蒙哥马利模乘的优点在于减少了取模的次数（在大数的条件下）以及简化了除法的复杂度（在2的k次幂的进制下除法仅需要进行左移操作）。模幂运算是RSA 的核心算法，最直接地决定了RSA 算法的性能。

针对快速模幂运算这一课题，西方现代数学家提出了大量的解决方案，通常都是先将幂模运算转化为乘模运算。

例如求D=C^15%N

由于：a\*b % n = (a % n)\*(b % n) % n

所以令：

C1 =C\*C % N =C^2 % N

C2 =C1\*C % N =C^3 % N

C3 =C2\*C2 % N =C^6 % N

C4 =C3\*C % N =C^7 % N

C5 =C4\*C4 % N =C^14 % N

C6 =C5\*C % N =C^15 % N

即：对于E=15的幂模运算可分解为6 个乘模运算，归纳分析以上方法可以发现：

对于任意指数E，都可采用以下算法计算D=C\*\*E % N：

D=1

WHILE E>0

IF E%2=0

C=C\*C % N

E=E/2

ELSE

D=D\*C % N

E=E-1

RETURN D

继续分析会发现，要知道E 何时能整除 2，并不需要反复进行减一或除二的操作，只需验证E 的二进制各位是0 还是1 就可以了，从左至右或从右至左验证都可以，从左至右会更简洁，

设E=Sum[i=0 to n](E\*2\*\*i)，0<=E<=1

则：

D=1

FOR i=n TO 0

D=D\*D % N

IF E=1

D=D\*C % N

RETURN D这样，模幂运算就转化成了一系列的模乘运算。

C++实现编辑

inline unsigned \_\_int64 MulMod(unsigned \_\_int64 a, unsigned \_\_int64 b, unsigned \_\_int64 n)

{

return a \* b % n;

}

/\*

模幂运算，返回值 x=base^pow mod n

\*/

unsigned \_\_int64 PowMod(unsigned \_\_int64 base, unsigned \_\_int64 pow, unsigned \_\_int64 &n)

{

unsigned \_\_int64 a=base, b=pow, c=1;

while(b)

{

while(!(b & 1))

{

b>>=1; //a=a \* a % n; //

a=MulMod(a, a, n);

} b--; //c=a \* c % n; //

c=MulMod(a, c, n);

} return c;

}