Universidad de Chile
Facultad de Ciencias
Departamento de Física

Programación y Métodos Numéricos

Corrección tarea Nº 11
Publicada el 19 de Octubre de 2006

Profesor: José Rogan
Ayudantes: María Daniela Cornejo
Max Ramírez
Alejandro Varas

1. El archivo `.h`

```
#include <iostream>
#include <cmath>

using namespace std;

class Complejo
{
 private:
  double real;
  double imaginaria;
 public:
  Complejo(double,double);
  ~Complejo();
  void setreal(double);
  void setimag(double);
  double getreal();
  double getimag();
  double modulo();
  double fase();
  Complejo conjugado();
  friend ostream & operator << (ostream &os, Complejo);
};

Complejo operator + (Complejo, Complejo);
Complejo operator - (Complejo, Complejo);
Complejo operator * (Complejo, Complejo);
Complejo operator / (Complejo, Complejo);
```

y su implementación

```
#include <iostream>
#include "classComplejos.h"
using namespace std;
```

```
//Constructores.
Complejo::Complejo (double x, double y):
real(x), imaginaria(y)
{}

Complejo::~Complejo()
{}

//Funciones.
void Complejo::setreal(double x){real=x;}

void Complejo::setimag(double y){imaginaria=y;}

double Complejo::getreal(){return real;}

double Complejo::getimag(){return imaginaria;}

double Complejo::modulo(){
 return sqrt(real*real+imaginaria*imaginaria);
}

Complejo Complejo::conjugado(){
 return Complejo (getreal(),(-1)*getimag());
}

double Complejo::fase(){
 return atan(imaginaria/real);
}

//Operadores.
Complejo operator + (Complejo z, Complejo w){
 return Complejo(z.getreal() + w.getreal(),z.getimag() +w.getimag());
}

Complejo operator - (Complejo z, Complejo w){
 return Complejo(z.getreal() - w.getreal(),z.getimag() - w.getimag());
}

Complejo operator * (Complejo z, Complejo w){
 return Complejo((z.getreal()*w.getreal())-(z.getimag()*w.getimag()),
 (z.getreal()*w.getimag())+(w.getreal()*z.getimag()));
}

Complejo operator / (Complejo z, Complejo w){
 double a=w.modulo()*w.modulo();
 return Complejo(((z.getreal()*w.getreal())+(z.getimag()*w.getimag()))/a,
((w.getreal()*z.getimag())-(z.getreal()*w.getimag()))/a);
}
```

```
ostream &operator << (ostream &os, Complejo z){
 os << z.getreal() << " +" << z.getimag() << "i";
 return os;
}
```

2.
```
#include <iostream>
#include <cmath>
#include "classComplejos.h"

using namespace std;

Complejo f(Complejo,Complejo,Complejo,Complejo);
Complejo fderivada(Complejo,Complejo,Complejo);
Complejo sol(Complejo);

int main(){

   double r,i;

   cout.precision(10);
   cout << "Ingrese una semilla: " << endl;
   cout << "real: ";
   cin >> r;
   cout << "imaginaria: ";
   cin >> i;

   Complejo x(r,i);
   cout << "El resultado es: " << sol(x) << endl;

   return 0;
}

Complejo sol(Complejo x)
{
   Complejo a(1,0);
   Complejo b((-1)*sqrt(2.0),(-1)*sqrt(2.0));
   Complejo c(0,2);

   Complejo dx(1e3,1e3);
   double epsilon=1e-8;
   while (dx.modulo()> epsilon){
      x=x-f(a,b,c,x)/fderivada(a,b,x);
      dx=a*x*x+b*x+c;
   }
   return x;
}
```

```
Complejo f(Complejo a, Complejo b, Complejo c, Complejo x)
{
   return a*x*x+b*x+c;
}

Complejo fderivada(Complejo a, Complejo b, Complejo x)
{
   return Complejo(2,0)*a*x+b;
}
```

