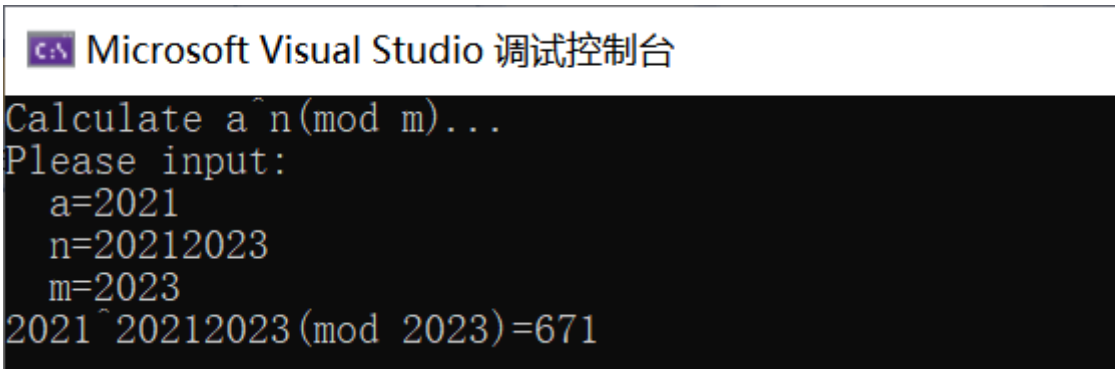
而$4(n+1)!+n(n+1)(n+4)\equiv 2n(n+1)\equiv 4(mod\ n+2)\ \Rightarrow\ n=2$矛盾，假设不成立，$n+2$必为素数.

综上，证毕.

编程练习（基于C/C++）

1. 编程实现平方-乘算法，效果如图所示.




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

int pow_mod(int a, int n, int m)
{
    int rst = 1;
    while (n > 0)
    {
        if (n & 1)
        {
            rst *= a;
            rst %= m;
        }
        a *= a;
        a %= m;
        n >>= 1;
    }
    return rst;
}

int main()
{
    cout << "Calculate a^n(mod m)..." << endl;
    cout << "Please input:" << endl;
    int a, n, m;
    cout << "  a="; cin >> a;
    cout << "  n="; cin >> n;
    cout << "  m="; cin >> m;
    cout << a << "^" << n << "(mod " << m << ")=" << pow_mod(a, n, m) << endl;
    return 0;
}
```

2. 编程实现扩展的欧几里得算法求逆元，效果如图所示.


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

void swap(int& a, int& b)
{
    a = a ^ b;
    b = a ^ b;
    a = a ^ b;
}

int extend_Euclid(int a, int b,int&inv_a,int&inv_b)
{
    if (a < b)return extend_Euclid(b, a, inv_b, inv_a);
    int a0 = a, b0 = b, q = 1;
    int s0 = 1, s1 = 0, t0 = 0, t1 = 1;
    while (a % b != 0)
    {
        q = a / b;
        a = a % b;
        swap(a, b);
        s0 -= q * s1;
        swap(s0, s1);
        t0 -= q * t1;
        swap(t0, t1);
    }
    inv_a = s1 > 0 ? s1 : s1 + b0;
    inv_b = t1 > 0 ? t1 : t1 + a0;
    return b;
}

int main()
{
    int a, b, inv_a, inv_b;
    cout << "a=";
    cin >> a;
    cout << "b=";
    cin >> b;
    int gcd = extend_Euclid(a, b, inv_a, inv_b);
    int lcm = a * b / gcd;
    cout << "gcd(a,b)=" << gcd << endl;
    cout << "lcm(a,b)=" << lcm << endl;
    cout << "a^(-1)=" << inv_a << "(mod " << b << ")" << endl;
    cout << "b^(-1)=" << inv_b << "(mod " << a << ")" << endl;
}
```