, total:%d\n",i,j,k,i+j+k);

                    count+=1;

                }

        }

    }

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

}

* 找“完数”

所谓完数就是该数恰好等于除自身外的因子之和。例如：6=1+2+3，其中1、2、3为6的因子。本题要求编写程序，找出任意两正整数m和n之间的所有完数。

输入格式：

输入在一行中给出2个正整数m和n（1<m≤n≤10000），中间以空格分隔。

输出格式：

逐行输出给定范围内每个完数的因子累加形式的分解式，每个完数占一行，格式为“完数 = 因子1 + 因子2 + ... + 因子k”，其中完数和因子均按递增顺序给出。若区间内没有完数，则输出“None”。

输入样例：

2 30

输出样例：

6 = 1 + 2 + 3

28 = 1 + 2 + 4 + 7 + 14

#include <stdio.h>

int **main**(){

    int m,n,sum,count=0;

**scanf**("%d %d",&m,&n);

    while(m<=n)

    {

        sum=0;

        for(int j=1;j<m;j++)

            if(m%j==0) sum+=j;

        if(m==sum)

        {

**printf**("%d = 1",m);

            for(int k=2;k<m;k++)

                if(m%k==0) **printf**(" + %d",k);

**printf**("\n");

            count++;

        }

        m++;

    }

    if(!count) **printf**("None");

}

* 拆分输出

本题要求编写程序，对输入的一个整数，从高位开始逐位分割并输出它的各位数字。

输入格式：

输入在一行中给出一个长整型范围内的非负整数。

输出格式：

从高位开始逐位输出该整数的各位数字，每个数字后面有一个空格。

输入样例：

123456

输出样例：

1 2 3 4 5 6

#include <stdio.h>

int **main**(){

    long int n;

**scanf**("%ld",&n);

    if(n==0) **printf**("0 ");

    long int nn=n;

    int count=0;

    while(nn>0){

        nn/=10;

        count+=1;

    }

    for(int i=0;i<count;i++){

        nn=n;

        for(int j=0;j<count-i-1;j++) nn/=10;

**printf**("%d ",nn%10);

    }

}

* 兑换人民币

1元5角钱人民币兑换5分、2分和1分的硬币（每一种都要有）共100枚，会有很多种兑换方案。请编写程序给出各种兑换方案。

输入格式:

输入为一个正整数n，表示要求输出前n种可能的方案。方案的顺序，是按照5分硬币从少到多排列的。

输出格式:

显示前n种方案中5分、2分、1分硬币各多少枚。每行显示一种方案，数字之间空一格，最后一个数字后没有空格。注意：如果全部方案不到n种，就顺序输出全部可能的方案。

输入样例:

5

输出样例:

1 46 53

2 42 56

3 38 59

4 34 62

5 30 65

#include <stdio.h>

int **main**(){

    int sum=150,n,count=0,i,j,k;

**scanf**("%d",&n);

    for(i=1;i<=sum/5;i++){

        for(j=1;j<=(sum-i\*5)/2;j++){

            k=100-i-j;

            if(sum==i\*5+j\*2+k\*1){

**printf**("%d %d %d\n",i,j,k);

                count+=1;

            }

            if(count==n) break;

        }

    }

}

* 打印空心字符菱形

本题目要求读入菱形起始字母和菱形的高度，然后输出空心字符菱形。

所谓“空心菱形”是指：每行由两端为字母、中间为空格的字符串构成，每行的字符串中心对齐；上半部分相邻两行字符串长度差2，且字母从给定的起始字母逐一递增；下半部分与上半部分对称。

输入格式:

输入在一行中给出起始字母（范围为英文大写字母A-G）和菱形的高度（为不超过10的奇数）。

输出格式:

输出空心字符菱形。

输入样例:

B 5

输出样例:

B

C C

D D

C C

B

#include <stdio.h>

int **main**(){

    char st;

    int high;