3.
```
#include <iostream>
#include <cmath>
#include "Matrix.h"

using namespace std;

Matrix::Matrix(Complejo z1,Complejo z2,Complejo z3,Complejo z4)
:a11(z1), a12(z2), a21(z3), a22(z4)
{}

Matrix::~Matrix()
{}

void Matrix::set(int x,int y, Complejo z)
{
   if(x==1&&y==1) a11=z;
   else if(x==1&&y==2) a12=z;
   else if(x==2&&y==1) a21=z;
   else if(x==2&&y==2) a22=z;
}

Complejo Matrix::get(int x, int y) const
{
   Complejo z = 0;
   if(x==1&&y==1) z = a11;
   else if(x==1&&y==2) z = a12;
   else if(x==2&&y==1) z = a21;
   else if(x==2&&y==2) z = a22;
   return z;
}

Matrix operator + (const Matrix &A, const Matrix &B)
{
   return Matrix(A.get(1,1)+B.get(1,1),A.get(1,2)+B.get(1,2),
             A.get(2,1)+B.get(2,1),A.get(2,2)+B.get(2,2));
}

Matrix operator - (const Matrix &A, const Matrix &B)
```

```
{
   return Matrix(A.get(1,1)-B.get(1,1),A.get(1,2)-B.get(1,2),
                           A.get(2,1)-B.get(2,1),A.get(2,2)-B.get(2,2));
}

Matrix operator * (const Matrix &A, const Matrix &B)
{
   return Matrix((A.get(1,1)*B.get(1,1))+(A.get(1,2)*B.get(2,1)),
                         (A.get(1,1)*B.get(1,2))+(A.get(1,2)*B.get(2,2)),
                         (A.get(2,1)*B.get(1,1))+(A.get(2,2)*B.get(2,1)),
                         (A.get(2,1)*B.get(1,2))+(A.get(2,2)*B.get(2,2)));
}

bool operator == (const Matrix &A, const Matrix &B)
{
  if (A.get(1,1)==B.get(1,1) && A.get(1,2)==B.get(1,2) &&
         A.get(2,1)==B.get(2,1) && A.get(2,2)==B.get(2,2)){
return true;
}
  else return false;
}

Complejo Matrix::determinante()
{
   return (a11*a22)+(-1)*(a12*a21);
}

Matrix Matrix::inversa()
{
   Complejo d=determinante();
   return Matrix(a22/d,(-1)*(a12/d),(-1)*(a21/d),a11/d);
}

Complejo Matrix::traza()
{
   return Complejo(a11*a22);
}

Matrix Matrix::hermitica()
{
   return Matrix(a11.conjugado(), a21.conjugado(),
                        a12.conjugado(), a22.conjugado());
}

bool Matrix::es_hermitica()
{
   Matrix h=hermitica();
   if(get(1,1)==h.get(1,1) && get(1,2)==h.get(1,2) &&
```

```
        get(2,1)==h.get(2,1) && get(2,2)==h.get(2,2)) return true;
    else return false;
}

ostream & operator << (ostream & os, Matrix & A)
{
    os<<"\t/"<<" "<<A.get(1,1)<<"  "<<A.get(1,2)<<" "<<"\\ \n"<<
    "\t\\"<<" "<<A.get(2,1)<<"  "<<A.get(2,2)<<" "<<"/"<<endl;
    return os;
}

Matrix conmutador (const Matrix &A, const Matrix &B)
{
    return ((A*B)-(B*A));
}
```

4. Las declaraciones

```
#ifndef _hermitica_h
#define _hermitica_h

#include <iostream>
#include <cmath>

class Hermitica : public Matrix
{
 public:
    Hermitica(Complejo,Complejo,Complejo,Complejo);
    Hermitica(Matrix);
    Complejo menor_valorp();
    Complejo mayor_valorp();
};

#endif
```

y sus implementaciones

```
#include <iostream>
#include <cmath>
#include "Complejo.h"
#include "Matrix.h"
#include "hermitica.h"

using namespace std;

Hermitica::Hermitica(Complejo a, Complejo b, Complejo c, Complejo d )
  :Matrix(a,b,c,d)
{}
```

```
Hermitica::Hermitica(Matrix A)
  :Matrix(A.get(1,1),A.get(1,2),A.get(2,1),A.get(2,2))
{}

Complejo Hermitica::menor_valorp()
{
   Complejo R=((get(1,1)+get(2,2))*(get(1,1)+get(2,2)))-
                      (4*(get(1,1)*get(2,2)-get(1,2)*get(2,1)));
   Complejo i=(get(1,1)+get(2,2));
   double V1=((i.getreal())-sqrt(R.getreal()))/2.0;
   double V2=((i.getreal())+sqrt(R.getreal()))/2.0;
   if(V1>V2) return V2;
   else if(V2>V1) return V1;
   else return V1;
}

Complejo Hermitica::mayor_valorp()
{
   Complejo R=((get(1,1)+get(2,2))*(get(1,1)+get(2,2)))-
                       (4*(get(1,1)*get(2,2)-get(1,2)*get(2,1)));
   Complejo i=(get(1,1)+get(2,2));
   double V1=((i.getreal())-sqrt(R.getreal()))/2.0;
   double V2=((i.getreal())+sqrt(R.getreal()))/2.0;
   if(V1>V2) return V1;
   else if(V2>V1) return V2;
   else return V1;
}
```