# 椭圆曲线编程练习报告

姓名：周钰宸 学号：2111408 班级：信安一班

##### **编程练习——实现基本的Z\_p上的椭圆曲线E\_p(a,b)的计算：**

* **源码部分：**

//第五次作业：实现基本的模P剩余类上椭圆曲线的计算

#include<iostream>

#include<stack>

#include<cmath>

#include<vector>

using namespace std;

int p, a, b;//全局变量曲线的参数p、a、b

bool flag;//定义功能一：是否为椭圆曲线的，若不是重新输入的标志位

class Point {//定义所需的点类

public:

    int x, y;

    Point(int x1, int y1) :x(x1), y(y1) {};

    Point() {};

};

vector<Point> AllPoints;//存储椭圆曲线上的所有点

bool if\_vegetable(int n) {//判断是否为素数,辅助使用计算欧拉函数,以及用来各种地方

    if (n <= 1)

        return false;

    else {

        for (int i = 2; i < n; i++) {

            if (n % i == 0) {

                return false;

            }

        }

        return true;

    }

}

stack<int> DecimalToBinary(unsigned  long long n) {//计算十进制转二进制，用一个栈s来保存，用来辅助使用平方乘算法

    stack<int> s;

    while (n != 0) {

        int r = n % 2;

        s.push(r);

        n = n / 2;

    }

    return s;

}

unsigned long long  SquareAndMultiplyAlgorithm(unsigned long long a, unsigned long long n, unsigned long long m) {//平方乘算法

    unsigned long long result = 1;//result定义为64位的整型,防止中间结果超过32位而发生溢出

    stack<int> s = DecimalToBinary(n);

    while (!s.empty()) {//由于算法的特性，先要处理的是二进制的头位，即在二进制转为十进制方法中最后得到的余数

        //符合FILO的特性，故采用栈的形式存储并操作

        result = (result \* result) % m;

        if (s.top() == 1) {

            result = (result \* a) % m;

        }

        s.pop();

    }

    return result;

}

int ExtendedEuclideanAlgorithm(int a, int b, int& x, int& y) {//扩展的欧几里得算法求逆元，用于求逆元

    if (!b) {

        x = 1;

        y = 0;

        return a;

    }

    int ret = ExtendedEuclideanAlgorithm(b, a % b, x, y);//递归结束返回时候得到的x和y是下一个状态的，由推导的公式决定此时的x和y

    int temp = x;

    x = y;

    y = temp - a / b \* (y);

    return ret;

}

int mod(int a, int p) {//重载取模运算mod

    if (a % p >= 0)

        return a % p;

    else

        return a % p + p;

}

int LegendreSymbol(int a, int p) {//计算勒让德符号

    return mod(pow(a, (p - 1) / 2), p);//由于p一般较小，所以直接采用欧拉判别式法求解

}

bool IfIsInfinitePoint\_O(Point P) {//判断是否为无穷远点，设置无穷远点O的x和y均为特殊值INT\_MAX

    return P.x == INT\_MAX && P.y == INT\_MAX ? true : false;

}

void If\_EllipticCurve() {//功能一：给定参数p，a，b，判断E\_p(a,b)是否为椭圆曲线

    if (p <= 3 || !if\_vegetable(p)) {//若p不为大于3的素数

        cout << "E\_" << p << "(" << a << "," << b << ") is not Elliptic Curve." << endl;

        flag = false;

        return;

    }

    int temp = 4 \* pow(a, 3) + 27 \* pow(b, 2);

    temp = mod(temp, p);

    if (!temp) {//判断该曲线是否为光滑曲线

        cout << "E\_" << p << "(" << a << "," << b << ") is not Elliptic Curve." << endl;

        flag = false;//若不是椭圆曲线，需要重新在main函数中输入s

        return;

    }

    cout << "E\_" << p << "(" << a << "," << b << ") is Elliptic Curve." << endl;

    return;

}

bool IfIsOnEllipticCurve(Point P) {//功能二：判断给定的点P，Q是否在椭圆曲线E\_p(a,b)上

    if (IfIsInfinitePoint\_O(P)) //若P为无穷远点，则一定在椭圆曲线上

        return true;

    int y\_2 = mod(pow(P.x, 3) + a \* P.x + b, p);//由椭圆方程求出y^2

    int y1 = 0, y2 = 0;//记录求出的两个二次同余方程的解

    if (LegendreSymbol(y\_2, p) == -1)//若勒让德符号为-1，一定无解，一定不在椭圆曲线上

        return false;

    else {

        for (int i = 1; i <= p - 1; i++) {//由于当p一般较小时，可以直接遍历求解二次同余方程

            if (mod(pow(i, 2), p) == y\_2) {

                y1 = i;

                y2 = p - y1;//另一解与y1相加为p

                break;

