# 第2次编程练习报告

姓名：胡博浩 学号：2212998 班级：信息安全

##### 编程练习1——平方-乘算法

* **源码部分：**

#include<iostream>

using namespace std;

long long square\_multiply(long long a, long long n, long long m) {

long long c = 1;

a %= m;//防止输入的a过大

while (n) {

if (n & 1) {

c = (c \* a) % m;//如果指数是奇数，将c乘以底数并取模

}

a = (a \* a) % m;//将底数平方并取模

n >>= 1;

}

return c;

}

int main() {

cout << "Calculate a^n(mod m)..." << endl;

cout << "Please input:" << endl;

long long a, n, m;

cout << " a=";cin >> a;

cout << " n=";cin >> n;

cout << " m=";cin >> m;

cout << a << "^" << n << "(mod " << m << ")=" << square\_multiply(a, n, m) << endl;

system("pause");

return 0;

}

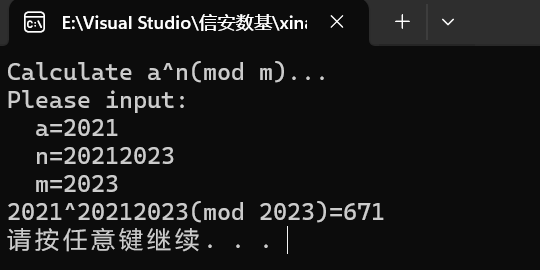
* **说明部分：**

在代码中，函数square\_multiply采用非递归方式实现平方乘算法，从低位到高位计算，输入参数包括底数a、指数n和模数m。

该算法利用了模幂运算的特性，将n表示为二进制形式，所以算法的时间复杂度较低，为O(log n)。

此函数进行了几次优化：一开始检查a，防止a过大；使用位运算。从而使算法的性能优越。

* **运行示例：**



##### 二、编程练习2——扩展的欧几里得算法求逆元

* **源码部分：**

#include<iostream>

using namespace std;

int gcd(int a, int b) {

if (b==0) {

return a;

}

else {

return gcd(b, a % b);

}

}

void find(int a, int m,int& s) {//m是模数，s是a的逆元

int s0 = 1, t0 = 0, s1 = 0, t1 = 1;

int m0 = m, t;//m0存储模数

int r = a % m;

int q = (a - r) / m;

while (r) {

s = s0 - q \* s1;

t = t0 - q \* t1;

s0 = s1;

t0 = t1;

s1 = s;

t1 = t;

a = m;

m = r;

r = a % m;

q = (a - r) / m;

}

s = s1;

if (s < 0) {//将x调整为正数

s += m0;

}

}

int main() {

int a, b, a0, b0;

cout << "a="; cin >> a;

cout << "b="; cin >> b;

int g = gcd(a, b);

find(a, b, a0);

find(b, a, b0);

cout << "gcd(a,b)=" << g << endl;

cout << "lcm(a,b)=" << a\*b/g << endl;

cout << "a^(-1)=" << a0 << "(mod " << b << ")" << endl;

cout << "b^(-1)=" << b0 <<"(mod " << a << ")" << endl;

system("pause");

return 0;

}

* **说明部分：**

gcd和lcm求解与之前的相同。函数find的参数a表示待求逆元的数，m表示模数，x表示逆元。

在函数内部，通过s0、t0、s1、t1等变量记录中间计算结果，不断循环求出逆元，并将结果存储在参数s中。如果s为负数，则将其加上模数。

* **运行示例：**

