# 第5次编程练习报告

姓名：孙蕗 学号：2112060 班级：信安一班

##### **编程练习1——椭圆曲线**

* **源码部分：**

|  |
| --- |
| #include<iostream>  #include<set>  using namespace std;  /\*  实现基本的Zp上的椭圆曲线Ep(a,b)的计算，具体如下：  1. 功能要求：  (1) 给定参数p,a,b ，判断Ep(a,b)是否为椭圆曲线；  (2) 判断给定的点P,Q是否在椭圆曲线Ep(a,b)上；  (3) 对在椭圆曲线Ep(a,b)上的两点P,Q ，计算P+Q ；  (4) 对在椭圆曲线Ep(a,b)上的点P，使用倍加-和算法计算mP ；  (5) 对在椭圆曲线Ep(a,b)上的点 P，计算阶 ord(P)；  (6) 对在椭圆曲线Ep(a,b)，计算阶#E；  (7) 对在椭圆曲线Ep(a,b)，计算所有点；  2. 编程要求：  不允许使用第三方的库；  按照面向对象的编程思想，封装类，调用公有接口实现；  符合一定的编程规范；  利用之前的知识模块解耦实现：如扩展Euclid算法求逆、二次互反律求Legendre符号、群的一些基础知识等；  \*/  //Ep(a,b):y^2=x^3+ax+b(mod p)  //(1) 给定参数p,a,b ，判断Ep(a,b)是否为椭圆曲线；  class EllipticCurve  {  public:  bool isEllipticCurve(int p, int a, int b)  {  int discriminant = (4 \* a \* a \* a + 27 \* b \* b) % p;//(4a^3+27b^2)(mod p)  if (discriminant != 0)  {  return true;  }  else  {  return false;  }  }  //(2) 判断给定的点P,Q是否在椭圆曲线Ep(a,b)上；  bool isPointOnCurve(int p, int a, int b, int x, int y)  {  int left = (y \* y) % p;//Ep(a,b):y^2=x^3+ax+b(mod p)  int right = ((x \* x \* x) + (a \* x) + b) % p;  if (left == right)  {  return true;  }  else  {  return false;  }  }  //(3) 对在椭圆曲线Ep(a,b)上的两点P,Q ，计算P+Q；  int gcd(int m, int n)//最大公约数  {  if (m < n)  {  int temp;  temp = m;  m = n;  n = temp;  }  while (n != 0)  {  int r = m % n;  m = n;  n = r;  }  return m;  }  int lcm(int m, int n)//最小公倍数  {  return m \* n / gcd(m, n);  }  int inverse(int a, int b)//扩展欧几里得算法求逆元  {  int s[10000], t[10000], q[10000], r[10000];  s[0] = 1;  t[0] = 0;  s[1] = 0;  t[1] = 1;  r[0] = a;  r[1] = b;  q[0] = 0;  q[1] = a / b;  int i = 2;  while (true)  {  r[i] = r[i - 2] % r[i - 1];  //cout << "r" << i << "=" << r[i] << endl;  if (r[i] == 0)  {  break;  }  q[i] = r[i - 1] / r[i];  //cout << "q" << i << "=" << q[i] << endl;  s[i] = s[i - 2] - s[i - 1] \* q[i - 1];  //cout << "s" << i << "=" << s[i] << endl;  t[i] = t[i - 2] - t[i - 1] \* q[i - 1];  //cout << "t" << i << "=" << t[i] << endl;  i++;  //cout << endl;  }  return s[i - 1];  }  void addPoints(int p, int a, int b, int x1, int y1, int x2, int y2, int& x3, int& y3)  {  int k;  if (x1 == x2)//对应倍加  {  k = ((3 \* x1 \* x1 + a) \* inverse(2 \* y1, p)) % p;//x1=x2，对应倍加，k=((3x1^2+a)/(2y1))(mod p)  }  else//对应点加  {  if (x2 - x1 > 0)  {  k = ((y2 - y1) \* inverse(x2 - x1, p)) % p;//x1!=x2，对应点加，k=((y2-y1)/(x2-x1))(mod p)  }  else//负数取绝对值，再取反  {  k = ((y2 - y1) \* (0 - inverse(x1 - x2, p))) % p;//x1!=x2，对应点加，k=((y2-y1)/(x2-x1))(mod p)  }  }  if (k < 0)  {  k += p;  }  x3 = (k \* k - x1 - x2) % p;//x3=(k^2-x1-x2)(mod p)  y3 = (k \* (x1 - x3) - y1) % p;//y3=(k(x1-x3)-y1)(mod p)  //如果是负数，变成正数  if (x3 < 0)  {  x3 += p;  }  if (y3 < 0)  {  y3 += p;  }  return;  }  //(4) 对在椭圆曲线Ep(a, b)上的点P，使用倍加 - 和算法计算mP  void multiPoint(int p, int a, int b, int m, int x1, int y1, int& x4, int& y4)  {  int x5 = x1;  int y5 = y1;  int tempX, tempY;  for (int i = 2; i <= m; i++)  {  addPoints(p, a, b, x5, y5, x1, y1, tempX, tempY);  x5 = tempX;  y5 = tempY;  }  x4 = x5;  y4 = y5;  return;  }  //(5) 对在椭圆曲线Ep(a,b)上的点 P，计算阶 ord(P)；  int calculate\_ord(int p, int a, int b, int x1, int y1)  {  int ord = 1;  int result\_x, result\_y;  int x6 = x1;//P'x  int y6 = y1;//P'y  // 存储已经出现过的点  set<pair<int, int>> visitedPoints;  visitedPoints.insert(make\_pair(x6, y6));  while (true)  {  addPoints(p, a, b, x6, y6, x1, y1, result\_x, result\_y);  x6 = result\_x;  y6 = result\_y;  ord++;  // 检查是否出现了重复的点  if (visitedPoints.count(make\_pair(x6, y6)) > 0) {  break; // 