

PRÁCTICA 1

Familiarización con el entorno

En esta práctica vamos a familiarizarnos con el entorno de C++ Builder creando nuestro primer programa orientado a objetos. Para ello vamos a utilizar una clase que representa a los números complejos y que incorpora las operaciones más comunes con dichos números.

Primero hay que hacer notar que los números complejos ya están incorporados en una de las bibliotecas estándar de C++ de modo que en un caso real no necesitaríamos realizar dicha implementación. Para ver su definición, uso y operaciones no hay más que consultar la ayuda que tiene incorporada el C++ Builder. No obstante es un caso interesante ya que habéis visto su implementación como estructura en clase de POO de modo que sin tener que abordar un nuevo problema podemos familiarizarnos con los aspectos de encapsulación y separación entre la interfaz y la implementación.

Hemos tomado de la red una implementación de los números complejos que vamos a compilar, ejecutar y a probar incorporando un programa principal en que se creen objetos de tipo complejo y se prueben sus operaciones. Lo primero que deberemos hacer es comprender que es lo que hace el programa cambiando los identificadores por otros más significativos y formateándolo y documentándolo de forma adecuada (según lo explicado en clase).

```
/*Complex number class 1.12*/
```

```
by Deepak <deepak-p@eth.net>
```

```
Released into the public domain on  
the 11th day of September 2001.
```

```
Please send me the improved versions
```

```
Looking forward to hear from you.
```

```
*****/
```

```
# include <iostream.h>
```

```
# include <math.h>
```

```
# define PI 3.1415926535
```

```
int ta,tb;
```

```
class complex
```

```
{
```

```
private:
```

```
float a,b;
```

```
float mag,angle;
```

```
public:
```

```
complex();//constructor 1
```

```
complex(float,float);//constructor 2
```

```
void euler(float,float);
```

```
void filleuler();
```

```
void fillcart();
```

```
complex operator+(complex);//Overloaded + operator for c=a+b etc.,
```

```
complex operator+(float);
```

```
complex operator-(complex);//Overloaded - operator for c=a-b etc.,
```

```
complex operator-(float);
```

```
friend complex operator+(float,complex);
```

```
friend complex operator-(float,complex);
```

```
complex operator*(complex);
```

```
complex operator*(float);
```

```
friend complex operator*(float,complex);
```

```
complex recip();
```

```
friend complex operator/(complex,complex);
```

```
friend complex operator/(float,complex);
```

```
friend complex operator/(complex,float);
```

```
operator float();
```

```
complex power(float);
```

```
void operator+=(float);
```

```
void operator-=(float);
```

```
void operator+=(complex);
```

```

void operator==(complex);
void operator==(float);
void operator*(complex);
void operator/=(complex);
void operator/=(float);
void operator++();
void operator--();
int operator==(complex);
int operator>(complex);
int operator<(complex);
int operator!=(complex);
void getdata(float *,float *);
void geteulerdata(float *,float *);
friend ostream& operator<<(ostream&,complex);
void displayeuler();
};
ostream& operator<<(ostream& out,complex x){out<<x.a<<" +i" <<x.b;return out;}
void complex::displayeuler(){cout<<x.mag<<" cis" <<x.angle<<" deg";}
complex::complex(){a=0;b=0;filleuler();}
complex::complex(float x,float y){a=x;b=y;filleuler();}
void complex::euler(float x,float y){mag=x;angle=y;fillcart();}
complex complex::operator+(float x){return complex(x+a,b);}
complex complex::operator-(float x){return complex(x-a,b);}
complex operator*(float x,complex y){return complex(x*y.a,x*y.b);}
complex complex::operator*(float x){return complex(a*x,b*x);}
complex complex::operator*(complex x){return complex(a*x.a-b*x.b,b*x.a+a*x.b);}
complex complex::recip(){return complex(a/(sqrt(a*a+b*b)),(-1)*b/(sqrt(a*a+b*b)));}
complex operator/(complex x,complex y){return (x*y.recip());}
complex operator/(complex x,float y){return complex(x.a/y,x.b/y);}
complex operator/(float x,complex y){return (x*y.recip());}
complex::operator float(){return (float)(sqrt(a*a+b*b));}
complex complex::operator+(complex x){return complex(a+x.a,b+x.b);}
complex operator+(float x,complex y){return complex(x+y.a,y.b);}
complex complex::operator-(complex x){return complex(a-x.a,b-x.b);}
complex operator-(float x,complex y){return complex(x-y.a,0-y.b);}
complex complex::power(float x){complex temp;temp.euler((float)(pow(mag,x)),x*angle);return temp;}
void complex::operator+=(float x){a+=x;filleuler();}
void complex::operator+=(complex x){a+=x.a;b+=x.b;filleuler();}
void complex::operator-=(float x){a-=x;filleuler();}
void complex::operator-=(complex x){a-=x.a;b-=x.b;filleuler();}
void complex::operator*=(float x){a*=x;b*=x;filleuler();}
void complex::operator/=(float x){a/=x;b/=x;filleuler();}
void complex::operator*=(complex x){ta=(a*x.a-b*x.b);tb=(b*x.a+a*x.b);a=ta;b=tb;filleuler();}
void complex::operator/=(complex x){ta=((a*x.a+b*x.b)/(sqrt(x.a*x.a+x.b*x.b)));tb=((b*x.a-a*x.b)/(sqrt(x.a*x.a+x.b*x.b)));a=ta;b=tb;filleuler();}
void complex::operator++(){a++;b++;filleuler();}
void complex::operator--(){a--;b--;filleuler();}
int complex::operator==(complex x){if(x.a==a && x.b==b){return 1;}else return 0;}
int complex::operator!=(complex x){if(x.a==a && x.b==b){return 0;}else return 1;}
int complex::operator>(complex x){if(mag>x.mag){return 1;}else return 0;}
int complex::operator<(complex x){if(mag<x.mag){return 1;}else return 0;}
void complex::getdata(float *x,float *y){*x=a;*y=b;}
void complex::geteulerdata(float *x,float *y){*x=mag;*y=angle;}
void complex::filleuler()
{
    mag=(float)(sqrt(a*a+b*b));
    if(mag==0){angle=0;}
    else
    {
        angle=(float)(acos(a/mag));
    }
}

```

```
        angle*=(180/(PI));
    }
}
void complex::fillcart()
{
    a=(float)(mag*(cos(angle*PI/180)));
    b=(float)(mag*(sin(angle*PI/180)));
}
```

Memoria de la práctica

La memoria de esta practica consiste en el código debidamente modificado con comentarios y en el programa principal con los casos de uso.

Se recuerda a todos los alumnos la necesidad de almacenar las practicas en 2 disquetes diferentes. Uno de trabajo y otro de seguridad que debe mantenerse actualizado. De este modo con 4 disquetes por grupo no es posible perder la practica.