CS3102-code-Knn

#utec/CS3102

```
#include <iostream>
#include <vector>
#include <time.h>
#include <chrono>
#include <stdio.h>
#include <unistd.h>
using namespace std;



template <typename T>
class CartesianCoord{
  public:
  typedef T value_t;
  value_t x;
  value_t y;

  public:
  CartesianCoord(value_t _x = 0, value_t _y = 0)
  :x(_x),y(_y){
  }

  ~CartesianCoord(void){
  }

  template<typename _T>
  friend ostream& operator<<(ostream &out, const CartesianCoord<_T> c){
    out << "(" << c.x << "," << c.y << ")";
    return out;
  }
};
typedef CartesianCoord<int> coord_t;
```

```
vector<coord_t> knn ( int k, vector<coord_t> &points, const coord_t &q){
  // KNNS

  // KNNS
  return vector<coord_t>();
}




int main() {
  srand (time(NULL));

  vector<coord_t> points;
  for(int i=0; i< 1000; i++)
    points.push_back(coord_t(rand()%1000,rand()%1000));

  vector<coord_t>::iterator it = points.begin();
  for (; it != points.end(); it++){
    fflush(stdout);
    cout << "\r" << *it;
    usleep(2000);
  }
  cout << endl;

  vector<coord_t> knns;
  auto start = chrono::high_resolution_clock::now();
  knns = knn(3, points, coord_t(100,200));
  auto stop = chrono::high_resolution_clock::now();
  auto duration = chrono::duration_cast<chrono::microseconds>(stop - start);
  cout << "Time: " << endl;
  cout << duration.count() << endl;

  cout << "knns" << endl;
  vector<coord_t>::iterator kit = knns.begin();
  for (; kit != knns.end(); kit++)
    cout << *kit << endl;



}
```