**Construcción de un ADT**

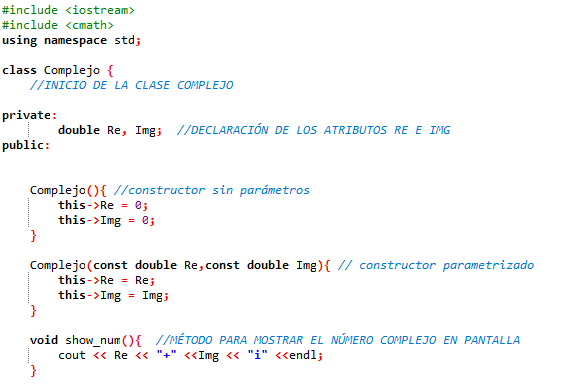
**Nivel Lógico**

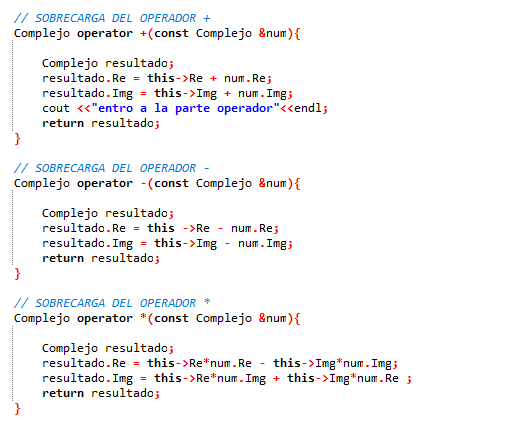
* Elementos que almacenará la estructura Complejo:
  + Atributo “Re”, que se refiera a la parte Real del número complejo.
  + Atributo “Img”, que se refiere a la parte Imaginaria del número complejo.
* Tipo de organización de los elementos:
  + Sin relación

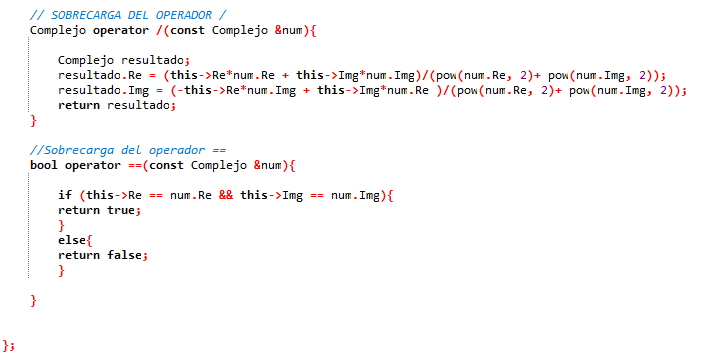
**Nivel Físico**

* **Nombre de la operación: “ + ” SUMA**
* **Descripción:** Suma de dos números complejos
* **Datos de entrada:** Dos objetos de clase Complejo
* **Datos de salida:** Un objeto de clase Complejo
* **Precondiciones:** Que las dos entradas sean tipo Complejo con su parte Re e Img definidas
* **Post condiciones:** Que la salida sea otro objeto de clase Complejo con su parte Re e Img definidas, y que sea igual a la suma de los dos números de entrada
* **Nombre de la operación: “ - ” RESTA**
* **Descripción:** Resta de dos números complejos
* **Datos de entrada:** Dos objetos de clase Complejo
* **Datos de salida:** Un objeto de clase Complejo
* **Precondiciones:** Que las dos entradas sean tipo Complejo con su parte Re e Img definidas
* **Post condiciones:** Que la salida sea otro objeto de clase Complejo con su parte Re e Img definidas, y que sea igual a la resta de los dos números de entrada
* **Nombre de la operación: “ \* ” MULTIPLICACIÓN**
* **Descripción:** Multiplicación de dos números complejos
* **Datos de entrada:** Dos objetos de clase Complejo
* **Datos de salida:** Un objeto de clase Complejo
* **Precondiciones:** Que las dos entradas sean tipo Complejo con su parte Re e Img definidas
* **Post condiciones:** Que la salida sea otro objeto de clase Complejo con su parte Re e Img definidas, y que sea igual a la multiplicación de los dos números de entrada
* **Nombre de la operación: “ / ” DIVISIÓN**
* **Descripción:** División de dos números complejos
* **Datos de entrada:** Dos objetos de clase Complejo
* **Datos de salida:** Un objeto de clase Complejo
* **Precondiciones:** Que las dos entradas sean tipo Complejo con su parte Re e Img definidas
* **Post condiciones:** Que la salida sea otro objeto de clase Complejo con su parte Re e Img definidas, y que sea igual a la división de los dos números de entrada
* **Nombre de la operación: “ = = ” Igualdad**
* **Descripción:** Compara si dos números complejos son iguales
* **Datos de entrada:** Dos objetos de clase Complejo
* **Datos de salida:** Una variable de tipo booleana.
* **Precondiciones:** Que las dos entradas sean tipo Complejo con su parte Re e Img definidas
* **Post condiciones:** Que la salida sea tipo booleana. Valor True si la parte real del primer número es igual a la parte real del segundo número, así como la parte imaginaria del primer número es igual a la parte Imaginaria del segundo número. Caso contrario es False.

**APLICACIÓN EN C++**







**Código para copiar directamente:**

**#include <iostream>**

**#include <cmath>**

**using namespace std;**

**class Complejo {**

**//INICIO DE LA CLASE COMPLEJO**

**private:**

**double Re, Img; //DECLARACIÓN DE LOS ATRIBUTOS RE E IMG**

**public:**

**Complejo(){ //constructor sin parámetros**

**this->Re = 0;**

**this->Img = 0;**

**}**

**Complejo(const double Re,const double Img){ // constructor parametrizado**

**this->Re = Re;**

**this->Img = Img;**

**}**

**void show\_num(){ //MÉTODO PARA MOSTRAR EL NÚMERO COMPLEJO EN PANTALLA**

**cout << Re << "+" <<Img << "i" <<endl;**

**}**

**// SOBRECARGA DEL OPERADOR +**

**Complejo operator +(const Complejo &num){**

**Complejo resultado;**

**resultado.Re = this->Re + num.Re;**

**resultado.Img = this->Img + num.Img;**

**cout <<"entro a la parte operador"<<endl;**

**return resultado;**

**}**

**// SOBRECARGA DEL OPERADOR -**

**Complejo operator -(const Complejo &num){**

**Complejo resultado;**

**resultado.Re = this ->Re - num.Re;**

**resultado.Img = this->Img - num.Img;**

**return resultado;**

**}**

**// SOBRECARGA DEL OPERADOR \***

**Complejo operator \*(const Complejo &num){**

**Complejo resultado;**

**resultado.Re = this->Re\*num.Re - this->Img\*num.Img;**

**resultado.Img = this->Re\*num.Img + this->Img\*num.Re ;**

**return resultado;**

**}**

**// SOBRECARGA DEL OPERADOR /**

**Complejo operator /(const Complejo &num){**

**Complejo resultado;**

**resultado.Re = (this->Re\*num.Re + this->Img\*num.Img)/(pow(num.Re, 2)+ pow(num.Img, 2));**

**resultado.Img = (-this->Re\*num.Img + this->Img\*num.Re )/(pow(num.Re, 2)+ pow(num.Img, 2));**

**return resultado;**

**}**

**//Sobrecarga del operador ==**

**bool operator ==(const Complejo &num){**

**if (this->Re == num.Re && this->Img == num.Img){**

**return true;**

**}**

**else{**

**return false;**

**}**

**}**

**};**