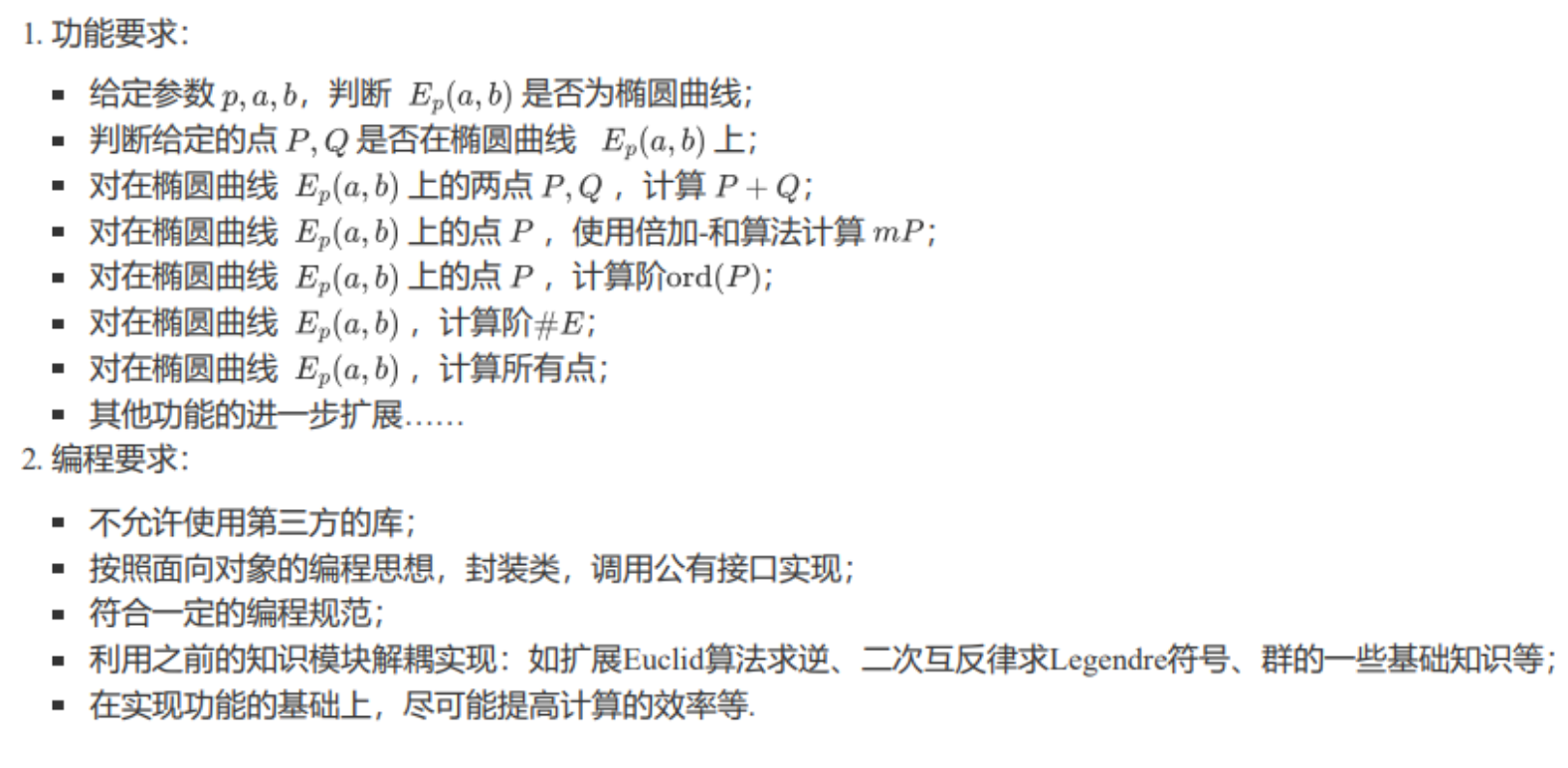
# 第8次编程练习报告

姓名：申宗尚 学号：2213924 班级：信安班

##### 编程练习1——实现基本的Zp上的椭圆曲线Ep(a,b)上的计算，具体要求如下：

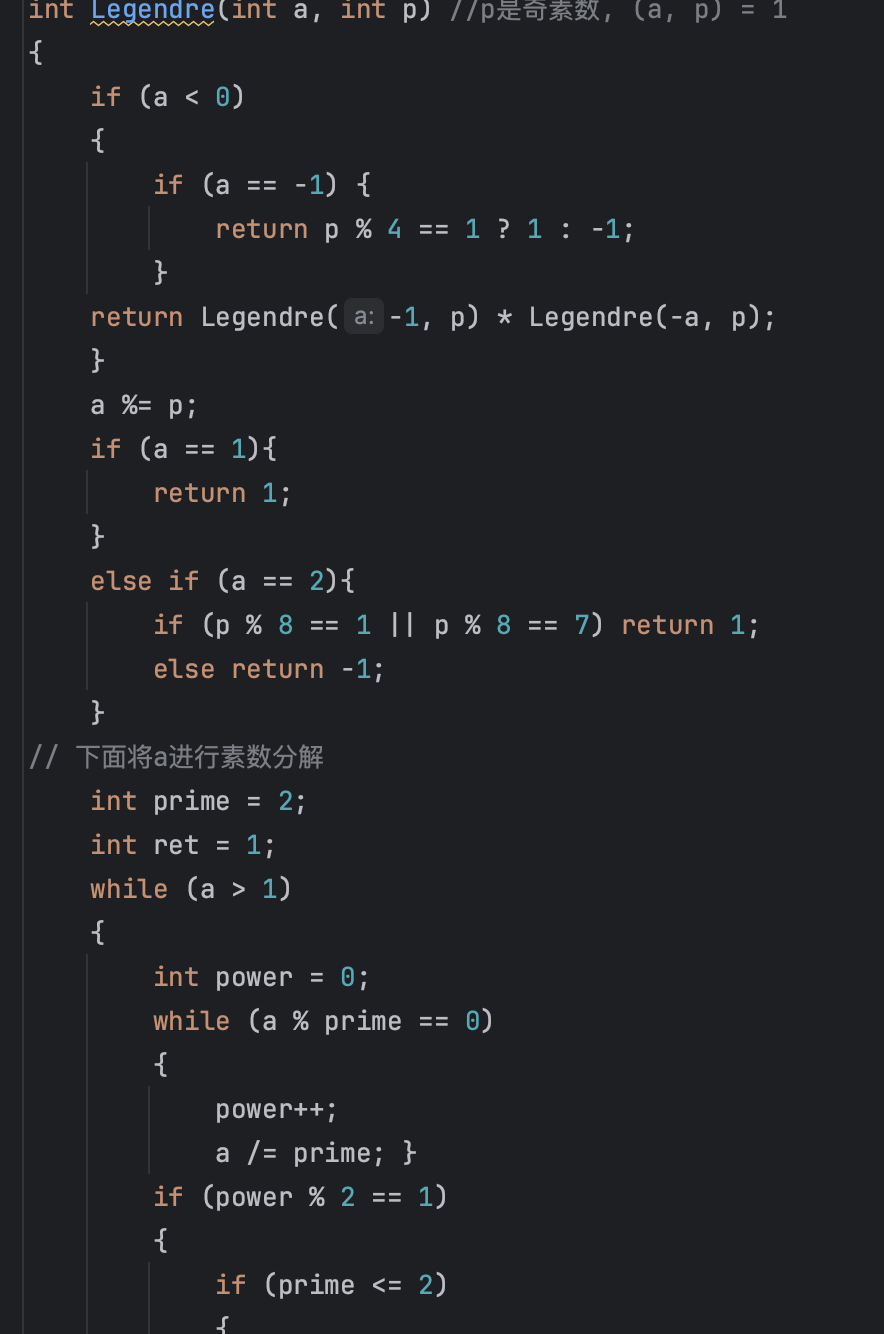


* **源码部分：**
* #include <iostream>  
  using namespace std;  
    
  #define Elliptic\_Curve\_EC "E\_" << p << "(" << a << ',' << b << ")"  
  #define Point\_P "P(" << x << "," << y << ")"  
    
  class Point{  
  public:  
   int x,y;  
   bool isINF;//是否为无穷远点  
   Point(int x = 0, int y = 0, bool isINF = 0);  
   friend ostream& operator<< (ostream& out, const Point& P);//重定义输出  
   bool operator ==(const Point& p);  
   void Output(ostream& out) const;  
  };  
    
  class Elliptic\_Curve  
  {  
  private:  
   int p;  
   int a, b;  
  public:  
   Elliptic\_Curve(int p, int a, int b);  
   bool Is\_Inverse(const Point& p1, const Point& p2); //判断两个点是否互逆  
   bool Test\_Is\_Elliptic\_Curve(); //检查当前参数是否能构成椭圆曲线  
   bool Is\_On\_Elliptic\_Curve(const Point& p); //判断p点是否在椭圆曲线上  
   Point Add(const Point& p1, const Point& p2); //进行点加运算  
   Point Add\_K\_Times(Point p, int k); //对点p进行k倍加  
   int Ord\_Of\_Point(const Point& p); //计算点p的阶  
   int Ord\_Of\_Elliptic\_Curve(); //计算此椭圆曲线的阶#E  
   int Show\_All\_Points(); //展示出椭圆曲线上的所有点  
    
