```
//
//  main.c
//  testSParameters
//
//  Created by John Mac on 9/18/16.
//  Copyright © 2016 John Wetters. All rights reserved.
//



#define pi 4*atan(1.0)
#define degToRadian  pi/180
//#define radianToDegree 180/pi



#include <stdio.h>
#include <math.h>
//#include "utils.h"

struct magAng {
    double mag;
    double ang;
};

struct realImag {
    double real;
    double imag;
};


struct  sParameter{
    struct magAng s[3][3];
};

/*mag ang to rectilinear*/
struct realImag magToReal(struct magAng a)
{
    struct realImag out;
    out.real = a.mag*cos(a.ang*degToRadian);
    out.imag = a.mag*sin(a.ang*degToRadian);
    return out;
}

/*rectilinear to mag ang*/
struct magAng realToMag(struct realImag a)
{
    double pi1,radianToDegree1;
    pi1=4.0*atan(1.0);
    radianToDegree1=180.0/pi1;
    struct magAng out;
    out.mag = sqrt(pow(a.real,2)+pow(a.imag,2));
    out.ang = atan2(a.imag,a.real)*radianToDegree1;
    return out;
}
```

```
/*s1-s2*/
struct magAng s_subtract(struct magAng a,struct magAng b)
{
    /*convert to realImag,subtract and convert to mag ang*/
    struct magAng out;
    struct realImag a1,b1,c1;
    //convert to realImag
    a1=magToReal(a);
    b1=magToReal(b);
    //subtract
    c1.real=a1.real-b1.real;
    c1.imag=a1.imag-b1.imag;
    //convert to magAng
    out=realToMag(c1);
    return out;
}

/*s1*s1*/
struct magAng s_multiply(struct magAng a,struct magAng b)
{
    struct magAng out;

    out.mag = a.mag*b.mag;
    out.ang= b.ang+a.ang;
    return out;
}

/*s1/s2*/
struct magAng s_divide(struct magAng a,struct magAng b)
{
    struct magAng out;

    out.mag = a.mag/b.mag;
    out.ang= a.ang-b.ang;
    return out;
}

/* s* complex conjugate*/
struct magAng s_complexConjugate(struct magAng a)
{
    struct magAng out;

    out.mag = a.mag;
    out.ang= -a.ang;
    return out;
}


int main(int argc, const char * argv[]) {
    // insert code here...
    printf("Hello, World!\n");

    int i,j;
    /***************test magAng*******************/
    struct magAng vector;
    vector.mag=1.0;
```

```
vector.ang=45;

printf("test mag= %e\n",vector.mag);
printf("test ang= %e\n",vector.ang);

/****************test realToMag ***************/
printf("***********test realToMag **********\n");


struct realImag test;
struct magAng test1;
test.real=1.0;
test.imag=1.0;

test1=realToMag(test);

printf("test1 mag= %e\n",test1.mag);
printf("test1 ang= %e\n",test1.ang);

printf("***********test realToMag **********\n");

/****************test sParameter subtraction *******/
struct magAng vector1;
vector1.mag=2.0;
vector1.ang=45;


struct magAng sub_test;

sub_test=s_subtract(vector1,vector);
printf("test magnitude= %e\n",sub_test.mag);
printf("test angle= %e\n",sub_test.ang);


/****************test magToReal *******/


struct realImag rect;
rect.real=magToReal(vector).real;
rect.imag=magToReal(vector).imag;

printf("test real= %e\n",rect.real);
printf("test imag= %e\n",rect.imag);

struct sParameter matrix;
matrix.s[1][1].mag=.641;
matrix.s[1][1].ang=-171.3;

printf("test s11 mag= %e\n",matrix.s[1][1].mag);
printf("test s11 ang= %e\n",matrix.s[1][1].ang);

matrix.s[1][2].mag=.057;
matrix.s[1][2].ang=16.3;

matrix.s[2][1].mag=2.058;
matrix.s[2][1].ang=28.5;
```

```
matrix.s[2][2].mag=.572;
