第 2 次编程练习报告

一、编程练习 1——平方-乘算法

➢ 源码部分：

```cpp
#include <iostream>
using namespace std;

long long square_multiply(long long base, long long exponent, long long modulus) {
    long long result = 1;  // 定义结果变量并初始化为1
    while (exponent > 0) { // 当指数大于0时
        if (exponent % 2 == 1) { // 如果指数是奇数
            result = (result * base) % modulus; // 将结果乘上底数并对模数取余
        }
        base = (base * base) % modulus; // 将底数平方并对模数取余
        exponent = exponent / 2; // 将指数除以2
    }
    return result; // 返回结果
}

int main() {
    cout << "Calculate a^n(mod m)" << endl;
    cout << "Please input:" << endl;
    long long base = 2, exponent = 10, modulus = 1000000007;
    cout << "a=";
    cin >> base;
    cout <<  "n=";
    cin >> exponent;
    cout <<  "m=";
    cin >> modulus;
    // 定义底数、指数和模数，并假设底数和指数都是正整数
    long long result = square_multiply(base, exponent, modulus); // 调用平方-乘算法
    cout << base << "^" << exponent << "(mod" << modulus << ")=" << result;// 输出结果
    system("pause");
    return 0;
}
```

➢ 说明部分：

这个程序中的 pow 函数使用平方-乘算法来实现幂运算，其中参数 base 是底数，参数 exp 是指数，参数 mod 是模数。函数的返回值是 base 的 exp 次方模 mod 的结果。算法的时

间复杂度为 O(log(exp))。

- **运行示例：**

```
Calculate a^n(mod m)
Please input:
a=2021
n=20212023
m=2023
2021^20212023(mod2023)=671请按任意键继续. . .
```

二、编程练习 2——计扩展的欧几里得算法求逆元

源码部分：

```
#include <iostream>
using namespace std;

// 扩展的欧几里得算法求逆元
// a: 待求逆元的数，m: 模数，x: 逆元
// 返回值: 是否存在逆元
bool extGcd(int a, int m, int& x)
{
    int x1, y1, x0, y0, y;
    x0 = 1;
    y0 = 0;
    x1 = 0;
    y1 = 1;
    int m0 = m;
    int r = a % m;
    int q = (a - r) / m;
    while (r)
    {
        x = x0 - q * x1;
        y = y0 - q * y1;
        x0 = x1;
        y0 = y1;
```

```
        x1 = x;
        y1 = y;
        a = m;
        m = r;
        r = a % m;
        q = (a - r) / m;
    }
    if (m != 1) // a和m不互质，不存在逆元
    {
        return false;
    }
    x = x1;
    if (x < 0) // 将x调整到[0, m)范围内
    {
        x += m0;
    }
    return true;
}
int gcd(int x, int y)
{
    int z = y;
    while (x % y != 0)
    {
        z = x % y;
        x = y;
        y = z;
    }
    return z;
}
int lcm(int a, int b)
{
    return (a * b) / gcd(a, b);
}

int main()
{
    int n, m;
    cout << "a=";
    cin >> n;
    cout << "b=";
    cin >> m;
    cout << "gcd(a,b)=" << gcd(n, m) << endl;
    cout << "lcm(a,b)=" << lcm(n, m) << endl;
    int x;
```

```
    if (extGcd(n, m, x))
    {
        cout << "a^(-1)=" << x << " (mod " << m << ")" << endl;
    }
    else
    {
        cout << "不存在逆元" << endl;
    }
    if (extGcd(m, n, x))
    {
        cout << "b^(-1)=" << x << " (mod " << n << ")" << endl;
    }
    else
    {
        cout << "不存在逆元" << endl;
    }
    system("pause");
    return 0;
}
```

➢ 说明部分：

此代码利用上周编程作业二中部分代码进行 gcd 与 lcm 的求解。函数 extGcd 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 x0、y0、x1、y1 等变量记录了中间计算结果，最终求出了逆元，并将结果存储在参数 x 中。函数返回值为 bool 类型，表示是否存在逆元。若存在逆元，则返回 true，否则返回 false。

➢ 运行示例：