**scanf**("%c %d",&st,&high);

    int hi=high/2;

    for(int a=0;a<hi;a++) **printf**(" ");

**printf**("%c\n",st);

    for(int b=0;b<hi;b++){

        for(int c=1;c<hi-b;c++) **printf**(" ");

**printf**("%c",st+b+1);

        for(int d=0;d<2\*b+1;d++) **printf**(" ");

**printf**("%c\n",st+b+1);

    }

    for(int bb=hi-2;bb>=0;bb--){

        for(int cc=1;cc<hi-bb;cc++) **printf**(" ");

**printf**("%c",st+bb+1);

        for(int dd=0;dd<2\*bb+1;dd++) **printf**(" ");

**printf**("%c\n",st+bb+1);

    }

    for(int aa=0;aa<hi;aa++) **printf**(" ");

    if(high>1) **printf**("%c",st); //单独列出

}

* 单个字符出现概率

小明最近对概率问题很感兴趣。一天，小明和小红一起玩一个概率游戏，首先小明给出一个字母和一个单词，然后由小红计算这个字母在这个单词中出现的概率。字母不区分大小写。

例如，给定的字母是a，单词是apple，那么概率是0.20000。

输入格式

输入包含多组测试数据。每组数据包含一个字母和一个单词。单词的长度不超过200。

输出格式

对于每一个输入，输出对应的概率，结果保留5位小数。

输入数据

a apple

c Candy

a banana

输出数据

0.20000

0.20000

0.50000

#include <stdio.h>

#include <string.h>

#include <ctype.h>

int **main**(){

    char le,word[200];

    while(**scanf**(" %c %s",&le,word)!=**EOF**){ // Question 1

        int count=0,total=0,len=**strlen**(word);

        for(int i=0;i<len;i++)

            if(**tolower**(word[i])==**tolower**(le)) count++; // Question 2

            total++;

**printf**("%.5f\n",(double)count/len);

    }

}

Question 1: 位于格式字符串'%c'前的空格有何作用？

跳过之前可能还在**输入缓冲区**中的空白字符（空格、制表符、换行等）并读取下一个**非空白字符**。

当输入字符时，我们通常会敲击键盘，每次敲击都会在输入缓冲区留下一个字符。如果在读取字符之前有一个空格字符（或其他空白字符）存在于输入缓冲区中，那么%c将会读取该空白字符，而非我们期望的字符。通过在格式字符串中加上空格字符，可以告诉输入函数跳过之前可能存在的空白字符，直到遇到第一个非空白字符为止。这样，我们就可以确保只读取到我们需要的有效字符。

这种方法通常用于与其他格式控制字符（例如%d、%f等）混合使用，以避免由于输入缓冲区中的残留空白字符导致的意外行为。但请注意，如果您确实需要读取空白字符，例如读取一行字符串（包括空白字符），则不需要在%c之前添加空格。

Question 2: 此处tolower函数的作用？

题干明确“字母不区分大小写”，将字母全部改为小写以便一一比较。

注意：必须先引入头文件 <ctype.h>，方可使用tolower()函数。

* 平均绩点

每门课的成绩分为A、B、C、D、F五个等级，为了计算平均绩点，规定A、B、C、D、F分别代表4分、3分、2分、1分、0分。

输入格式

有多组测试样例。每组输入数据占一行，由一个或多个大写字母组成，字母之间由空格分隔。

输出格式

每组输出结果占一行。如果输入的大写字母都在集合｛A,B,C,D,F｝中，则输出对应的平均绩点，结果保留两位小数。否则，输出“Unknown”。

输入数据

A B C D F

B F F C C A

D C E F

输出数据

2.00

1.83

Unknown

#include <stdio.h>

float **GPA**(char *grade*) {

    switch (*grade*) {

        case 'A':return 4.00;

        case 'B':return 3.00;

        case 'C':return 2.00;

        case 'D':return 1.00;

        case 'F':return 0.00;