            }

        }

        if (mod(P.y, p) != y1 && mod(P.y, p) != y2)//若坐标的y均不为两个同余方程解，则一定不在椭圆曲线上

            return false;

        else

            return true;

    }

}

Point AddTwoPoints(Point P, Point Q) {//功能三：对在椭圆曲线E\_p(a,b)上的两点P,Q，计算P+Q

    Point RP;

    if (IfIsInfinitePoint\_O(P)) {//若其中P为无穷远点O

        if (IfIsInfinitePoint\_O(Q)) {//若Q也为无穷远点O

            RP.x = RP.y = INT\_MAX;

            return RP;

        }

        else {//若Q不为无穷远点O，则结果为Q

            RP.x = Q.x, RP.y = Q.y;

            return RP;

        }

    }

    if (IfIsInfinitePoint\_O(Q)) {//若其中Q为无穷远点O

        if (IfIsInfinitePoint\_O(Q)) {//若P也为无穷远点O,则结果为O

            RP.x = RP.y = INT\_MAX;

            return RP;

        }

        else {//若P不为无穷远点P，则结果为P

            RP.x = P.x, RP.y = P.y;

            return RP;

        }

    }

    if (P.x == Q.x && P.y != Q.y) {//若两个点x坐标相同，但y坐标不同，即为不同点，二者相加结果为与椭圆曲线相交的第三个点——无穷远点O

        //此时对返回的结果点x和y设为最大值即特殊值INT\_MAX

        RP.x = RP.y = INT\_MAX;

        return RP;

    }

    if (P.x == Q.x && P.y == Q.y) {//若两点相同，为同一点，相加的结果为切线

        int y0 = mod(2 \* P.y, p);

        int x0 = mod(3 \* P.x \* P.x + a, p);

        int a, b;

        ExtendedEuclideanAlgorithm(y0, p, a, b);//调用扩展欧几里得算法求逆元

        int y0ni = a;

        int k = mod(x0 \* y0ni, p);

        int x3 = mod(k \* k - 2 \* P.x, p);

        int y3 = mod(k \* (P.x - x3) - P.y, p);

        RP.x = x3, RP.y = y3;

        return RP;

    }

    else {//若两点不相同

        int y0 = mod(Q.y - P.y, p);

        int x0 = mod(Q.x - P.x, p);

        int a, b;

        ExtendedEuclideanAlgorithm(x0, p, a, b);//调用扩展欧几里得算法求逆元

        int x0ni = a;

        int k = mod(y0 \* x0ni, p);

        int x3 = mod(k \* k - P.x - Q.x, p);

        int y3 = mod(k \* (P.x - x3) - P.y, p);

        RP.x = x3, RP.y = y3;

        return RP;

    }

}

Point DoubleAndAdd\_ScalarProductOfPoint(Point P, int m) {//功能四：对在椭圆曲线E\_p(a,b)上的点P，使用倍加-和算法计算mP

    stack<int> s = DecimalToBinary(m);

    Point T = P;//转换为二进制的结果最高位一定为1，所以T初始化一定为（dt=1）\*P=P

    s.pop();

    for (int i = s.size(); i > 0; i--) {

        T = AddTwoPoints(T, T);//Double

        if (s.top())

            T = AddTwoPoints(T, P);//Double+Add

        s.pop();

    }

    return T;

}

Point ScalarProductOfPoint(Point P, int m) {//普通算法实现功能四

    Point temp\_P = P;

    for (int i = 2; i <= m; i++)

        temp\_P = AddTwoPoints(temp\_P, P);

    return temp\_P;

}

void OrdOfPointOnEllipticCurve(Point P) {//功能五：对在椭圆曲线E\_p(a,b)上的点P，计算阶ord（P）

    if (IfIsInfinitePoint\_O(P)) {//若P为O，则其阶为1

        cout << "P which is O:ord(P)=1" << endl;

        return;

