

## Classe dei numeri complessi

1. Introduzione e requisiti del problema
2. Specifica
3. Progetto della soluzione
4. Codifica

## Requisiti del problema

Si definisca una classe C++ `numero_complesso` che consenta di:

1. creare un numero complesso a partire da parte intera e parte immaginaria
2. creare un numero complesso a partire da modulo e argomento (si consideri come modulo un numero reale e come argomento un numero intero variabile tra 0 e 360 che rappresenti la misura dell'arco in gradi)
3. stampare un numero complesso secondo la rappresentazione algebrica
4. stampare un numero complesso secondo la rappresentazione polare
5. sommare e moltiplicare numeri complessi con notazione infissa (ad es:  $X+Y$ )
6. verificare l'uguaglianza tra numeri complessi
7. effettuare l'assegnamento tra numeri complessi

Un numero complesso è formato da una coppia di numeri reali detti parte *reale* e parte *immaginaria*.  
Un numero complesso può essere rappresentato in *forma algebrica* come:

$$a + ib$$

dove  $a$  è la parte *reale*  
 $b$  è la parte *immaginaria*

Un numero complesso ammette anche una *rappresentazione polare*:

$$(r, q)$$

dove  $r$  è un numero reale detto *modulo*  
 $q$  è un arco compreso tra 0 e  $2\pi$  detto *argomento* del numero complesso

Le relazioni tra le due rappresentazioni  $a+ib$  e  $(r,q)$  sono:

$$\begin{aligned} r &= \sqrt{a^2 + b^2} ; \quad q = \operatorname{atan}(b/a) \\ a &= r \cos(q) ; \quad b = r \sin(q) \end{aligned}$$

Sui numeri complessi sono definiti **somma** e **prodotto** come segue:

$$\begin{aligned} (a_1 + ib_1) + (a_2 + ib_2) &= (a_1 + a_2) + i(b_1 + b_2) \quad (\text{notazione algebrica}) \\ (r_1, q_1) * (r_2, q_2) &= (r_1 r_2, q_1 + q_2) \quad (\text{notazione polare}) \end{aligned}$$

La progettazione di una classe richiede:

- ☛ *individuare l'insieme dei servizi pubblici messi a disposizione dalla classe (parte pubblica)*
- ☛ *definire la struttura dati e le funzionalità non visibili della classe (parte privata)*
- ☛ *implementare tutte le funzioni dichiarate nella classe*

## Parte pubblica

### Specifica

1. creare un numero complesso a partire da parte intera e parte immaginaria
2. creare un numero complesso a partire da modulo e argomento
3. stampare un numero complesso secondo la rappresentazione algebrica
4. stampare un numero complesso secondo la rappresentazione polare
5. moltiplicare numeri complessi con notazione infissa
6. sommare numeri complessi con notazione infissa
7. effettuare l'assegnamento tra numeri complessi
8. verificare l'uguaglianza tra numeri complessi

## Parte pubblica

```
classe Complex
{
    ...

public:
    Complex(float a, float b);
    Complex(float a, int b);
    void algPrint();
    void polPrint();
    Complex operator*(const Complex altroNum);
    Complex operator+(const Complex altroNum);
    Complex operator=(const Complex altroNum);
    Boolean operator==(const Complex altroNum);

};
```


## Parte pubblica

```
classe Complex
{
    ...

public:
    Complex(float a, float b);
    Complex(float a, int b);
    void algPrint();
    void polPrint();
    Complex operator*(const Complex altroNum);
    Complex operator+(const Complex altroNum);
    Complex operator=(const Complex altroNum);
    Boolean operator==(const Complex altroNum);

};
```

Costruttore con parte  
*reale* ed *immaginaria*






## Parte pubblica

```
classe Complex
{
    ...

public:
    Complex(float a, float b);
    Complex(float a, int b);
    void algPrint();
    void polPrint();
    Complex operator*(const Complex altroNum);
    Complex operator+(const Complex altroNum);
    Complex operator=(const Complex altroNum);
    Boolean operator==(const Complex altroNum);

};
```

Costruttore con parte  
*reale* ed *immaginaria*



## Parte pubblica


```
classe Complex
{
    ...
```

```
public:
```

```
    Complex(float a, float b);
    Complex(float a, int b);
    void algPrint();
    void polPrint();
    Complex operator*(const Complex altroNum);
    Complex operator+(const Complex altroNum);
    Complex operator=(const Complex altroNum);
    Boolean operator==(const Complex altroNum);
```

```
};
```