matrix.s[2][2].ang=-95.7;

for (i=1; i<3 ; i++){
    for (j=1; j<3 ; j++){
        printf("s%d,%d mag= %e\n",i,j,matrix.s[i][j].mag);
        printf("s%d,%d angle= %e\n",i,j,matrix.s[i][j].ang);
    }}

/**********************Calculate delta****p.99*******************/

struct magAng s11s22;
struct magAng s12s21;
struct magAng delta;

s11s22=s_multiply(matrix.s[1][1],matrix.s[2][2]);
printf("s11s22 magnitude= %e\n",s11s22.mag);
printf("s11s22 angle= %e\n",s11s22.ang);
s12s21=s_multiply(matrix.s[1][2],matrix.s[2][1]);
printf("s12s21 magnitude= %e\n",s12s21.mag);
printf("s12s21 angle= %e\n",s12s21.ang);
delta=s_subtract(s11s22,s12s21);
printf("delta magnitude= %e\n",delta.mag);
printf("delta angle= %e\n",delta.ang);

/**********************Calculate K***************************/

struct magAng s11Sq;
struct magAng s22Sq;
double K;
double deltaSq;
deltaSq=pow(delta.mag,2.0);

s11Sq=s_multiply(matrix.s[1][1],matrix.s[1][1]);
printf("s11Sq mag= %e\n",s11Sq.mag);
printf("test s22 mag= %e\n",matrix.s[2][2].mag);
s22Sq=s_multiply(matrix.s[2][2],matrix.s[2][2]);
printf("s22Sq mag= %e\n",s22Sq.mag);
K=(1-s11Sq.mag-s22Sq.mag+deltaSq)/(2.0*s12s21.mag);
printf("s11Sq mag= %e\n",s11Sq.mag);
printf("s22Sq mag= %e\n",s22Sq.mag);
printf("deltaSq= %e\n",deltaSq);
printf("s12s21 mag= %e\n",s12s21.mag);
printf("K= %e\n",K);

/**********************Calculate B1********p.113***************/

double B1;
B1=1+s11Sq.mag+-s22Sq.mag-deltaSq;
printf("B1= %e\n",B1);

/**********************Calculate B2********p.113***************/
double B2;
B2=1+s22Sq.mag+-s11Sq.mag-deltaSq;
printf("B2= %e\n",B2);
```

```
/**********************Calculate C1********p.113****************/
struct magAng C1,delta_s22Product,s22conjugate;
s22conjugate=s_complexConjugate(matrix.s[2][2]);

printf("s22conjugate magnitude= %e\n",s22conjugate.mag);
printf("s22conjugate angle= %e\n",s22conjugate.ang);

delta_s22Product=s_multiply(delta,s22conjugate);
C1=s_subtract(matrix.s[1][1],delta_s22Product);
printf("C1 magnitude= %e\n",C1.mag);
printf("C1 angle= %e\n",C1.ang);

/**********************Calculate C2********p.113****************/

struct magAng C2,delta_s11Product,s11conjugate;

s11conjugate=s_complexConjugate(matrix.s[1][1]);
delta_s11Product=s_multiply(delta,s11conjugate);
C2=s_subtract(matrix.s[2][2],delta_s11Product);
printf("C2 magnitude= %e\n",C2.mag);
printf("C2 angle= %e\n",C2.ang);

/**********************Calculate gama Ms********p.113****************/
struct magAng gamaMs,C1sq,numerator,twoC1;
double FourC1sq,sqRootB1;
C1sq=s_multiply(C1,C1);
FourC1sq=4.0*C1sq.mag;
sqRootB1=sqrt(B1-FourC1sq);
numerator.mag=B1-sqRootB1;
numerator.ang=0;
twoC1.mag=2*C1.mag;
twoC1.ang=C1.ang;
gamaMs=s_divide(numerator,twoC1);
printf("gamaMs magnitude= %e\n",gamaMs.mag);
printf("gamaMs angle= %e\n",gamaMs.ang);

/**********************Calculate gama ML********p.113****************/
struct magAng gamaML,C2sq,twoC2;
double FourC2sq,sqRootB2;
C2sq=s_multiply(C2,C2);
FourC2sq=4.0*C2sq.mag;
sqRootB2=sqrt(pow(B2,2)-FourC2sq);
numerator.mag=B2-sqRootB2;
numerator.ang=0;
twoC2.mag=2*C2.mag;
twoC2.ang=C2.ang;
gamaML=s_divide(numerator,twoC2);
printf("gamaML magnitude= %e\n",gamaML.mag);
printf("gamaML angle= %e\n",gamaML.ang);
```

```
    return 0;
}
```