  };  
    
  int Legendre(int a, int p) //p是奇素数, (a, p) = 1  
  {  
   if (a < 0)  
   {  
   if (a == -1) {  
   return p % 4 == 1 ? 1 : -1;  
   }  
   return Legendre(-1, p) \* Legendre(-a, p);  
   }  
   a %= p;  
   if (a == 1){  
   return 1;  
   }  
   else if (a == 2){  
   if (p % 8 == 1 || p % 8 == 7) return 1;  
   else return -1;  
   }  
  // 下面将a进行素数分解  
   int prime = 2;  
   int ret = 1;  
   while (a > 1)  
   {  
   int power = 0;  
   while (a % prime == 0)  
   {  
   power++;  
   a /= prime; }  
   if (power % 2 == 1)  
   {  
   if (prime <= 2)  
   {  
   return Legendre(prime, p);  
   }  
   else  
   {  
   if (((prime - 1) \* (p - 1) / 4) % 2 == 1)  
   {  
   ret = -ret; }  
   ret \*= Legendre(p, prime);  
   }  
   }  
   prime++; }  
   return ret;  
  }  
    
  int pow(int x, int n)//x的n次方  
  {  
   int ret = 1;  
   while (n)  
   {  
   if (n & 1) {  
   ret \*= x;  
   }  
   x \*= x;  
   n >>= 1; }  
   return ret; }  
    
  int Get\_Inverse(int a, int m) //在 (a, m) = 1 的条件下，求a模m的乘法逆元  
  {  
   a = (a + m) % m;  
   int s0 = 1, s1 = 0;  
   int r0 = a, r1 = m;  
   while (1)  
   {  
   int q = r0 / r1;  
   int tmp = r1;  
   r1 = r0 % r1;  
   r0 = tmp;  
   if (r1 == 0) {  
   break; }  
   tmp = s1;  
   s1 = s0 - s1 \* q;  
   s0 = tmp;  
   }  
   return (s1 + m) % m;  
  }  
    
  Point::Point(int x, int y, bool isINF)  
  {  
   this->x = x;  
   this->y = y;  
   this->isINF = isINF;  
  }  
    
  bool Point::operator == (const Point& p)  
  {  
   return x == p.x && y == p.y;  
  }  
    
  ostream& operator<< (ostream& out, const Point& p)  
  {  
   p.Output(out);  
   return out;  
  }  
    
  void Point::Output(ostream& out) const  
  {  
   if (isINF) cout << 'O';  
   else cout << '(' << x << ',' << y << ')';  
  }  
    
  Elliptic\_Curve::Elliptic\_Curve(int p, int a, int b) {//椭圆曲线构造函数  
  this->p = p;  
  this->a = a;  
  this->b = b;  
  }  
    
  bool Elliptic\_Curve::Is\_Inverse(const Point& p1, const Point& p2){ //判断两个点是否互逆  
  return (p1.x - p2.x) % p == 0 && (p1.y + p2.y) % p == 0;  
  }  
    
  bool Elliptic\_Curve::Test\_Is\_Elliptic\_Curve() {//检查当前参数是否能构成椭圆曲线  
  int tmp = pow(a, 3) \* 4 + pow(b, 2) \* 27;  
  return tmp % p != 0;  
  }  
    
  bool Elliptic\_Curve::Is\_On\_Elliptic\_Curve(const Point& pt) {//判断p点是否在椭圆曲线上  
  int tmp = pow(pt.y, 2) - (pow(pt.x, 3) + a \* pt.x + b);  
  return tmp % p == 0;  
  }  
    
  Point Elliptic\_Curve::Add(const Point& p1, const Point& p2){ //进行点加运算  
   if (p1.isINF){  
   return p2;  
   }  
   else if (p2.isINF){  
   return p1;  
   }  
   else if (Is\_Inverse(p1, p2)){  
   return { 0, 0, true };  
   }  
   else  
   {  
   if((p1.x-p2.x) % p == 0)  
   {  
   int k = ((3 \* p1.x \* p1.x + a) \* Get\_Inverse(2 \* p1.y, p) % p + p) % p;  
   int x3 = ((k \* k - 2 \* p1.x) % p + p) % p;  
   int y3 = ((k \* (p1.x - x3) - p1.y) % p + p) % p;  
   return { x3, y3 };  
   }  
   else{  
   int k = ((p2.y - p1.y) \* Get\_Inverse(p2.x - p1.x, p) % p + p) % p;  
   int x3 = ((k \* k - p1.x - p2.x) % p + p) % p;  
   int y3 = ((k \* (p1.x - x3) - p1.y) % p + p) % p;  
   return { x3, y3 };  
   }  
   }  
   }  
    
  Point Elliptic\_Curve::Add\_K\_Times(Point p, int k){ //对点p进行k倍加  
   Point ret(0, 0, true);  
   while (k)  
   {  
   if (k & 1) {  
   ret = Add(ret, p);  
   }  
   p = Add(p, p);  
   k >>= 1; }  
   return ret;  
  }  
    
  int Elliptic\_Curve::Ord\_Of\_Point(const Point& pt){ //计算点p的阶  
   int ret = 1;  
  Point tmp = pt;  
  while (!tmp.isINF)  
  {  
  tmp = Add(tmp, pt);  
  ++ret; }  
  return ret;  
  }  
    