    }

    int i = 2;

    while (1) {

        if (IfIsInfinitePoint\_O(DoubleAndAdd\_ScalarProductOfPoint(P, i))) {//如果iP=O,则i就是最小的正整数

            cout << "ordP(" << P.x << "," << P.y << ")=" << i << endl;

            return;

        }

        i++;

    }

}

void OrdOfEllipticCurve() {//功能六：对在椭圆曲线E\_p(a,b)，计算阶#E

    Point p\_O(INT\_MAX, INT\_MAX);

    AllPoints.push\_back(p\_O);//无穷远点O一定在椭圆曲线上

    for (int i = 0; i < p; i++) {

        int y\_2 = mod(pow(i, 3) + a \* i + b, p);//由椭圆方程求出y^2

        int y1 = 0, y2 = 0;//记录求出的两个二次同余方程的解

        if (LegendreSymbol(y\_2, p) == -1)//若勒让德符号为-1，一定无解，一定不在椭圆曲线上

            continue;

        else {//若勒让德符号为1，则一定有解，求解二次同余方程

            for (int j = 0; j <= p - 1; j++) {//由于当p一般较小时，可以直接遍历求解二次同余方程

                if (mod(pow(j, 2), p) == y\_2) {

                    if (j == 0) {//若j==0，则实际上只解出来一个点

                        Point p(i, 0);

                        AllPoints.push\_back(p);

                        break;

                    }

                    y1 = j;

                    y2 = p - y1;//另一解与y1相加为p

                    Point p1(i, y1);

                    AllPoints.push\_back(p1);

                    Point p2(i, y2);

                    AllPoints.push\_back(p2);

                    break;

                }

            }

        }

    }

    cout << "#E of E\_" << p << "(" << a << "," << b << ")= " << AllPoints.size() << endl;//此时vector的长度容量即所有有理点的个数，即椭圆曲线的阶

    return;

}

void AllPointsOfEllipticCurve() {//功能七：对在椭圆曲线E\_p(a,b)，计算所有点

    //由于此时已经通过OrdOfEllipticCurve()将所有点都放入了AllPoints，此时只需要输出即可

    int size = AllPoints.size();

    cout << "All the points on E\_" << p << "(" << a << "," << b << ") are:" << endl;

    cout << "{ O , ";

    for (int i = 1; i < size; i++) {

        if (i == size - 1) {

            Point P = AllPoints[i];

            cout << "(" << P.x << "," << P.y << ") }." << endl;

        }

        else {

            Point P = AllPoints[i];

            cout << "(" << P.x << "," << P.y << ") ,";

        }

    }

    return;

}

Point InversePoint(Point P) {//功能八（拓展功能）：对在椭圆曲线E\_p(a,b)上的点P，计算其逆元-P

    Point IP;

    if (IfIsInfinitePoint\_O(P)) {

        IP.x = IP.y = INT\_MAX;

        return IP;

    }

    IP.x = mod(P.x, p);

    IP.y = mod(-P.y, p);

    return IP;

}

int main() {

    cout << "Z\_P——E\_p(a, b) " << endl;

    cout << "If you want to input the point of infinity:O，its x and y shouble be INT\_MAX——2147483647" << endl;

    cout << "Fowllowing are some fuctions of E\_p(a,b): " << endl;

    //功能一：给定参数p，a，b，判断E\_p(a,b)是否为椭圆曲线：

    while (1)

    {

        cout << "Please input p,a,b of E\_p(a,b):" << endl;

        cout << "p=";

        cin >> p;

        cout << "a=";

        cin >> a;

        cout << "b=";

        cin >> b;

        cout << "Q1:With the given factor p、a、b,is this curve a Elliptic Curve?" << endl;

        cout << "A1:";

        flag = true;

        If\_EllipticCurve();

        if (flag)

            break;

        else

            cout << "Please input again!" << endl;

    }

    //功能二：判断给定的点P，Q是否在椭圆曲线E\_p(a,b)上

    int xp, yp, xq, yq;

    Point P, Q;

    cout << "Please input x&y for P、Q:" << endl;

    while (1) {

        cout << "Point P:";

        cin >> xp >> yp;

        P.x = xp, P.y = yp;

        if (IfIsOnEllipticCurve(P)) {

            if (IfIsInfinitePoint\_O(P)) {//若输入的点就是无穷远点O

                cout << "P which is O is on E\_" << p << "(" << a << "," << b << ")" << endl;

                break;

