求两个整数的最大公约数、最小公倍数

欧几里德算法

最大公约数（非递归算法）

问题描述：

从键盘输入两个正整数 a 和 b，求 a 和 b 的最大公约数和最小公倍数。

问题分析：

使用辗转相除法来求两个正整数的最大公约数步骤如下：

(1) 以大数 a 作为被除数，小数 b 作为除数，r 保存 a 除以 b 的余数；

(2) 如果 r=0，则 b 就是最大公约数。否则辗转赋值：a=b； b=r；重复(1)；

(3) 最小公倍数=a*b\r

程序清单：

```
int gcd(int a, int b)    // 求 a 和 b 的最大公约数
{
  int r;
  r=a%b;
  while(r!=0)
  { a=b; b=r; r=a%b; }
  return b;
}
main()
{
  int a,b,k;
  printf("Input a,b:");
  scanf("%d%d",&a,&b);
  k=gcd(a,b);
  printf("M=%d   N=%d\n",k,a*b/k);
}
```

最大公约数（递归算法）

问题描述：

从键盘输入两个正整数 A 和 B，求 A 和 B 的最大公约数。

问题分析：

数学上求最大公因子(Great Common Divisor)的运算时常使用辗转相除法，反复计算到余数为零为止，这种方法也称为欧几里得定理(Euclid’s Algorithm)，其定义为：

$$GCD(A,B)=\begin{cases}A & B=0\\ GCD(B,A\%B) & B>0\end{cases}$$

我们可以运用递归来设计求最大公因子的程序。

当 B 等于 0，返回最大公因子为 A。

当 N 大于 0 时，返回最大公因子为 GCD(B,A%B)。

例如：求 GCD(3155,2545)则：

GCD(3155,2545)=GCD(2545,3155%2545)=GCD(2545,610)

=GCD(610,2545%610)=GCD(610,105)

=GCD(105,610%105)=GCD(105,85)

=GCD(85,105%85)=GCD(85,20)

=GCD(20,85%20)=GCD(20,5)

=GCD(5,20%5)=GCD(5,0)

=5

程序清单：

```
#include "stdio.h"
int gcd(int a,int b)
{
  if(b==0)
    return a;
  else
    return gcd(b,a%b);
}
main()
{
  int a,b,ans;
  scanf("%d%d",&a,&b);
  ans=gcd(a,b);
  printf("ans=%d\n",ans);
}
```

最小公倍数

问题描述：

从键盘输入两个正整数 A 和 B，求 A 和 B 的最小公倍数。

程序清单：

```
#include "stdio.h"
```

```
int lcm(int a,int b)
{
  int t;
  if(a<b)
  { t=a; a=b; b=t; }
  t=a;
  while(t%b>0) t=t+a;
  return t;
}
main()
{
  int a,b,ans;
  scanf("%d%d",&a,&b);
  ans=lcm(a,b);
  printf("ans=%d\n",ans);
}
```

Stein 算法

欧几里德算法是计算两个数最大公约数的传统算法，他无论从理论还是从效率上都是很好的。但是他有一个致命的缺陷，这个缺陷只有在大素数时才会显现出来。

考虑现在的硬件平台，一般整数最多也就是 64 位，对于这样的整数，计算两个数之间的模是很简单的。对于字长为 32 位的平台，计算两个不超过 32 位的整数的模，只需要一个指令周期，而计算 64 位以下的整数模，也不过几个周期而已。但是对于更大的素数，这样的计算过程就不得不由用户来设计，为了计算两个超过 64 位的整数的模，用户也许不得不采用类似于多位数除法手算过程中的试商法，这个过程不但复杂，而且消耗了很多 CPU 时间。对于现代密码算法，要求计算 128 位以上的素数的情况比比皆是，设计这样的程序迫切希望能够抛弃除法和取模。

Stein 算法由 J.Stein 1961 年提出，这个方法也是计算两个数的最大公约数。和欧几里德算法 算法不同的是，Stein 算法只有整数的移位和加减法，这对于程序设计者是一个福音。

为了说明 Stein 算法的正确性，首先必须注意到以下结论：

gcd(a,a) = a，也就是一个数和他自身的公约数是其自身

gcd(ka,kb) = k gcd(a,b)，也就是最大公约数运算和倍乘运算可以交换，特殊的，当 k=2 时，说明两个偶数的最大公约数必然能被 2 整除。

程序清单：

```
int gcd(int a,int b)
{
  int temp;
  if(a<b) { temp = a;  a = b;  b=temp; }
  if(b==0) return a;
  if(a%2==0 && b%2 ==0) return 2*gcd(a/2,b/2);
```

```
  if( a%2 == 0) return gcd(a/2,b);
  if( b%2==0 ) return gcd(a,b/2);
  return gcd((a+b)/2,(a-b)/2);
}

main()
{
  int a,b,k;
  printf("Input a,b:");
  scanf("%d%d",&a,&b);
  k=gcd(a,b);
  printf("M=%d   N=%d\n",k,a*b/k);
  getch();
}
```