        default:return -1.00;*//* **非法的成绩等级**

    }

}

int **main**() {

    char grades[100];*//* **假设一行最多有100个大写字母**

    while (**gets**(grades)) {*//* **循环读取每组输入数据**

        float sum = 0;int count = 0, i = 0;

        while (grades[i] != '\0') {*//* **遍历输入的字符数组**

            if (grades[i] != ' ') {*//* **跳过空格**

                float gpa = **GPA**(grades[i]);*//* **计算绩点**

                if (gpa == -1.00) {*//* **非法的成绩等级**

**printf**("Unknown\n");

                    break;*//* **结束当前组的循环**

                }

                sum += gpa;*//* **累加绩点**

                count++;*//* **成绩科目数量+1**

            }

            i++;

        }

        if (grades[i] == '\0') {*//* **当前组的所有字符都已处理完毕**

            float aver = sum / count;*//* **计算平均绩点**

**printf**("%.2f\n", aver);

        }

    }

}

* 手机话费

小明的手机每天消费1元，每消费K元就可以获赠1元，一开始小明有M元，问最多可以用多少天？

输入格式

包括多个测试实例。每个测试实例包括2个整数M，K（2<=k<=M<=1000)。M=0，K=0代表输入结束。

输出格式

对于每个测试实例输出一个整数，表示M元可以用的天数。

输入数据 1

2 2

4 3

0 0

输出数据 1

3

5

#include <stdio.h>

int **main**(){

    int M,K;

    while(1){

**scanf**("%d %d", &M, &K);

        if (M==0 && K==0) break;

        int days=M,rest=M; // 初始天数、初始余额均赋值为M

        while(rest>=K){

            int temp=rest/K;

            rest=rest-temp\*K+temp;

            days+=temp;

        }

**printf**("%d\n",days);

    }

}

* 螺旋数组

输入样例：

5

输出样例：

1 16 15 14 13

2 17 24 23 12

3 18 25 22 11

4 19 20 21 10

5 6 7 8 9

#include <stdio.h>

int **main**()

{

     int n;

**scanf**("%d",&n);

     int a[100][100]={0};

     int t=1,direct=0;

     int i,j,count=n\*n;

     for(i=0;i<n+2;i++) a[i][n]=1;

     for(j=0;j<n+1;j++) a[0][j]=a[n+1][j]=1;

     i=1;j=0;

     while(1){

        a[i][j]=t;

        t++; count--;

        if(count==0) break;

        switch(direct){*//* **0:向下；1:向右；2:向上；3:向左**

            case 0: i++;

                if(a[i][j]!=0){i--;direct=1;t--;count++;}

                break;

            case 1:j++;

                 if(a[i][j]!=0){j--;direct=2;t--;count++;}

                 break;

            case 2:i--;

                 if(a[i][j]!=0){i++;direct=3;t--;count++;}

                 break;

            case 3:j--;

                 if(a[i][j]!=0){j++;direct=0;t--;count++;}

                 break;

        }

     }

    for(i=1;i<=n;i++){

        for(j=0;j<n;j++) **printf**("%3d ",a[i][j]);

**printf**("\n");

    }

}

* 查找字符串

输入一个正整数repeat (0<repeat<10)，做repeat次下列运算：

输入一个字符，再输入一个以回车结束的字符串（少于80个字符），在字符串中查找该字符，如果找到，输出该字符在字符串中所对应的最大下标 (下标从0开始)，输出格式为"index = %d\n"；否则输出"Not Found"。

输入样例:

2 (repeat=2)

m (字符'm')

programming (字符串"programming")

a (字符'a')

1234 (字符串"1234")

输出样例:

index = 7 ('m'在"programming"中对应的最大下标是7)

Not Found ("1234"中没有'a')

#include <stdio.h>

int **main**(){

    char p;

    int re,pos;

