

In[1]:=

Модуль сложения точек;

```
EllipticAdd[p_, a_, b_, c_, P_List, Q_List] :=  
Module[{lam, x3, y3, P3},  
  программный модуль  
  Which[  
    условный оператор с множественными ветвями  
    P == {0}, Q,  
      О большое  
    Q == {0}, P,  
      О большое  
    P[[1]] != Q[[1]],  
      lam = Mod[(Q[[2]] - P[[2]]) PowerMod[Q[[1]] - P[[1]], p - 2, p], p];  
        остаток от деления    степень по модулю  
      x3 = Mod[lam^2 - a - P[[1]] - Q[[1]], p];  
        остаток от деления  
      y3 = Mod[-(lam (x3 - P[[1]]) + P[[2]]), p];  
        остаток от деления  
      {x3, y3},  
    (P == Q) ^ (P[[2]] == 0), {0},  
      О большое  
    (P == Q) ^ (P != {0}),  
      О большое  
      lam = Mod[(3 * P[[1]]^2 + 2 a * P[[1]] + b) PowerMod[2 P[[2]], p - 2, p], p];  
        остаток от деления    степень по модулю  
      x3 = Mod[lam^2 - a - P[[1]] - Q[[1]], p];  
        остаток от деления  
      y3 = Mod[-(lam (x3 - P[[1]]) + P[[2]]), p];  
        остаток от деления  
      {x3, y3},  
    (P[[1]] == Q[[1]]) ^ (P[[2]] != Q[[2]]), {0}]]  
      О большое
```

Модуль сложения точки N – раз (N – множитель PхN);

численное... численное приближение

In[10]:=

```
EllipticPointMultiply[p_, a_, b_, c_, Q_, n_] :=  
Module[{i = n - 1, q = Q, p1 = p, a1 = a, b1 = b, c1 = c},  
  программный модуль  
  pnt = q;  
  While[i > 0, i--, q = EllipticAdd[p1, a1, b1, c1, pnt, q]];  
  цикл-пока  
  q  
]  
P = {9, 4};  
a = 0;  
b = 6;  
c = 3;  
p = 11;  
EllipticPointMultiply[p, a, b, c, P, 5]
```

Out[16]= {0}