```cpp
#pragma once
#include <iostream>
#include "math.h"

class Complex
{
	double a,
	double b;
public:
/************************* constructori ************************************/
	//constructor de initializare
	Complex(double x=0.0, double y=0.0):a(x), b(y){};

	//constructor de copiere
	Complex(const Complex &c)
	{
		a = c.a;
		b = c.b;
	}

/************************* supraincarcare operatori  ************************/
	//supraincarcarea operatorului = (egal) printr-o functie membra a clasei
	Complex& operator=(const Complex &c)
	{
		a = c.a;
		b = c.b;
		return *this;
	}

//*	//supraincarcarea operatorului + (plus) printr-o functie membra a clasei
	Complex operator+(const Complex &c)
	{
		Complex tmp(*this);
		tmp.a += c.a;
		tmp.b += c.b;
		return tmp;
	}

	//supraincarcarea operatorului - (minus) printr-o functie prietena globala
	//(nemembra a clasei)
	friend Complex operator-(const Complex &a, const Complex &b)
	{
		Complex tmp(a);
		tmp.a -= b.a;
		tmp.b -= b.b;
		return tmp;
	}
//*   //supraincarcarea operatorului prefixat ++ (incrementare) printr-o functie
      //membra a clasei
      Complex& operator++(void)
      {
            a += 1.0;
            b += 1.0;
            return *this;
      }
      //supraincarcarea operatorului prefixat -- (decrementare) printr-o functie
      //prietena globala (nemembra a clasei)
      friend Complex& operator--(Complex &c)
      {
            c.a -= 1.0;
            c.b -= 1.0;
            return c;
      }

//*   //supraincarcarea operatorului postfixat ++ (incrementare) printr-o functie
      //membra a clasei
      Complex operator++(int)
      {
            Complex tmp(*this);
            a += 1.0;
            b += 1.0;
            return tmp;
      }
      //supraincarcarea operatorului postfixat -- (decrementare) printr-o functie
      //prietena globala (nemembra a clasei)
      friend Complex operator--(Complex &c, int)
      {
            Complex tmp(c);
            c.a -= 1.0;
            c.b -= 1.0;
            return c;
      }

//*   //supraincarcarea operatorului double() de conversie de tip
      operator double() const
      {
            return sqrt(a*a + b*b);
      }

//*   //supraincarcarea operatorului de intrare
      friend std::istream& operator>>(std::istream& input, Complex &c)
      {
            return input >> c.a >> c.b;
      }
      //supraincarcarea operatorului de iesire
      friend std::ostream& operator<<(std::ostream& output, Complex &c)
      {
            return output << c.a << "+i" << c.b;
      }
      //destructor
      ~Complex(void){};
};
```