```cpp
#include "Swarm.h"
#include <cstdlib>


Swarm::Swarm() {

}
void Swarm::UpdateSwarm(float time) {
    UpdateAcceleration();

    for (auto &b : *boids) {
        b.Velocity = (b.Velocity + b.Acceleration * time).SetMaxLength(MaxVelocity);
    }

}

void Swarm::UpdateAcceleration() {
    if (PerceptionRadius == 0) {
        PerceptionRadius = 1;
    }
    for (auto &b : *boids) {
        UpdateBoid(b);
     }
}

std::vector<boidInRange> Swarm::getBoidsInRange(const Boid &b) {
    std::vector<boidInRange> BoidsInRange;
    for (Boid &compare : *boids) {
        MyVector temp = b.Position - compare.Position;
        float distance = temp.length();
        if (distance <= PerceptionRadius && distance > 0) {
            if ((b.Velocity*-1).AngelBetweenVectors(temp) > BlindSpot){
                boidInRange bIR = boidInRange(&compare, (compare.Position - b.Position).normalize(), distance);
                BoidsInRange.push_back(bIR);
            }
        }
    }
    return BoidsInRange;
}

void Swarm::UpdateBoid(Boid &b) {
    std::vector<boidInRange> BoidsInRange = getBoidsInRange(b);
    MyVector steeringForce = MyVector(0, 0, 0);

    MyVector seperation = MyVector(0, 0, 0);
    MyVector alignment = MyVector(0, 0, 0);
    MyVector cohesion = MyVector(0, 0, 0);
    int numberOfBoidsInRange = 0;
    for (boidInRange &currentBoidInRange : BoidsInRange)
    {
        seperation = seperation + ( currentBoidInRange.boid->Position -
            b.Position ).normalize();
        alignment = alignment + currentBoidInRange.boid->Velocity.normalize
          ();
        cohesion = cohesion + currentBoidInRange.boid->Position;
        numberOfBoidsInRange++;
    }

    if (numberOfBoidsInRange != 0) {

        cohesion = cohesion / numberOfBoidsInRange;
        alignment = alignment / numberOfBoidsInRange;
    }
    else {
        cohesion = MyVector(0, 0, 0);
        alignment = MyVector(0, 0, 0);
    }
    cohesion = (b.Position - cohesion).normalize();

    alignment = alignment.normalize();
    seperation = seperation.normalize();

    MyVector wander = MyVector(0, 0, 0);
    wander.x = (float) rand();
    wander.y = (float)rand();
    wander.z = (float)rand();
    wander.normalize();

    steeringForce = wander * WanderWeight +
                    seperation * SeperationWeight +
                    alignment * AlignmentWeight +
                    cohesion * CohesionWeight;
    b.Acceleration = steeringForce.SetMaxLength(MaxAcceleration);
}


```