  int Elliptic\_Curve::Ord\_Of\_Elliptic\_Curve(){ //计算此椭圆曲线的阶#E  
  int ret = 1;  
  for (int x = 0; x < p; ++x){  
   int tmp = (x \* x \* x + a \* x + b + p) % p;  
   if (tmp == 0){  
   ret += 1;  
   }  
   else if (Legendre(tmp, p) == 1){  
   ret += 2; }  
   }  
   return ret;  
  }  
    
  int Elliptic\_Curve::Show\_All\_Points() //展示出椭圆曲线上的所有点  
  {  
  cout << "O ";  
  int sum = 1;  
  for (int x = 0; x < p; ++x) {  
   int tmp = (x \* x \* x + a \* x + b + p) % p;  
   if (tmp == 0) {  
   cout << " (" << x << ',' << "0) ";  
   sum++;  
   } else if (Legendre(tmp, p) == 1) //贡献两个点  
   {  
   for (int y = 1; y < p; ++y) //从1遍历到p-1，寻找解  
   {  
   if ((y \* y - tmp) % p == 0) {  
   cout << " (" << x << ',' << y << ") ";  
   sum++;  
   cout << " (" << x << ',' << p - y << ") ";  
   sum++;  
   break;  
   }  
   }  
   }  
  }  
  cout<<endl;  
  return sum;  
   }  
    
  int main() {  
   int p, a, b;  
   cout << "请输入椭圆曲线的参数 p: "; cin >> p;  
   cout << "请输入椭圆曲线的参数 a: "; cin >> a;  
   cout << "请输入椭圆曲线的参数 b: "; cin >> b;  
   Elliptic\_Curve ec(p, a, b);  
   int x, y;  
   cout << endl;  
   cout << "1.判断所给参数是否能构成一个椭圆曲线" << endl; cout << Elliptic\_Curve\_EC ;  
   if (!ec.Test\_Is\_Elliptic\_Curve())  
   {  
   cout << "不构成 ";  
   return 0; }  
   cout << "构成Elliptic\_Curve" << endl;  
   cout << endl;  
   cout << "2.判断给出的点是否在给定的椭圆曲线上" << endl;  
   cout << "输入 x: ";  
   cin >> x;  
   cout << "输入 y: ";  
   cin >> y;  
   cout << Point\_P " is ";  
   if (!ec.Is\_On\_Elliptic\_Curve(Point(x, y))) cout << "not "; cout << "on " << Elliptic\_Curve\_EC << endl;  
   cout << endl;  
   cout << "3.计算给定的两点相加" << endl; int x1, y1, x2, y2;  
   cout << "输入 x1: ";  
   cin >> x1;  
   cout << "输入 y1: ";  
   cin >> y1;  
   cout << "输入 x2: ";  
   cin >> x2;  
   cout << "输入 y2: ";  
   cin >> y2;  
   cout << "结果是" << ec.Add({ x1, y1 }, { x2, y2 }) << endl;  
   cout << endl;  
   cout << "4.计算给出的点的倍加" << endl;  
   cout << "输入 x: ";  
   cin >> x;  
   cout << "输入 y: ";  
   cin >> y;  
   int times;  
   cout << "输入倍数: ";  
   cin >> times;  
   cout << "结果是" << ec.Add\_K\_Times({ x, y }, times) << endl;  
   cout<<endl;  
   cout << "5.计算给出的点的阶" << endl;  
   cout << "输入 x: ";  
   cin >> x;  
   cout << "输入 y: ";  
   cin >> y;  
   cout << Point\_P << "的阶是" << ec.Ord\_Of\_Point({ x, y }) << endl;  
   cout << endl;  
   cout << "6.计算给出的椭圆曲线的阶" << endl;  
   cout << Elliptic\_Curve\_EC << "的阶是" << ec.Ord\_Of\_Elliptic\_Curve() << endl;  
   cout << endl;  
   cout << "7.列出给出的椭圆曲线上的所有点" << endl;  
   cout << ec.Show\_All\_Points();  
   return 0;  
  }
* **说明部分：**

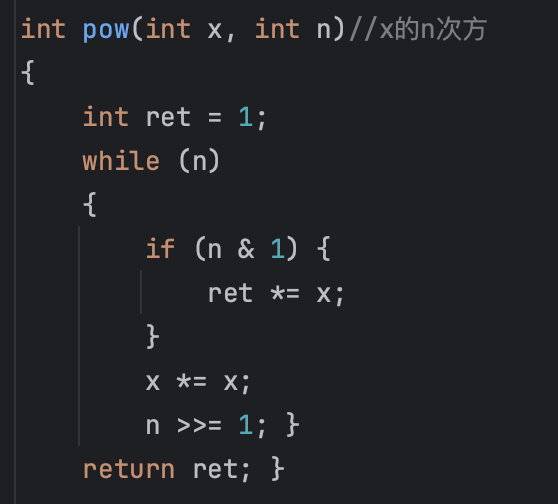
代码中主要实现了两个类：点Point类和椭圆曲线Elliptic\_Curve类。

