代码示例1：

输入两个正整数a，b，求出它们的最小公倍数和最大公约数

分析：

最小公倍数：

找出a、b中较大的数max和较小的数min，然后依次验证max的1倍、2倍、3倍......直到找到一个数c，使得c可以被min除尽，则c就是a和b的最大公倍数

最大公约数：

辗转相除法（欧几里得算法）

设g(a,b)为a和b的最大公约数，则有：

if(a%b=0) {g(a,b)=b;}

else {g(a,b)=g(b,a%b);}

也就是说当a%b的结果不是0时，由于g(a,b)=g(b,a%b)，也就是以b代a、再以a%b代b，继续判断a%b是否等于0。

int a,b;

cin>>a>>b;

int max,min;

if(a>b)

{

max=a;

min=b;

}

else

{

max=b;

min=a;

} //将两个数中的较大的赋值给max，较小者赋值给min

//求最小公倍数

int i,gbs=max;

for(i=2;gbs%min;i++)

{

gbs=max\*i;

}

cout<<a<<"和"<<b<<"的最小公倍数为："<<gbs<<endl;

cout<<a<<"和"<<b<<"的最大公约数为："; //最后再写，因为ab值发生了变化

//求最大公约数

int tmp;

while(a%b)

{

tmp=a;

a=b;

b=tmp%b;

}

cout<<b<<endl; //最大公约数

代码示例2：

设n是一任意自然数。若将n的各位数字反向排列所得自然数n1与n相等，则称n为一**回文数**。例如，若n=1234321，则称n为一回文数；但若n=1234567，则n不是回文数。

**平方回数**：一个回文数，它同时还是某一个数的平方，这样的数字叫做**平方回数**。例如：121

设计一个程序，按照**从大到小**的顺序依次输出100-999之间（含100和999）的所有平方回数

int i,j;

int a,b; //分别代表个位数和百位数

for(i=999;i>=100;i--)

{

/\*

三位数是回文数的条件：个位数与百位数相同

\*/

a=i%10;

b=i/100;

if(a==b) //是回文数，接下来判断是不是平方回数

{

//cout<<i<<" ";

for(j=10;j<32;j++) //10^2=100;32^2=1024

{

if(i==j\*j) //是平方回数，\*的优先级高于==，不用加括号

{cout<<i<<" ";} //输出平方回数

}

}

}

cout<<endl;

类比练习：输出0-1000当中所有的回文数

代码示例3：

输入正整数N（N<1000），输出N以内的全部质数,并输出N以内质数的个数。

int N,i,j,isprime=1,count; //应在每次循环开始前重置isprime的值,count要初始化为0

cin>>N;

for(i=2;i<=N;i++)

{

//isprime=1;

//sqrt()只支持浮点数，使用CSDN查询

for(j=i/2;j>1;j--) //j=sqrt(float(i))

{

if(i%j==0)

{ isprime=0; }

}

if(isprime/\*=1\*/)

{

cout<<i<<" ";

count++;

}

}

cout<<endl;

cout<<N<<"以内的质数有"<<count<<"个"<<endl;

return 0;