欧几里德算法

1.欧几里德算法的思想：

欧几里德算法的思想基于辗转相除法的原理，辗转相除法是欧几里德算法的核心思想，欧几里德算法说白了其实就是辗转相除法的计算机算法的实现而已。下面我们先说说辗转相除法，辗转相除法的内容：如果用 gcd(a,b)来表示 a 和 b 的最大公约数，那么根据辗转相除法的原理，有 gcd(a,b)=gcd(b,a mod (b))，其中 mod()表示模运算，并且不妨让 a>b,这样方便于模运算。

2.辗转相除法的正确性 gcd(a,b)=gcd(b,a mod (b))的证明：

第一步：令 c 为 a 和 b 的最大公约数，数学符号表示为 c=gcd(a,b).因为任何两个实数的最大公约数 c 一定是存在的，也就是说必然存在两个数 k1,k2 使得 a=k1.c, b=k2.c

第二步：a mod (b)等价于存在整数 r,k3 使得余数 r=a － k3.b.

即 r = a － k3.b

= k1.c － k3.k2.c

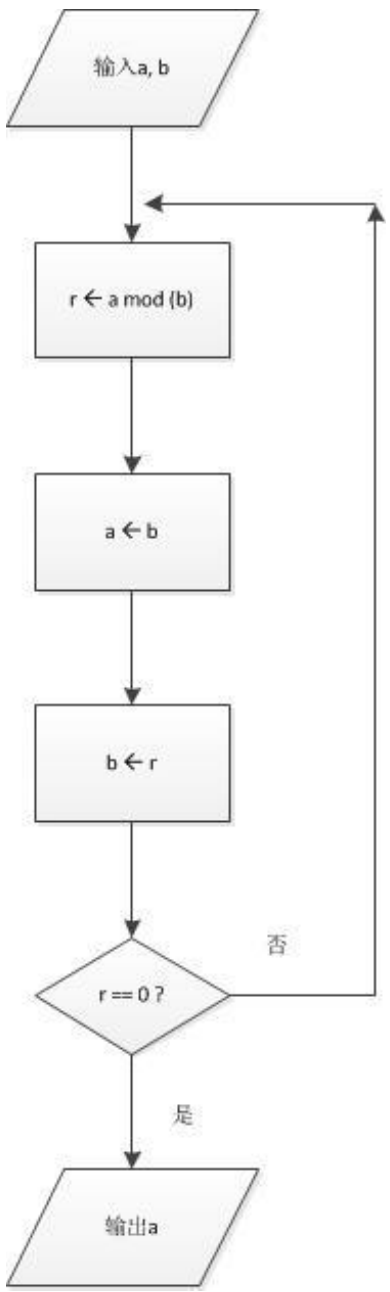
= (k1 － k3.k2).c

显然，a 和 b 的余数 r 是最大公因数 c 的倍数。

3.欧几里德算法的优点：

通过模运算的余数是最大公约数之间存在的整数倍的关系，来给比较大的数字进行降维，方便手算；同时，也避免了在可行区间内进行全局的最大公约数的判断测试，只需要选取其余数进行相应的计算就可以直接得到最大公约数，大大提高了运算效率。

4.欧几里德算法流程图：



5.欧几里德算法的 C 语言实现：

```
//功能：利用欧几里德算法，求整数 a，b 的最大公约数
//参数：整数 a，b
//返回：a,b 的最大公约数
int gcd(int a, int b){
if(a < b){ //保证 a 大于等于 b，便于 a%b 的运算
int temp;
temp = a;
a = b;
b = temp;
}
while(a % b){ //如果余数不为 0，就一直进行辗转相除
int r = a % b; //r 为 a 和 b 的余数，即 r = a mod(b);
a = b;
b = r;
r = a % b;
}
return b;
}

//测试函数
#include
int main(){
printf << gcd(4,12) << endl;
}
```