Gradient Descent run in cpp

```
Pick Real Model:
1. Linear: a*x + b
2. Square: a*x^2 + b*x + c
3. Manual input (x = 0 -> 10)
Choice: 2
Input Coefficients:
Input a: 3.53
Input b: 6.56
Input c: 0
```

```
a = 4.28492, b = 0.735548, c = 0.130815: error = 58.6806
runtime: 1501349
predict: 0.130815 | real: 0
predict: 5.15128 | real: 10.09
predict: 18.7416 | real: 27.24
predict: 40.9017 | real: 51.45
predict: 71.6317 | real: 82.72
predict: 110.932 | real: 121.05
predict: 158.801 | real: 166.44
predict: 215.241 | real: 218.89
predict: 280.25 | real: 278.4
predict: 353.829 | real: 344.97
```

Runtime = 1501349 microseconds = 1.501349 seconds

Flower Pollination run in rust

```
Input real A constant:
3.53
Input real B constant:
6.56
Input Model Choice
2
g_best_a: 3.5151363680225236 | g_best_b: 6.608203991203238 | obj: 0.11479454086713488
Elapsed time: 8.08s
predict: 10.123340359225761 | real: 10.09
predict: 27.27695345449657 | real: 27.24
predict: 51.460839285812426 | real: 51.45
predict: 82.67499785317332 | real: 82.72
predict: 120.91942915657927 | real: 121.05
predict: 166.1941331960303 | real: 166.44
predict: 218.4991099715263 | real: 218.89
predict: 277.83435948306743 | real: 278.4
predict: 344.1998817306536 | real: 344.97
```

Analysis: gradient descent seems to be a lot faster compared to flower pollination though it lacks the accuracy. Using Rust programming language may decrease the runtime in flower pollination so it is possible it might be slightly slower.