            }

            else {

                cout << "P(" << P.x << "," << P.y << ") is on E\_" << p << "(" << a << "," << b << ")" << endl;

                break;

            }

        }

        else {

            cout << "P(" << P.x << "," << P.y << ") is not on E\_" << p << "(" << a << "," << b << ")" << endl;

            cout << "Please input again!" << endl;

        }

    }

    while (1) {

        cout << "Point Q:";

        cin >> xq >> yq;

        Q.x = xq, Q.y = yq;

        if (IfIsOnEllipticCurve(Q)) {

            if (IfIsInfinitePoint\_O(Q)) {//若输入的点就是无穷远点O

                cout << "Q which is O is on E\_" << p << "(" << a << "," << b << ")" << endl;

                break;

            }

            else {

                cout << "Q(" << Q.x << "," << Q.y << ") is on E\_" << p << "(" << a << "," << b << ")" << endl;

                break;

            }

        }

        else {

            cout << "Q(" << Q.x << "," << Q.y << ") is not on E\_" << p << "(" << a << "," << b << ")" << endl;

            cout << "Please input again!" << endl;

        }

    }

    //功能三：对在椭圆曲线E\_p(a,b)上的两点P,Q，计算P+Q

    cout << "Q3:With the given Point P、Q,what is the result of P+Q?" << endl;

    cout << "A3:";

    Point RP = AddTwoPoints(P, Q);

    if (!IfIsInfinitePoint\_O(RP))

        cout << "P(" << P.x << "," << P.y << ") + Q(" << Q.x << "," << Q.y << ")=(" << RP.x << "," << RP.y << ")" << endl;

    else

        cout << "P(" << P.x << "," << P.y << ") + Q(" << Q.x << "," << Q.y << ")=O" << endl;

    //功能四：对在椭圆曲线E\_p(a,b)上的点P，使用倍加-和算法计算mP

    int m;

    cout << "Please input m in mP:" << endl;

    cin >> m;

    cout << "Q4:With the given factor m,what is the result of mP?" << endl;

    cout << "A4:";

    Point T = DoubleAndAdd\_ScalarProductOfPoint(P, m);

    if (!IfIsInfinitePoint\_O(T))//若不为无穷远点O，则正常输出结果

        cout << m << "P(" << P.x << "," << P.y << ")=(" << T.x << "," << T.y << ")" << endl;

    else

        cout << m << "P(" << P.x << "," << P.y << ")=O" << endl;

    //功能五：对在椭圆曲线E\_p(a,b)上的点P，计算阶ord（P）

    cout << "Q5:With the given Point P,what is order(P)?" << endl;

    cout << "A5:";

    OrdOfPointOnEllipticCurve(P);

    //功能六：对在椭圆曲线E\_p(a,b)，计算阶#E

    cout << "Q6:With the given E\_p(a,b) , what is #E?" << endl;

    cout << "A6:";

    OrdOfEllipticCurve();

    //功能七：对在椭圆曲线E\_p(a,b)，计算所有点

    cout << "Q7:With the given E\_p(a,b),what are those points on E?" << endl;

    cout << "A7:";

    AllPointsOfEllipticCurve();

    //功能八（拓展功能）：对在椭圆曲线E\_p(a,b)上的点P，计算其逆元-P

    cout << "Q8:With the given Point P,what is its inverse Point -P?" << endl;

    cout << "A8:";

    Point IP = InversePoint(P);

    if (IfIsInfinitePoint\_O(IP))

        cout << "-P which is -O=O." << endl;

    else

        cout << "-P(" << P.x << "," << P.y << ")=(" << IP.x << "," << IP.y << ")" << endl;

    //所有功能展示结束，程序退出

    return 0;

}

* **说明部分：**

1.首先选择Windows平台，编程工具使用Visual Studio 2022，**利用之前的知识模块解耦实现**，使用如下函数：

1. bool if\_vegetable(int n)//判断是否为素数,辅助使用计算欧拉函数,以及用来各种地方
2. stack<int> DecimalToBinary(unsigned  long long n) //计算十进制转二进制，用一个栈s来保存，用来辅助使用平方乘算法

（3）unsigned long long  SquareAndMultiplyAlgorithm(unsigned long long a, unsigned long long n, unsigned long long m) //平方乘算法

（4）int ExtendedEuclideanAlgorithm(int a, int b, int& x, int& y)//扩展的欧几里得算法求逆元，用于求逆元