Costruttore con parte  
*reale* ed *immaginaria*



## Parte pubblica

```
classe Complex
{
    ...

public:
    Complex(float a, float b);
    Complex(float a, int b);
    void algPrint();
    void polPrint();
    Complex operator*(const Complex altroNum);
    Complex operator+(const Complex altroNum);
    Complex operator=(const Complex altroNum);
    Boolean operator==(const Complex altroNum);

};
```

Costruttore con parte  
*reale* ed *immaginaria*

Costruttore con  
*modulo* e *argomento*

## Parte pubblica

```
classe Complex
{
    ...

public:
    Complex(float a, float b);
    Complex(float a, int b);
    void algPrint();
    void polPrint();
    Complex operator*(const Complex altroNum);
    Complex operator+(const Complex altroNum);
    Complex operator=(const Complex altroNum);
    Boolean operator==(const Complex altroNum);

};
```

Costruttore con parte  
*reale* ed *immaginaria*

Costruttore con  
*modulo* e *argomento*

## Parte pubblica

```
classe Complex
{
    ...

public:
    Complex(float a, float b);
    Complex(float a, int b);
    void algPrint();
    void polPrint();
    Complex operator*(const Complex altroNum);
    Complex operator+(const Complex altroNum);
    Complex operator=(const Complex altroNum);
    Boolean operator==(const Complex altroNum);

};
```

Costruttore con parte  
*reale* ed *immaginaria*

Costruttore con  
*modulo* e *argomento*

## Parte pubblica

```
classe Complex
{
    ...

public:
    Complex(float a, float b);
    Complex(float a, int b);
    void algPrint();
    void polPrint();
    Complex operator*(const Complex altroNum);
    Complex operator+(const Complex altroNum);
    Complex operator=(const Complex altroNum);
    Boolean operator==(const Complex altroNum);

};
```

Costruttore con parte  
*reale* ed *immaginaria*

Costruttore con  
*modulo* e *argomento*

Stampe

## Parte pubblica

```
classe Complex
```

```
{
```

```
...
```

```
public:
```

```
Complex(float a, float b);
```

```
Complex(float a, int b);
```

```
void algPrint();
```

```
void polPrint();
```

```
Complex operator*(const Complex altroNum);
```

```
Complex operator+(const Complex altroNum);
```

```
Complex operator=(const Complex altroNum);
```

```
Boolean operator==(const Complex altroNum);
```

```
};
```

Costruttore con parte  
*reale* ed *immaginaria*

Costruttore con  
*modulo* e *argomento*

Stampe

Overloading  
degli operatori  
\* e +

## Parte pubblica

```
classe Complex
```

```
{
```

```
...
```

```
public:
```

```
    Complex(float a, float b);
```

```
    Complex(float a, int b);
```

```
    void algPrint();
```

```
    void polPrint();
```

```
    Complex operator*(const Complex altroNum);
```

```
    Complex operator+(const Complex altroNum);
```

```
    Complex operator=(const Complex altroNum);
```

```
    Boolean operator==(const Complex altroNum);
```

```
};
```

Costruttore con parte  
*reale* ed *immaginaria*

Costruttore con  
*modulo* e *argomento*

Stampe

Overloading  
degli operatori  
*\** e *+*



## Parte pubblica

```
classe Complex
```

```
{
```

```
...
```

```
public:
```

```
Complex(float a, float b);
```

```
Complex(float a, int b);
```

```
void algPrint();
```

```
void polPrint();
```

```
Complex operator*(const Complex altroNum);
```

```
Complex operator+(const Complex altroNum);
```

```
Complex operator=(const Complex altroNum);
```

```
Boolean operator==(const Complex altroNum);
```

```
};
```

Costruttore con parte  
*reale* ed *immaginaria*

Costruttore con  
*modulo* e *argomento*

Stampe

Overloading  
degli operatori  
\* e +

Overloading  
degli operatori  
= e ==

## Parte pubblica

```
classe Complex
```

```
{
```

```
...
```

```
public:
```

```
    Complex(float a, float b);
```

```
    Complex(float a, int b);
```

```
    void algPrint();
```

```
    void polPrint();
```

```
    Complex operator*(const Complex altroNum);
```

```
    Complex operator+(const Complex altroNum);
```

```
    Complex operator=(const Complex altroNum);
```

```
    Boolean operator==(const Complex altroNum);
```

```
};
```

