```cpp
#include "PurePursuitPathGen.h"
#include <vector>
#include <string>
#include <fstream>
#include <iostream>
#include <algorithm>
#include <math.h>

PurePursuitPathGen::PurePursuitPathGen(double spacing, double a, double b, double tolerance,
std::vector<point> points, double max_vel, double max_accel, int k) {
    this->spacing = spacing;
    this->a = a;
    this->b = b;
    this->tolerance = tolerance;
    this->max_vel = max_vel;
    this->max_accel = max_accel;
    this->k = k;
    this->initial_points = points;
}

void PurePursuitPathGen::interpolate() {
    point vec;
    int mag;
    int num_points;
    final_points.push_back(initial_points[0]);
    for(int i = 1; i < initial_points.size(); i++) {


        vec.x = initial_points[i].x-initial_points[i-1].x;
        vec.y = initial_points[i].y-initial_points[i-1].y;

        mag = sqrt((vec.x*vec.x)+(vec.y*vec.y));

        num_points = ceil(mag/spacing);

        vec.x = (vec.x/mag)*spacing;
        vec.y = (vec.y/mag)*spacing;
        point new_vec;
        for(int j = 1; j < num_points; j++) {
            new_vec.x = (initial_points[i-1].x+(vec.x*j));
            new_vec.y = (initial_points[i-1].y+(vec.y*j));
            final_points.push_back(new_vec);
        }
        final_points.push_back(initial_points[i]);
    }
}
void PurePursuitPathGen::calc_distances() {
    final_points[0].distance = 0;
    for(int i = 1; i < final_points.size(); i++) {
        final_points[i].distance = final_points[i-1].distance + sqrt(pow(final_points[i].x-
final_points[i-1].x, 2)+pow(final_points[i].y-final_points[i-1].y, 2));
    }
}
void PurePursuitPathGen::calc_curvature() {
    final_points[0].curve = 0;
    double k1, k2, center1, center2, r, x1, x2, x3, y1, y2, y3;
    for(int i = 1; i < final_points.size()-1; i++) {
        x1 = final_points[i].x+0.001;
        x2 = final_points[i-1].x;
        x3 = final_points[i+1].x;

        y1 = final_points[i].y;
        y2 = final_points[i-1].y;
        y3 = final_points[i+1].y;

        k1=0.5*((x1*x1)+(y2*y2)-(x2*x2)-(y2*y2))/(x1-x2);
        k2 = (y1-y2)/(x1-x2);
        center2 = 0.5*((x2*x2)-(2*x2*k1)+(y2*y2)-(x3*x3)+(2*x3*k1)-(y3*y3))/(((x3*k2)-y3+y2-(x2*k2)));
        center1 = k1-k2*center2;
        r = sqrt((x1-center1)*(x1-center1) + (y1-center1)*(y1-center1));
        final_points[i].curve = 1/r;
    }
    final_points[final_points.size()-1].curve = 0;
}
void PurePursuitPathGen::smooth() {
    std::vector<point> copy;
    copy = final_points;
    double change = tolerance;
    while(change>=tolerance) {
        change = 0.0;
        for(int i = 1; i < final_points.size()-1; i++) {
            double aux = copy[i].x;
            copy[i].x += a*(final_points[i].x-copy[i].x) + b*(copy[i-1].x + copy[i+1].x-(2.0*(copy[i].x)));
            change+=abs(aux-copy[i].x);
            aux = copy[i].y;
            copy[i].y += a*(final_points[i].y-copy[i].y) + b*(copy[i-1].y + copy[i+1].y-(2.0*(copy[i].y)));
            change+=abs(aux-copy[i].y);
        }
    }
    final_points = copy;
}

void PurePursuitPathGen::calc_velocities() {
    for(int i = 0; i < final_points.size(); i++) {
        //std::cout << "Max vels" << k/final_points[i].curve << "\n";
        final_points[i].vel = std::min(max_vel, k/final_points[i].curve);
    }
    final_points[final_points.size()-1].vel = 0;
    for(int i = final_points.size()-2; i >=0; i--) {
        final_points[i].vel = std::min(final_points[i].vel, sqrt(pow(final_points[i+1].vel, 2)+2*max_accel*(final_points[i+1].distance-final_points[i].distance)));
    }
}

void PurePursuitPathGen::write_to_file() {
    std::ofstream fout;
    fout.open("path.txt");
    for(int i = 0; i < final_points.size(); i++) {
        fout << final_points[i].x << " "<< final_points[i].y << " " << final_points[i].vel << "\n";
    }
    fout.close();
```

```cpp
}
void PurePursuitPathGen::print_path() {
    printf("INITIAL\n");
    for(int i = 0; i < initial_points.size(); i++) {
        printf("%f %f\n", initial_points[i].x, initial_points[i].y);
    }
    printf("FINAL\n");
    for(int i = 0; i < final_points.size(); i++) {
        printf("%f %f\n", final_points[i].x, final_points[i].y);
    }
    printf("DISTANCE\n");
    for(int i = 0; i < final_points.size(); i++) {
      printf("%f\n", final_points[i].distance);
    }
    printf("CURVATURE\n");
    for(int i = 0; i < final_points.size(); i++) {
        printf("%f\n", final_points[i].curve);
    }
    printf("VELOCITIES\n");
    for(int i = 0; i < final_points.size(); i++) {
        printf("%f\n", final_points[i].vel);
    }

}

std::vector<point> PurePursuitPathGen::get_points() {
  return final_points;
}
```