（5）int mod(int a, int p) //重载取模运算mod

（6）int LegendreSymbol(int a, int p) //计算勒让德符号

（7）bool IfIsInfinitePoint\_O(Point P) {//判断是否为无穷远点，设置无穷远点O的x和y均为特殊值INT\_MAX

PS：

1. 由于无穷远点O的特殊性，输入坐标时候输入**INT\_MAX——2147483647表示无穷远点O**。并此后将其作为有限域的零元特殊处理。
2. 取模运算int mod(int a, int p)不同于一般的取余，目的是将所有最后的坐标结果都统一为正值，即坐标的最简结果x与y都在Z\*\_p中。
3. 由于基于Z\_p的椭圆曲线E\_p(a,b)中的参数p往往是一个较小的值，因此计算勒让德符号直接使用欧拉判别条件，而没有使用二次互反律等进行初步简化。
4. 基于以上七种基础函数实现题目所要求的以下八种功能：（**七种基础功能+一种拓展功能**）
5. 给定参数p,a,b ，判断E\_p(a,b)是否为椭圆曲线。
6. 判断给定的点P,Q是否在椭圆曲线E\_p(a,b)上。

【特殊处理无穷远点O，其一定在椭圆曲线上；

**对于一般的给定点，由于p一般较小，采用遍历的方式求解二次同余方程】**

1. 对在椭圆曲线E\_p(a,b)上的两点P,Q，计算P+Q。

【考虑以下多种情况——两点或其中一点为无穷远点、两点均不为无穷远点但是x坐标相同、两点完全相同、两点完全不相同的一般情况。

】

1. 对在椭圆曲线E\_p(a,b)上的点P，使用倍加-和算法计算mP。

【调用写好的功能（3）函数Point AddTwoPoints(Point P, Point Q)，并使用Double-And-Add算法实现；

同时实现了计算效率较低的Point ScalarProductOfPoint(Point P, int m)作为对比】

1. 对在椭圆曲线E\_p(a,b)上的点P，计算阶ord(P)。

【调用写好的（4）函数Point DoubleAndAdd\_ScalarProductOfPoint(Point P, int m)遍历实现】

1. 对在椭圆曲线E\_p(a,b)，计算阶#E。

【采用一个vector类型变量储存所有椭圆曲线上的点，依次代入曲线计算，并求解对应的二次同余方程，判断该点是否在曲线上。

最后vector变量的大小size即椭圆曲线上有理点的个数，即椭圆的阶#E。**注意无穷远点O！**】

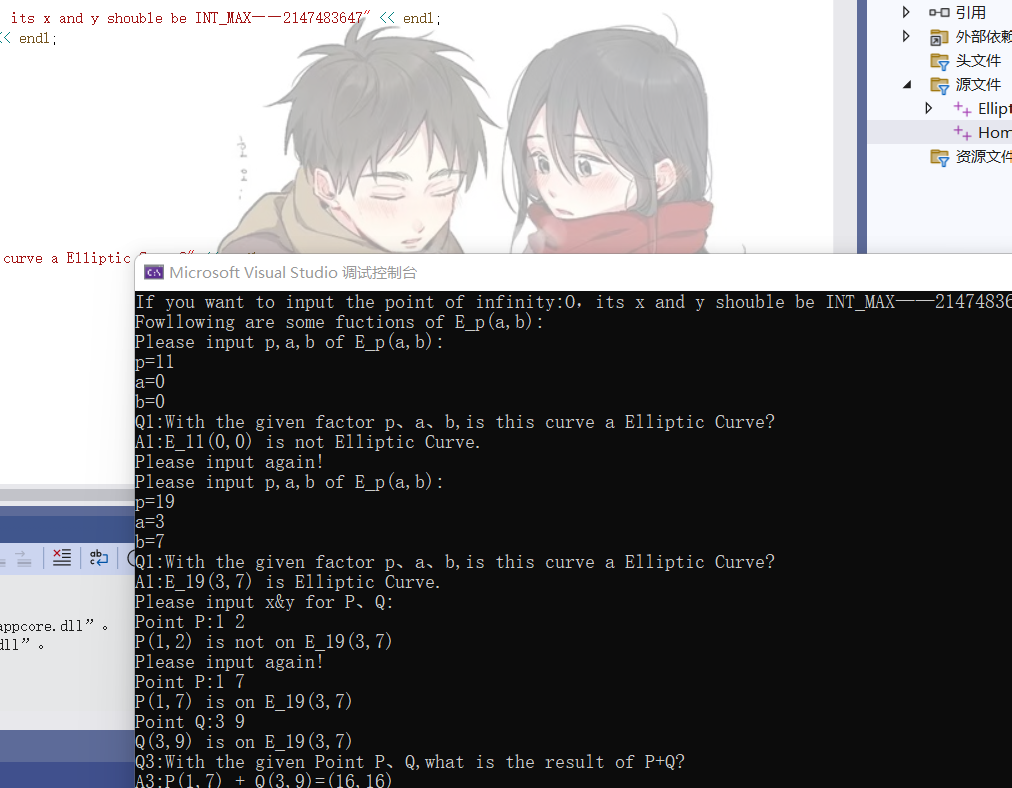
1. 对在椭圆曲线E\_p(a,b),计算所有点。
2. **拓展功能**：**对在椭圆曲线E\_p(a,b)上的点P，计算其逆元-P。**