在点类中，记录三个值x，y和isINF，分别表示点的x坐标，y坐标和是否为无穷远点，然后多态定义了点类的输出、相等判定。

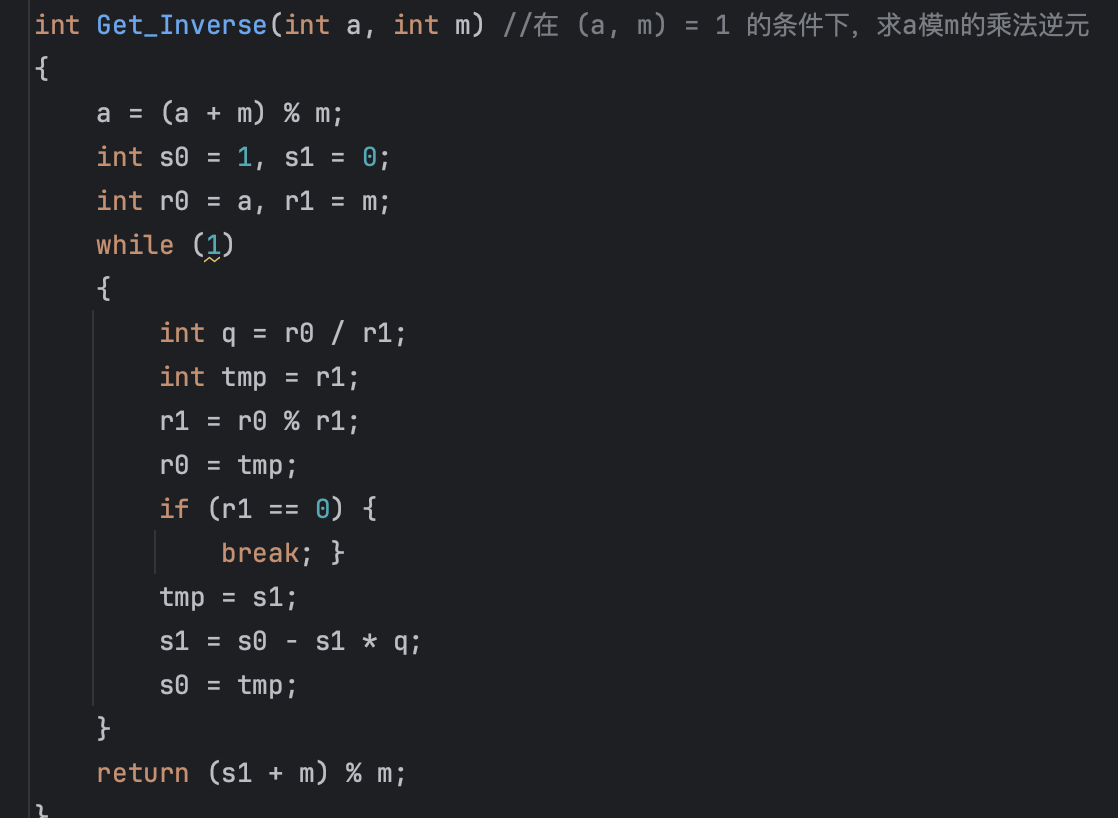
在椭圆曲线类中，通过不同的函数，实现了种种功能，下面挨个介绍：



首先，通过二次互反律进行Legendre符号的计算，这可以计算是否为二次剩余，后面阶的计算等等函数都可以在此利用。



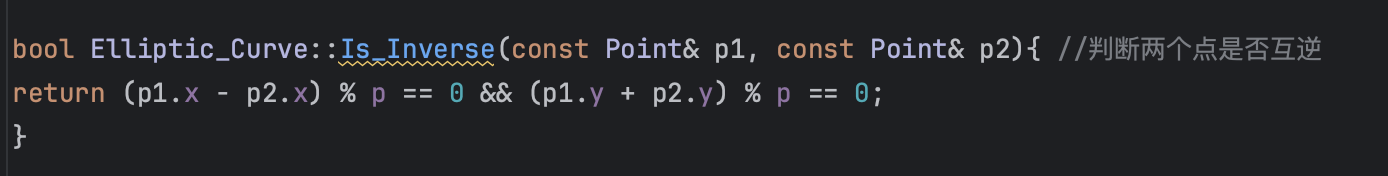
然后，写一个次方pow函数



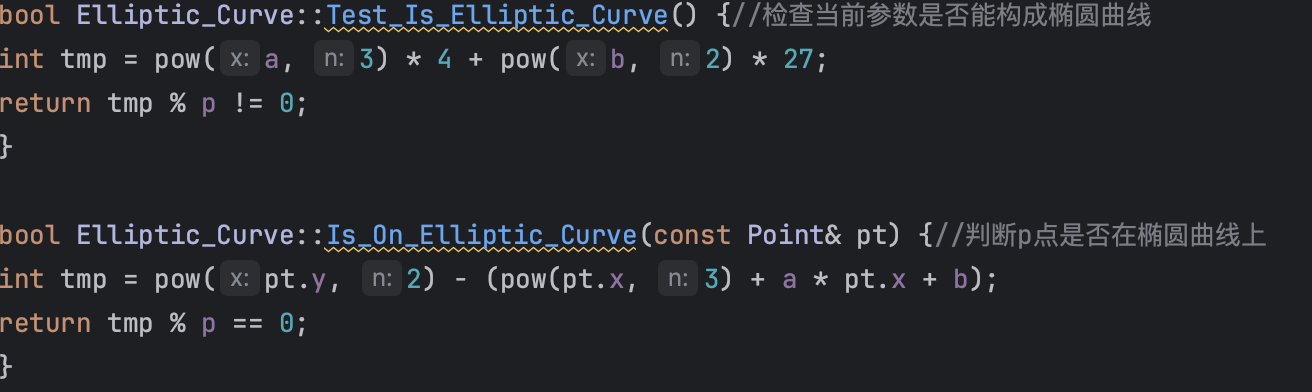
通过之前的编程作业代码，拿出来使用扩展欧几里得定理的求逆元部分，实现get\_Inverse函数。