Costruttore con parte  
*reale* ed *immaginaria*

Costruttore con  
*modulo* e *argomento*

Stampe

Overloading  
degli operatori  
*\** e *+*

Overloading  
degli operatori  
*=* e *==*

## Parte pubblica

```
classe Complex
```

```
{
```

```
...
```

```
public:
```

```
    Complex(float a, float b);
```

```
    Complex(float a, int b);
```

```
    void algPrint();
```

```
    void polPrint();
```

```
    Complex operator*(const Complex altroNum);
```

```
    Complex operator+(const Complex altroNum);
```

```
    Complex operator=(const Complex altroNum);
```

```
    Boolean operator==(const Complex altroNum);
```

```
};
```

Costruttore con parte  
*reale* ed *immaginaria*

Costruttore con  
*modulo* e *argomento*

Stampe

Overloading  
degli operatori  
\* e +

Overloading  
degli operatori  
= e ==

## Parte pubblica

```
classe Complex
```

```
{
```

```
...
```

```
public:
```

```
    Complex(float a, float b);
```

```
    Complex(float a, int b);
```

```
    void algPrint();
```

```
    void polPrint();
```

```
    Complex operator*(const Complex altroNum);
```

```
    Complex operator+(const Complex altroNum);
```

```
    Complex operator=(const Complex altroNum);
```

```
    Boolean operator==(const Complex altroNum);
```

```
};
```

Costruttore con parte  
*reale* ed *immaginaria*

Costruttore con  
*modulo* e *argomento*

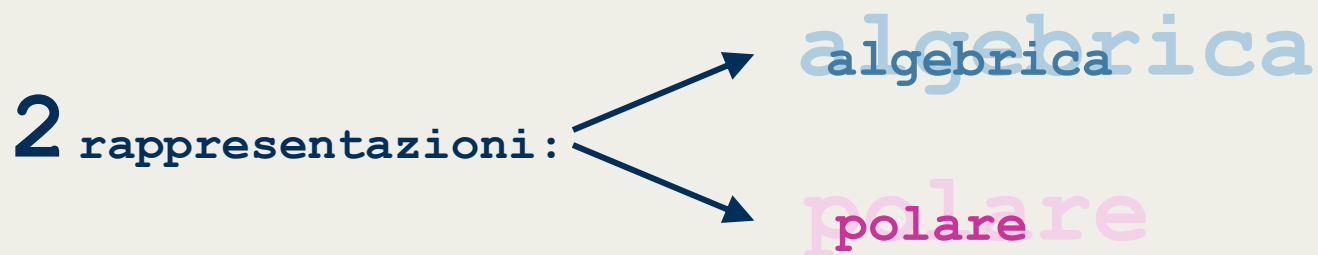
Stampe

Overloading  
degli operatori  
*\** e *+*

Overloading  
degli operatori  
*=* e *==*

## Parte *privata*

☛ Progettazione della struttura dati:



*Quale rappresentazione?*

## Parte *privata*

```
classe Complex
{
    ...

public:
    Complex(float a, float b);
    Complex(float a, int b);
    void algPrint();
    void polPrint();
    Complex operator*(const Complex altroNum);
    Complex operator+(const Complex altroNum);
    Complex operator=(const Complex altroNum);
    Boolean operator==(const Complex altroNum);

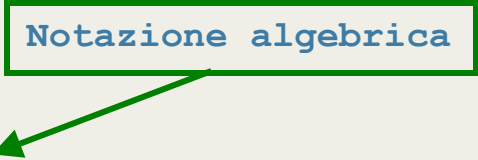
};
```

## Parte *privata*

```
classe Complex
{
    ...

public:
    Complex(float a, float b);
    Complex(float a, int b);
    void algPrint();
    void polPrint();
    Complex operator*(const Complex altroNum);
    Complex operator+(const Complex altroNum);
    Complex operator=(const Complex altroNum);
    Boolean operator==(const Complex altroNum);

};
```



A green rectangular box with the text "Notazione algebrica" is positioned above the first constructor. A green arrow points from the bottom-left corner of this box to the parameter "float b" in the constructor signature "Complex(float a, float b);".

## Parte *privata*

```
classe Complex
{
    ...

public:
    Complex(float a, float b);
    Complex(float a, int b);
    void algPrint();
    void polPrint();
    Complex operator*(const Complex altroNum);
    Complex operator+(const Complex altroNum);
    Complex operator=(const Complex altroNum);
    Boolean operator==(const Complex altroNum);

};
```

Notazione algebrica

Notazione polare



## Parte *privata*

```
classe Complex
```

```
{
```

```
    ...
```

```
public:
```

```
    Complex(float a, float b);
```

```
    Complex(float a, int b);
```

```
    void algPrint();
```

```
    void polPrint();
```

```
    Complex operator*(const Complex altroNum);
```

```
    Complex operator+(const Complex altroNum);
```

```
    Complex operator=(const Complex altroNum);
```

```
    Boolean operator==(const Complex altroNum);
```

```
};
```