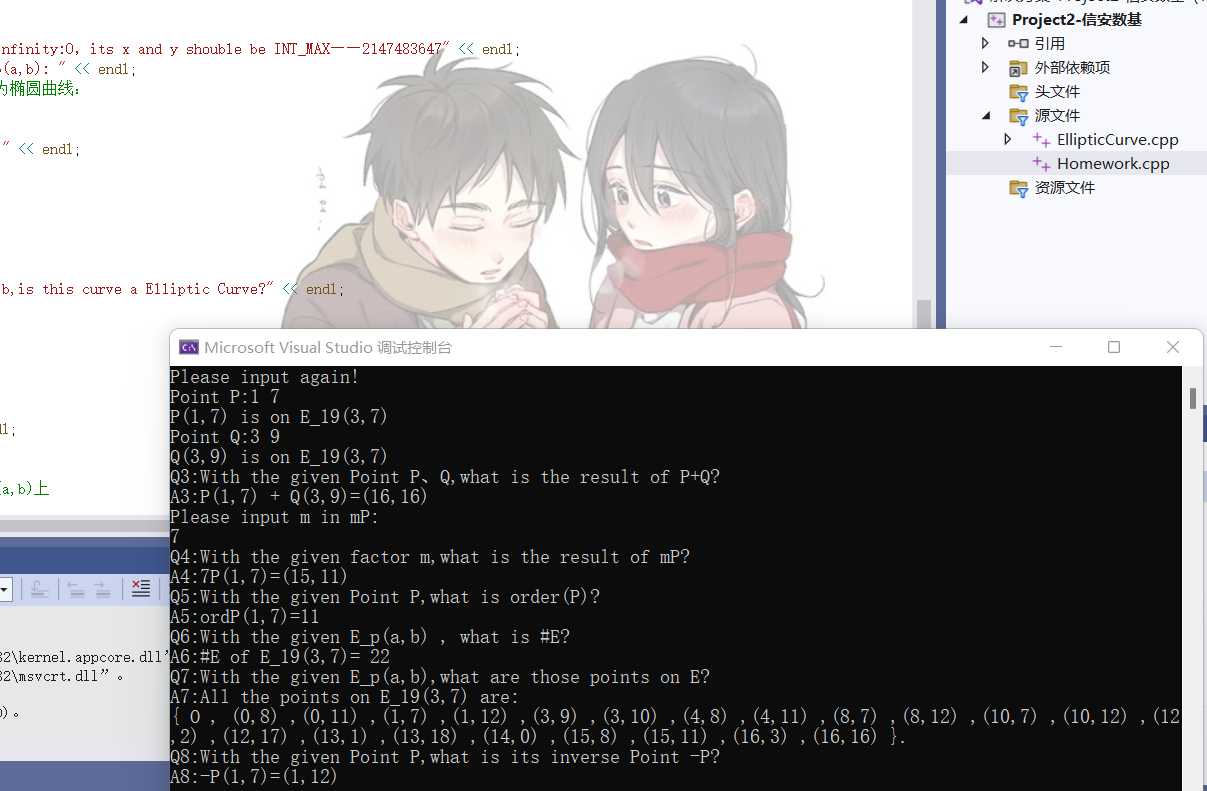
* **运行示例：**

1. 两个点均为非无穷远点

1.作业例子：

**输入样例：p=19 a=3 b=7 P=(1,7) Q=(3,9) m=7**





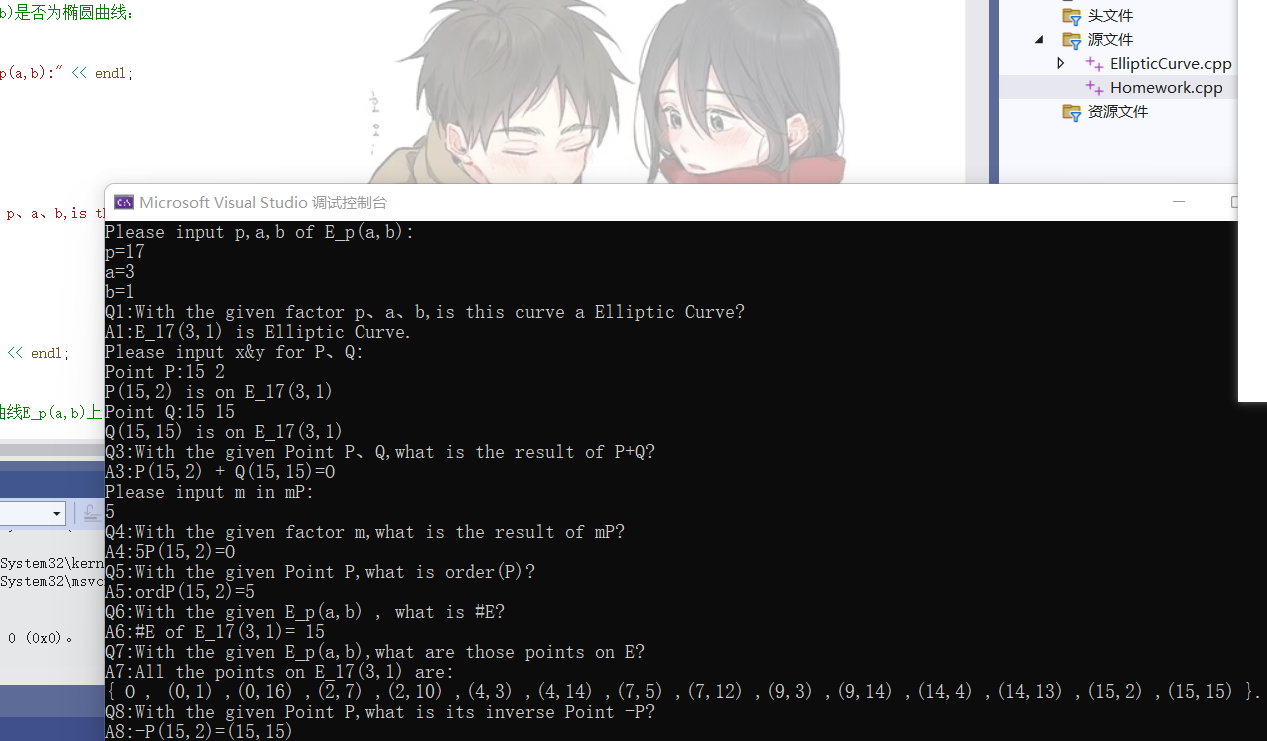
2第五次作业题1：

**输入样例：p=11 a=1 b=6 P= (3,6) Q= (7,9) m=13**



1. 第五次作业题2：

**输入样例：p=17 a=3 b=1 P=(15,2) Q=(15,15) m=5**



二．两个点其中之一为无穷远点

**输入样例：p=17 a=3 b=1 P=O Q=(15,2) m=3**



三.两个点都为无穷点

**输入样例：p=17 a=3 b=1 P=O Q=O m=4**



* **其他：思考问题——**

1. 针对二次同余方程的遍历求解和勒让德符号的欧拉判别式求解——**若p较大怎么办？**

答：本实验中由于默认p一般会作为较小的参数出现——因此二次同余方程采用了遍历求解，勒让德符号也直接使用了欧拉判别式求解，但p较大时，这种方式计算效率较低，因此应该改用以下方式，以此来提升效率：

1. 二次同余方程的求解：采用首先原根，再将原同余方程转换为的方式求解。
2. 勒让德符号的计算：结合二次互反律、勒让德符号的基础性质以及-1、2、1的常用勒让德符号结论进行求解。



**彩蛋！**

**我的专属水印！**