接下来，进行正式的功能函数介绍：

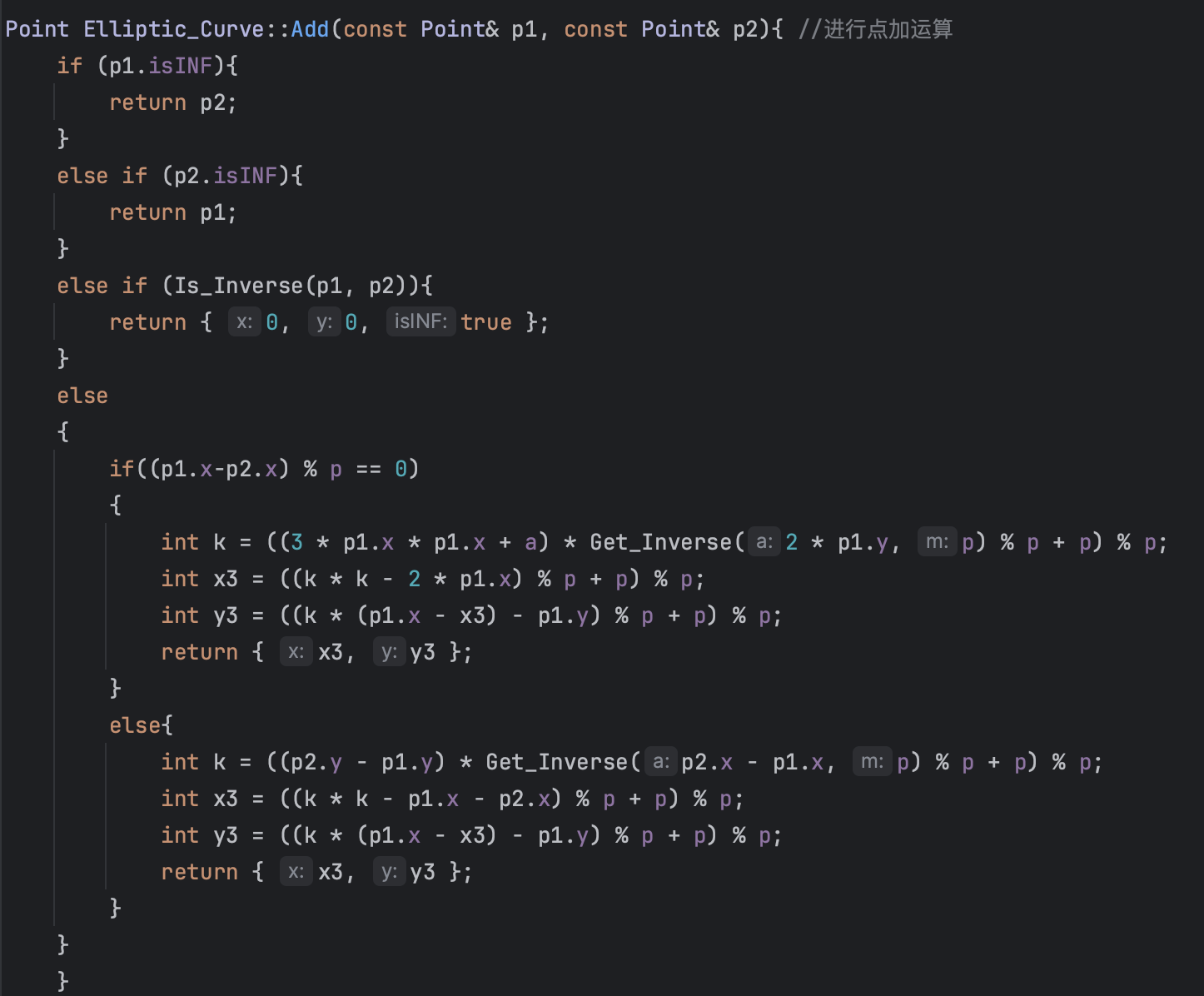
1、判断是否互逆，只需要判断如下通过公式判断。



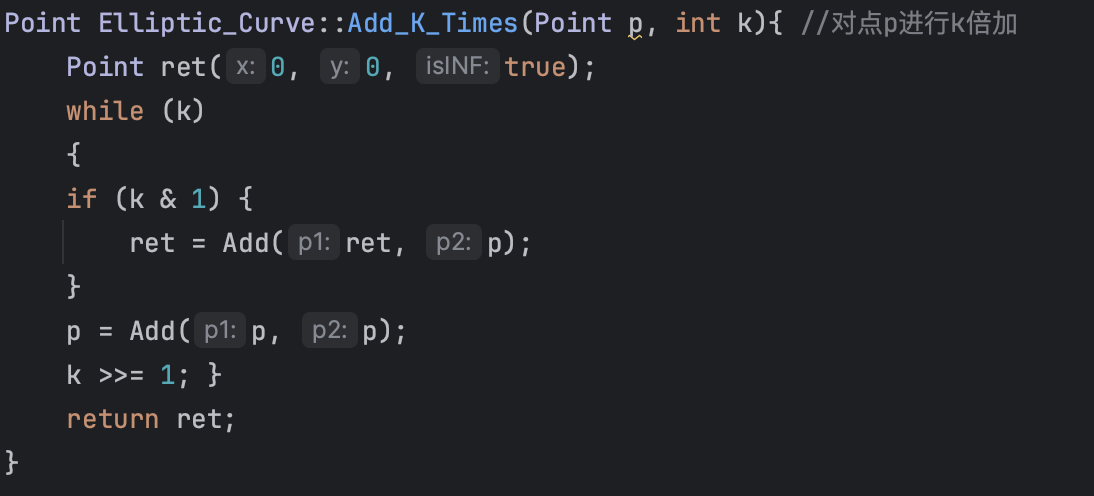
2、3判断是否能构成椭圆曲线、是否在曲线上，也是类似的，使用公式