退出循环  }  visitedPoints.insert(make\_pair(x6, y6));  }  return ord;  }  //(6) 对在椭圆曲线Ep(a,b)，计算阶#E；  int calculateCurveOrder(int p, int a, int b)  {  int E = 0;  for (int x = 0; x < p; x++)  {  int y\_sqrt = (x \* x \* x + a \* x + b) % p;  int y = 0;  for (int y = 0; y < p; y++)  {  if ((y \* y) % p == y\_sqrt)  {  E++;  }  }  }  E += 1;//无穷远点  return E;  }  //(7) 对在椭圆曲线Ep(a,b)，计算所有点；  void calculateAllPoints(int p, int a, int b)  {  cout << 0 << " ";//无穷远点  for (int x = 0; x < p; x++)  {  int y\_sqrt = (x \* x \* x + a \* x + b) % p;  int y = 0;  for (int y = 0; y < p; y++)  {  if ((y \* y) % p == y\_sqrt)  {  cout << "(" << x << "," << y << ")" << " ";  }  }  }  }  };  int main()  {  int p, a, b;  cout << "p= ";  cin >> p;  cout << "a= ";  cin >> a;  cout << "b= ";  cin >> b;  EllipticCurve e;  //(1) 给定参数p, a, b ，判断Ep(a, b)是否为椭圆曲线；  cout << "1. 给定参数p, a, b ，判断Ep(a, b)是否为椭圆曲线" << endl;  if (e.isEllipticCurve(p, a, b))  {  cout << "E" << p << "(" << a << "," << b << ")" << "是椭圆曲线" << endl;  }  else  {  cout << "E" << p << "(" << a << "," << b << ")" << "不是椭圆曲线" << endl;  }  //(2) 判断给定的点P,Q是否在椭圆曲线Ep(a,b)上；  cout << "2. 判断给定的点P,Q是否在椭圆曲线Ep(a,b)上" << endl;  int Px, Py, Qx, Qy;  cout << "Px= ";  cin >> Px;  cout << "Py= ";  cin >> Py;  cout << "Qx= ";  cin >> Qx;  cout << "Qy= ";  cin >> Qy;  if (e.isEllipticCurve(p, a, b))//判断是不是椭圆曲线  {  if (e.isPointOnCurve(p, a, b, Px, Py))  {  cout << "点P(" << Px << "," << Py << ")在椭圆曲线E" << p << "(" << a << "," << b << ")" << "上" << endl;  }  else  {  cout << "点P(" << Px << "," << Py << ")不在椭圆曲线E" << p << "(" << a << "," << b << ")" << "上" << endl;  }  if (e.isPointOnCurve(p, a, b, Qx, Qy))  {  cout << "点Q(" << Qx << "," << Qy << ")在椭圆曲线E" << p << "(" << a << "," << b << ")" << "上" << endl;  }  else  {  cout << "点Q(" << Qx << "," << Qy << ")不在椭圆曲线E" << p << "(" << a << "," << b << ")" << "上" << endl;  }  }  //(3) 对在椭圆曲线Ep(a,b)上的两点P,Q ，计算P+Q；  cout << "3. 对在椭圆曲线Ep(a,b)上的两点P,Q ，计算P+Q" << endl;  int Rx = -1, Ry = -1;//设R=P+Q  if (e.isEllipticCurve(p, a, b))//是不是椭圆曲线  {  if (e.isPointOnCurve(p, a, b, Px, Py) && e.isPointOnCurve(p, a, b, Qx, Qy))//如果PQ不全在椭圆曲线上，不能计算P+Q。PQ都在才能进一步计算  {  e.addPoints(p, a, b, Px, Py, Qx, Qy, Rx, Ry);  cout << "点R=P+Q的坐标是(" << Rx << "," << Ry << ")" << endl;  }  }  //(4) 对在椭圆曲线Ep(a, b)上的点P，使用倍加 - 和算法计算mP  cout << "4. 对在椭圆曲线Ep(a, b)上的点P，使用倍加 - 和算法计算mP" << endl;  int R1x, R1y;//设R1=mP  int m;  if (e.isEllipticCurve(p, a, b))//是不是椭圆曲线  {  if (e.isPointOnCurve(p, a, b, Px, Py))//P在椭圆曲线上才能进一步计算  {  cout << "m= ";  cin >> m;  e.multiPoint(p, a, b, m, Px, Py, R1x, R1y);  cout << "点R1=mP的坐标是(" << R1x << "," << R1y << ")" << endl;  }  }    //(5) 对在椭圆曲线Ep(a,b)上的点 P，计算阶 ord(P)；  cout << "5. 对在椭圆曲线Ep(a,b)上的点 P，计算阶 ord(P)" << endl;  if (e.isEllipticCurve(p, a, b))//是不是椭圆曲线  {  if (e.isPointOnCurve(p, a, b, Px, Py))//P在椭圆曲线上才能进一步计算  {  int ord = e.calculate\_ord(p, a, b, Px, Py);  cout << "在椭圆曲线E" << p << "(" << a << "," << b << ")的点P的阶ord(P)为" << ord << endl;  }  }  //(6) 对在椭圆曲线Ep(a,b)，计算阶#E；  cout << "6.对在椭圆曲线Ep(a,b)，计算阶#E" << endl;  if (e.isEllipticCurve(p, a, b))//是不是椭圆曲线  {  int E = e.calculateCurveOrder(p, a, b);  cout << "在椭圆曲线E" << p << "(" << a << "," << b << ")的阶#E= " << E << endl;  }  //(7) 对在椭圆曲线Ep(a,b)，计算所有点；  cout << "7. 对在椭圆曲线Ep(a,b)，计算所有点" << endl;  if(e.isEllipticCurve(p, a, b))//判断是不是椭圆曲线  cout << "椭圆曲线E" << p << "(" << a << "," << b << ")的所有点为: " << endl;  e.calculateAllPoints(p, a, b);  return 0;  } |