Notazione algebrica

Notazione polare

Notazione algebrica

## Parte *privata*

```
classe Complex
```

```
{
```

```
...
```

```
public:
```

```
Complex(float a, float b);
```

```
Complex(float a, int b);
```

```
void algPrint();
```

```
void polPrint();
```

```
Complex operator*(const Complex altroNum);
```

```
Complex operator+(const Complex altroNum);
```

```
Complex operator=(const Complex altroNum);
```

```
Boolean operator==(const Complex altroNum);
```

```
};
```

Notazione algebrica

Notazione polare

Notazione algebrica

Notazione polare

## Parte *privata*

```
classe Complex
```

```
{
```

```
    ...
```

```
public:
```

```
    Complex(float a, float b);
```

```
    Complex(float a, int b);
```

```
    void algPrint();
```

```
    void polPrint();
```

```
    Complex operator*(const Complex altroNum);
```

```
    Complex operator+(const Complex altroNum);
```

```
    Complex operator=(const Complex altroNum);
```

```
    Boolean operator==(const Complex altroNum);
```

```
};
```

Notazione algebrica

Notazione polare

Notazione algebrica

Notazione polare

Notazione algebrica

## Parte *privata*

```
classe Complex
```

```
{
```

```
    ...
```

```
public:
```

```
    Complex(float a, float b);
```

```
    Complex(float a, int b);
```

```
    void algPrint();
```

```
    void polPrint();
```

```
    Complex operator*(const Complex altroNum);
```

```
    Complex operator+(const Complex altroNum);
```

```
    Complex operator=(const Complex altroNum);
```

```
    Boolean operator==(const Complex altroNum);
```

```
};
```

Notazione algebrica

Notazione polare

Notazione algebrica

Notazione polare

Notazione algebrica

Notazione polare

## Parte *privata*

```
classe Complex
```

```
{
```

```
    ...
```

```
public:
```

```
    Complex(float a, float b);
```

```
    Complex(float a, int b);
```

```
    void algPrint();
```

```
    void polPrint();
```

```
    Complex operator*(const Complex altroNum);
```

```
    Complex operator+(const Complex altroNum);
```

```
    Complex operator=(const Complex altroNum);
```

```
    Boolean operator==(const Complex altroNum);
```

```
};
```

Notazione algebrica

Notazione polare

Notazione algebrica

Notazione polare

Notazione algebrica

Notazione polare

## Parte *privata*

```
classe Complex
{
    float reale,compl;
    float arg; float modulo;

    float calcArg      (float a,float b);
    float calcModulo   (float a,float b);
    float calcReale    (float a,float b);
    float calcComp1    (float a,float b);

public:
    Complex(float a,float b);
    ...
};
```

## Parte *privata*

```
classe Complex
```

```
{
```

```
    float reale, compl;
```

```
    float arg; float modulo;
```

```
    float calcArg      (float a, float b);
```

```
    float calcModulo   (float a, float b);
```

```
    float calcReale    (float a, float b);
```

```
    float calcComp1    (float a, float b);
```

```
public:
```

```
    Complex(float a, float b);
```

```
    ...
```

```
};
```



Notazione algebrica

## Parte *privata*

```
classe Complex
```

```
{
```

```
    float reale,compl;
```

```
    float arg; float modulo;
```

Notazione algebrica

Notazione polare

```
    float calcArg      (float a,float b);
```

```
    float calcModulo   (float a,float b);
```

```
    float calcReale    (float a,float b);
```

```
    float calcComp1    (float a,float b);
```

```
public:
```

```
    Complex(float a,float b);
```

```
    ...
```

```
};
```



## Parte *privata*

```
classe Complex
```

```
{
```

```
    float reale,compl;
```

```
    float arg; float modulo;
```

```
    float calcArg      (float a,float b);
```

```
    float calcModulo   (float a,float b);
```

```
    float calcReale    (float a,float b);
```

```
    float calcComp1    (float a,float b);
```

```
public:
```

```
    Complex(float a,float b);
```

```
    ...
```

```
};
```

Notazione algebrica

Notazione polare

Funzioni di supporto  
per le conversioni

## Parte *privata*

```
classe Complex
{
public:
    Complex(float a,float b);
    ...

private:
    float reale,compl;
    float arg; float modulo;

    float calcArg      (float a,float b);
    float calcModulo   (float a,float b);
    float calcReale    (float a,float b);
    float calcComp1    (float a,float b);

};
```