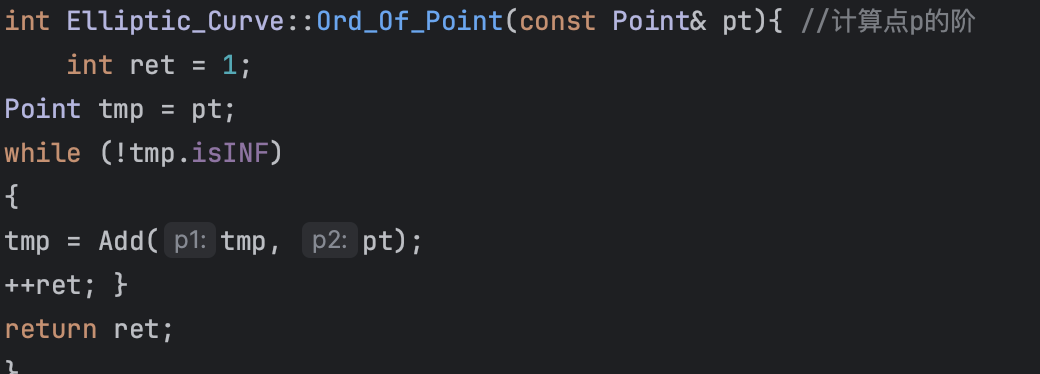
4、点加，先进行无穷远点的讨论，如果一个是无穷远点，最后的值直接等于另一个，如果两个点是互逆的点，则最后的值就是无穷远点。如果都不是特殊情况，就通过点加\倍加的两种公式进行讨论，然后用公式获得最后的结果。