**scanf**("%d",&re);

**getchar**(); // 消耗掉scanf读取后面的回车符

    for(int j=0;j<re;j++){

**scanf**(" %c",&p); // 详见下方解释

        char s[81]; // 数组定义放到循环内部，避免每次迭代覆盖

**scanf**("%s",s); // 不带&，因为数组名本身就是指向数组首元素的指针

        pos=-1;

        for(int i=0;s[i]!='\0';i++){

            if(s[i]==p) pos=i;

        }

        if(pos==-1) **printf**("Not Found\n");

        else **printf**("index = %d\n",pos);

    }

}

scanf(" %c",&p); // 在格式控制字符串中加入空格，消耗掉换行符

在C语言中，scanf函数用于从标准输入（通常是键盘）读取输入，并根据指定的格式进行格式化输入。在这里，空格字符" "在格式字符串中的作用是跳过输入中的空白字符（空格、制表符、换行符等），直到遇到第一个非空白字符为止。

例如，如果格式字符串是" %c"，那么在输入时，如果用户在输入之前输入了一个或多个空格、制表

符或换行符，这些空白字符会被忽略，直到用户输入了一个实际的字符为止。这样可以让程序在输入时更加灵活，不会因为用户的输入习惯而出现问题。

另外，如果不在格式字符串中包含空格字符，那么scanf函数在读取字符时可能会读取到之前的换行符或空格，这通常不是我们想要的行为。

总的来说，将空格字符" "包含在scanf函数的格式字符串中，可以帮助程序忽略输入中的空白字符，从而更准确地读取用户的输入。

* 查找K位素数

从任一给定的长度为L的数字中，找出最早出现的K位连续数字所组成的素数。

输入格式：

在第一行给出2个正整数，分别是L（≤1000）和K（<10）。

接下来一行给出一个长度为L的正整数N。

输出格式：

在一行中输出N中最早出现的K位连续数字所组成的素数。若不存在，则输出404。

注意，原始数字中的前导零也计算在位数之内。例如在200236中找4位素数，0023算是解；但第一位2不能被当成0002输出，因为在原始数字中不存在这个2的前导零。

输入样例 1：

20 5

23654987725541023819

输出样例 1：

49877

输入样例 2：

10 3

2468001680

输出样例 2：

404

#include <stdio.h>

#include <stdbool.h>

#include <string.h>

int **ss**(int *num*) { // 判断素数函数，不再多提

    if(*num*<2) return 0;

    for(int i=2;i\*i<=*num*;i++) if(*num*%i==0) return 0;

    return 1;}

void **find**(int *l*,int *k*,char\* *n*){

    for (int i=0;i<=*l*-*k*;i++){

        char str[*k*+1];

**strncpy**(str,&*n*[i],*k*); // 详见下方解释

        str[*k*]='\0';

        int num=**atoi**(str); // 详见下方解释

        if(**ss**(num)){

**printf**("%s\n",str);

            return;} // 提前结束函数的执行（减少超时风险）

    }

**printf**("404\n");

}

int **main**(){

int l,k;**scanf**("%d %d",&l,&k);

char n[1001];**scanf**("%s",n);**find**(l,k,n);

}

* strncpy函数：将n字符串中以第i个字符开始的k个字符复制到str字符串中。参数str是目标字符串，&n[i]表示n字符串中从第i个字符开始的地址，k表示要复制的字符个数。

【需包含头文件】<string.h>

【注意】strncpy函数会复制k个字符到目标字符串中，并且如果源字符串长度小于k，那么会在目标字符串中补充'\0'直到复制了k个字符为止。如果源字符串长度大于k，那么只会复制前k个字符到目标字符串中。

* atoi函数：将一个字符串转换为整数。具体来说，atoi函数会将输入的字符串中的数字部分转换为整数，忽略前导的空格，并遇到非数字字符或者字符串结束标志'\0'时停止转换。如果无法转换，或者字符串为空，atoi函数会返回0。

【需包含头文件】<stdlib.h>