```
/* vector.h */
typedef struct VectorStruct vector;
struct VectorStruct {
        double angle;
        double length;
};

void scaleVector(vector *v, double alpha);
vector addVectors(vector a, vector b);


/* vector.c */
#include <stdio.h>
#include <math.h>
#include "vector.h"

int main(){
        vector a = {1.5,2}, b={2.6,3},c;
        printf("c's length is %f, c's angle is %f\n", c.length, c.angle);

        c = addVectors(a,b);
        printf("c's length is %f, c's angle is %f\n", c.length, c.angle);

        return 0;
}

//function scaleVector to scale the length of a vector
void scaleVector(vector *v, double alpha) {
        v->length *= alpha;
}

//function addVectors to add two vectors and return the result
vector addVectors(vector a, vector b) {
        vector c;
        double ax = a.length*cos(a.angle);
        double bx = b.length*cos(b.angle);
        double ay = a.length*sin(a.angle);
        double by = b.length*sin(b.angle);
        double cx = ax+bx;
        double cy = ay+by;
        c.length = sqrt(cx*cx+cy*cy);
        c.angle = acos( cx/c.length );

        return c;
}
```

```
//vector.hpp
class vector {  // class is generalization of struct
  public: // public identifies the 'public exposure' of this class to the 'outside'
  // functions that build instances, so-called 'constructors'
  vector(double a, double l);
  vector();
  //function that delete instances, so-called 'destructor'
  ~vector();

  // functions tied to variables of this class
   double getAngle();
   double getLength();
   void scale(double a);
   vector add(vector b);

   private:  // default access specifier, compile throws error if direct access is attempted
   double angle_, length_;
};


//vector.cpp
#include <iostream>     // used for IO
#include <cmath>
#include "vector.hpp"

using namespace std;    // creates textual container for variables and functions

// functions that build instances, so-called 'constructors'
vector::vector(double a, double l) { angle_ = a; length_ = l; }

vector::vector() { angle_ =0.0; length_ = 0.0; }   // notice "overloading" of function name

// function that delete instances, so-called 'destructor'
vector:: ~vector() { cout << "vector object destructed" << endl;}

// functions tied to variables of this class
double vector::getAngle() { return angle_; }

double vector::getLength() { return length_; }

void vector::scale(double a) { length_ *= a; }   // extend or contract vector

vector vector::add(vector b) {  // produce new vector by adding argument to variable
        vector c;
        double ax = length_*cos(angle_);
        double bx = b.length_*cos(b.angle_);
        double ay = length_*sin(angle_);
        double by = b.length_*sin(b.angle_);
        double cx = ax+bx;
        double cy = ay+by;
        c.length_ = sqrt(cx*cx+cy*cy);
        c.angle_ = acos(cx/c.length_);

        return c;
   }

int main(){
 vector c(1.5,2);
 vector d(2.6,3);
 vector e = c.add(d);
 cout << "e's length is " << e.getLength() << " and e's angle is " << e.getAngle() << endl;

 return 0;
}
```