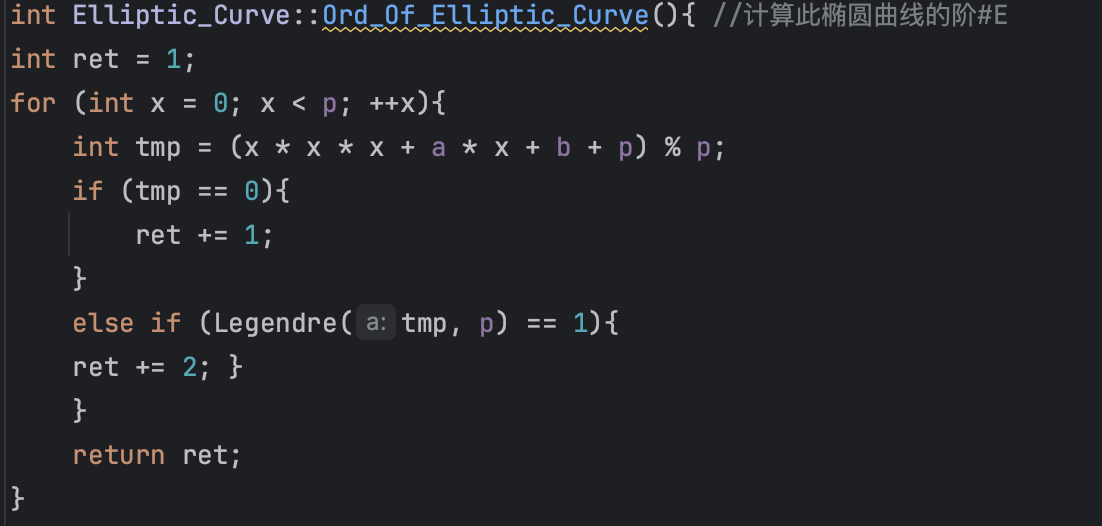
5、k倍加，使用循环，Add k次。



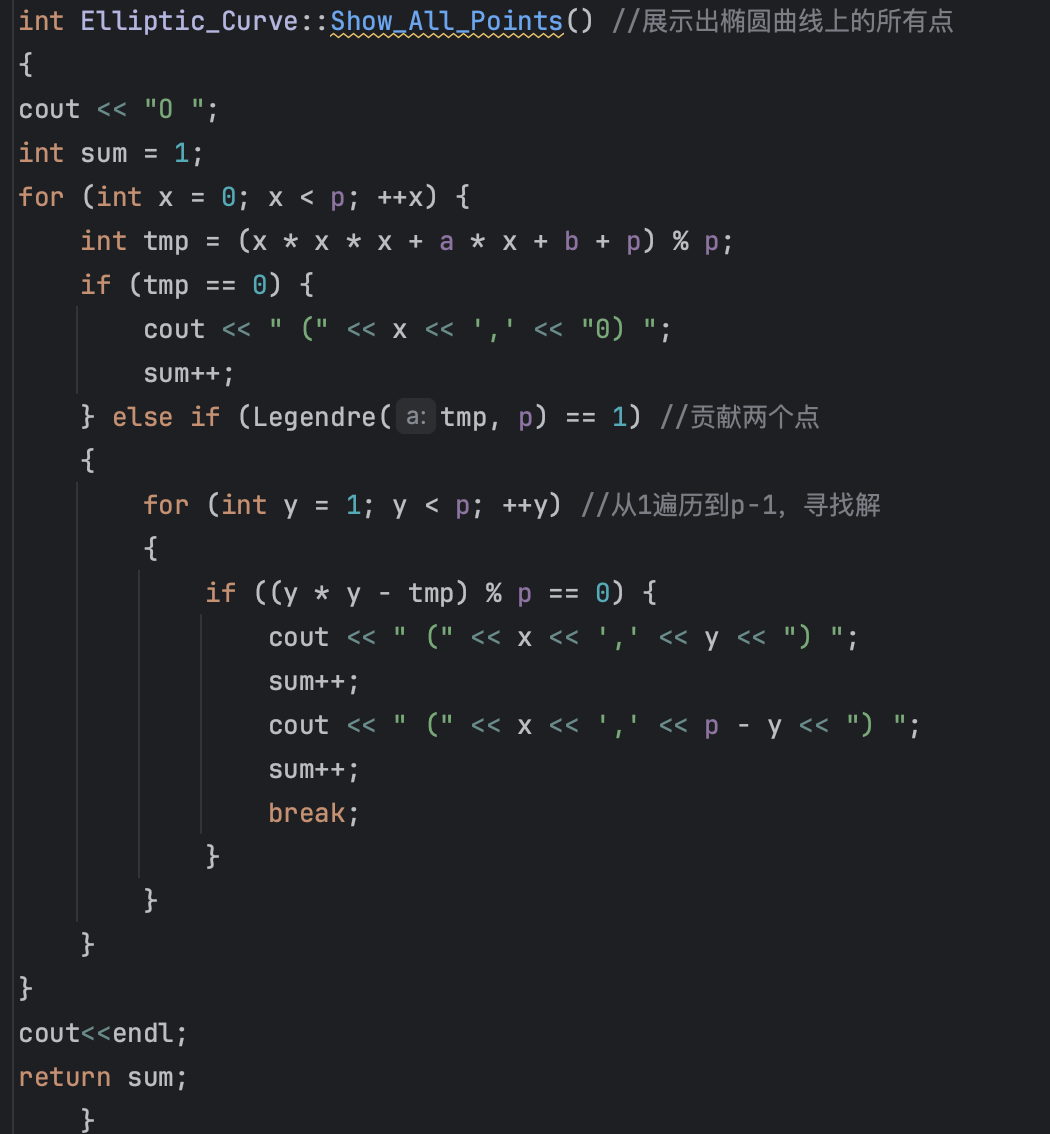
6、计算点p的阶：对p一直进行倍加，知道结果为INF，返回加法次数即为p点的阶：



7、计算椭圆曲线的阶：初始化变量ret为1，存储椭圆曲线的阶，遍历所有可能的x值，对于每个x，计算椭圆曲线上的值，如果是0，则ret+1，不是的话判断是不是p的二次剩余（通过Legendre符号），如果是的话，说明y^2 = tmp有解，因此有两个点，ret+2，从而最终计算出阶。



8、展示所有点：遍历找解。



* **运行示例：**