* **说明部分：**

1. 给定参数p,a,b ，判断Ep(a,b)是否为椭圆曲线

计算(4a^3+27b^2)(mod p)的值，如果值为0，则不是椭圆曲线，如果值不为0，则是椭圆曲线。

1. 判断给定的点P,Q是否在椭圆曲线Ep(a,b)上

分别计算y^2(mod p)和x^3+ax+b(mod p)的值，如果两边相等，则点在椭圆曲线上

1. 对在椭圆曲线Ep(a,b)上的两点P,Q ，计算P+Q；

如果x1=x2，对应倍加，k=((3x1^2+a)/(2y1))(mod p)，需要提前求2y1的逆元。

如果x1!=x2，对应点加，k=((y2-y1)/(x2-x1))(mod p)，需要提前求(x2-x1)的逆元。

如果负数需要先取绝对值再取反。

x3=(k^2-x1-x2)(mod p)

y3=(k(x1-x3)-y1)(mod p)

如果是负数需要变成正数

1. 对在椭圆曲线Ep(a, b)上的点P，使用倍加 - 和算法计算mP

使用（3）的addPoints函数，进行多次加法运算，实现倍加

1. 对在椭圆曲线Ep(a,b)上的点 P，计算阶 ord(P)

椭圆曲线上点的阶的是另nP=O的最小整数，O表示无穷远点。当不是无穷远点时，进行倍加运算，直到到无穷远点结束。

1. 对在椭圆曲线Ep(a,b)，计算阶#E

椭圆曲线的阶是椭圆曲线上点的数目，求出椭圆曲线上点的数目，即为阶。

1. 对在椭圆曲线Ep(a,b)，计算所有点

对x和y取值计算y^2(mod p)和x^3+ax+b(mod p)的值，如果能有(x,y)满足y^2(mod p)和x^3+ax+b(mod p)值相等，则(x,y)在椭圆曲线上。无穷远点需要考虑在其中。

* **运行示例：**

