代码示例3：

闰年是公历中的名词。闰年分为普通闰年和世纪闰年。

普通闰年：公历年份是4的倍数的，且不是100的倍数，为普通闰年（如2004年、2020年是闰年，1900年不是闰年）。

世纪闰年：公历年份是整百数的，必须是400的倍数才是世纪闰年（如1900年不是世纪闰年，2000年是世纪闰年）。

输入一组表示年份的整数，编写一个程序，输出与每个年份相邻最近的闰年年份，并输出其相距的年数。如果两侧有相距相同时间长度的闰年，则输出较早的年份及相应距离。

如果输入的年份本身就是闰年，则输出该年份本身，距离输出0

第一行给出测试数据组数N(0<N<=100)

接下来的N行每行有一个年份M(0<M<=2020)，

每行输出两个整数 A B.

其中A表示离相应测试数据最近的闰年年份，B表示其间相距的年数。

样例输入

3

2000

2006

1901

样例输出

2000 0

2004 2

1904 3

int N,M,M1,M2;

cin>>N;

int i;

for(i=0;i<N;i++) //用N控制测试组数，每次循环处理一个年份

{

cin>>M; //输入年份

/\*

思路：

先判断M是否是闰年，如果是，直接输出；

若不是——

从这一年开始分别向前、向后找出距离M最近的闰年M1和M2,并求出年份的差距；

如果M-M1<M2-M,则M1为所求，否则M2为所求

\*/

if( (M%4==0 && M%100!=0) || M%400==0 ) //如果M是闰年，直接输出年份，年份差为0

{

cout<<M<<" "<<0<<endl;

}

else //M不是闰年

{ M1=M2=M;

while( (M1%400!=0) && (M1%4!=0 || M1%100==0) ) //注意循环的条件为M1不是闰年

{

//从M向前找最近的闰年M1，for改while

M1--;

}

//这时M1已经是从M向前找最近的闰年

while( (M2%400!=0) && (M2%4!=0 || M2%100==0) )

{

//从M向后找最近的闰年M2

M2++;

}

//这时M2已经是从M向后找最近的闰年

if( M-M1<=M2-M ) //减法优先级高于<,细节——小于等于！！！

{

cout<<M1<<" "<<M-M1<<endl;

}

else

{

cout<<M2<<" "<<M2-M<<endl;

}

}

}

代码示例2：

根据给定的数输出空心正方形和“漏斗形”。

要求：输入一个数n（3<=n<=10），这个n代表空心正方形的边长（即“\*”的个数），也代表“漏斗形”最长的一行的长度（也是“\*”的个数），每两个“\*”之间空一格。

样例输入：5

样例输出：

\* \* \* \* \*

\* \*

\* \*

\* \*

\* \* \* \* \*

\* \* \* \* \*

\* \* \*

\*

\* \* \*

\* \* \* \* \*

int n;

cin>>n;

int i,j;

for(i=1;i<=2\*n;i++) //每个循环体为对第i行图形的处理

{

if( i==1 || i==n || i==n+1 || i==2\*n ) //最直接简单的几行

{

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

{

cout<<"\* ";

}

}

if( i>1 && i<n )

{

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

{

if( j==1 || j==n )

{ cout<<"\* "; }

else

{ cout<<" "; }

}

}

if( i>n+1 && i<2\*n )

{

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

{ if(i<=(n+1+2\*n)/2)

{

if( j>(i-n-1) && j<=(n-(i-n-1)) )

{ cout<<"\* "; }

else

{ cout<<" "; }

}

else

{

if( j>(2\*n-i) && j<=(n-(2\*n-i)) )

{ cout<<"\* "; }

else

{ cout<<" "; }

}

}

}

cout<<endl;

}