

ESTRUCTURAS DE DATOS

NOTAS SOBRE C++

Plantillas en clases

Manuel Montenegro Montes

Departamento de Sistemas Informáticos y Computación
Facultad de Informática – Universidad Complutense de Madrid



Repaso: números complejos

```
class Complejo {  
public:  
    Complejo(double real, double imag);  
  
    double get_real() const;  
    double get_imag() const;  
    void display() const;  
  
    Complejo operator+(const Complejo &z) const;  
    Complejo operator*(const Complejo &z) const;  
  
private:  
    double real, imag;  
};
```

- ¿Y si quisiera también una clase Complejo en la que las partes reales o imaginarias sean float, en lugar de double?
- Para evitar duplicidad de código puedo utilizar plantillas.

Generalización de una clase

```
template<typename T>
class Complejo {
public:
    Complejo(T real, T imag);

    T get_real() const;
    T get_imag() const;

    void display(std::ostream &out) const;

    Complejo operator+(const Complejo &z) const;
    Complejo operator*(const Complejo &z) const;

private:
    T real, imag;
};
```



Generalización de los métodos

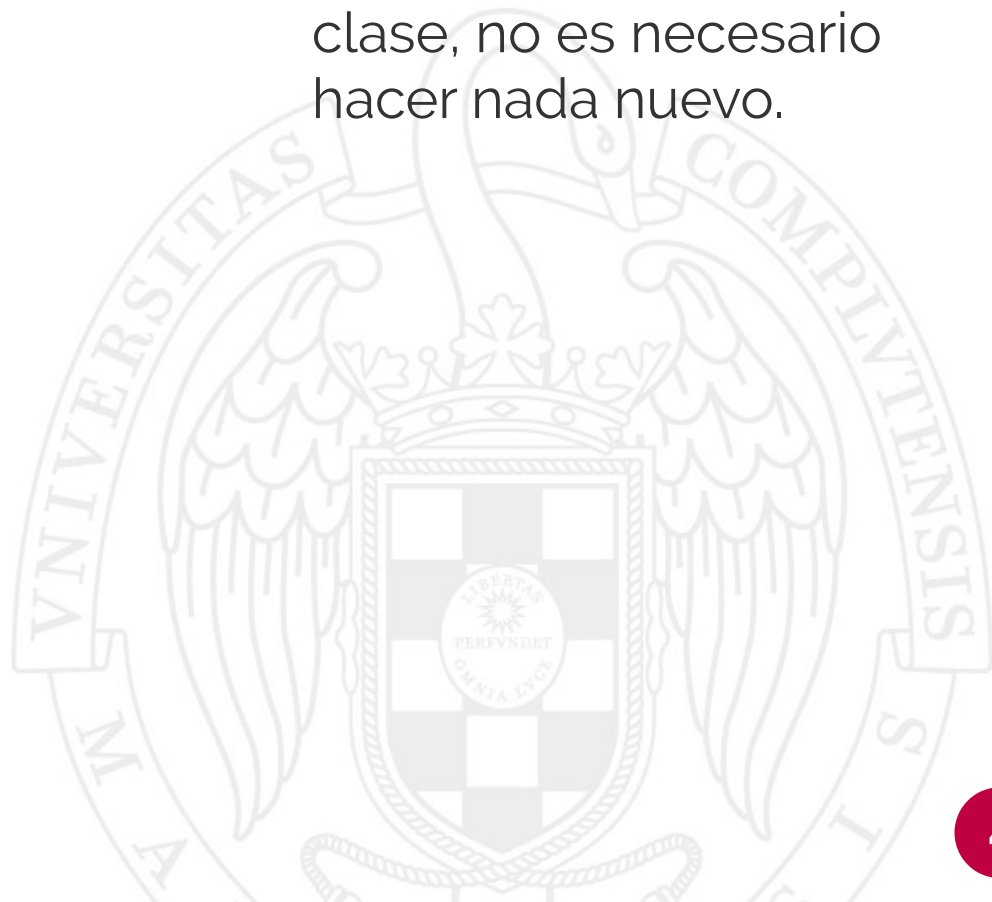
```
template<typename T>
class Complejo {
public:
    ...

    T get_real() const {
        return real;
    };

    T get_imag() const {
        return imag;
    };

    ...
private:
    T real, imag;
};
```

- Si el método se implementa dentro de la clase, no es necesario hacer nada nuevo.



Generalización de los métodos

```
template<typename T>
class Complejo {
public:
    ...

    Complejo operator+(const Complejo &z1) const;
    Complejo operator*(const Complejo &z1) const;
    ...
private:
    T real, imag;
};
```

```
template<typename T>
Complejo<T> Complejo<T>::operator+(const Complejo<T> &z) const {
    return { real + z.real, imag + z.imag };
}
```

- Si el método se implementa fuera de la clase, es necesario indicar que el método también es una plantilla.

Uso de una clase genérica

- A la hora de crear una instancia de una clase genérica, hay que indicar el tipo con el que se instancia:

```
Complejo<double> z1(2.0, -3.0), z2(1.0, 0.0);
```

```
Complejo<float> z4(2.0, -3.0);
```



*En las clases, es **obligatorio** indicar el tipo con el que instanciar la plantilla*

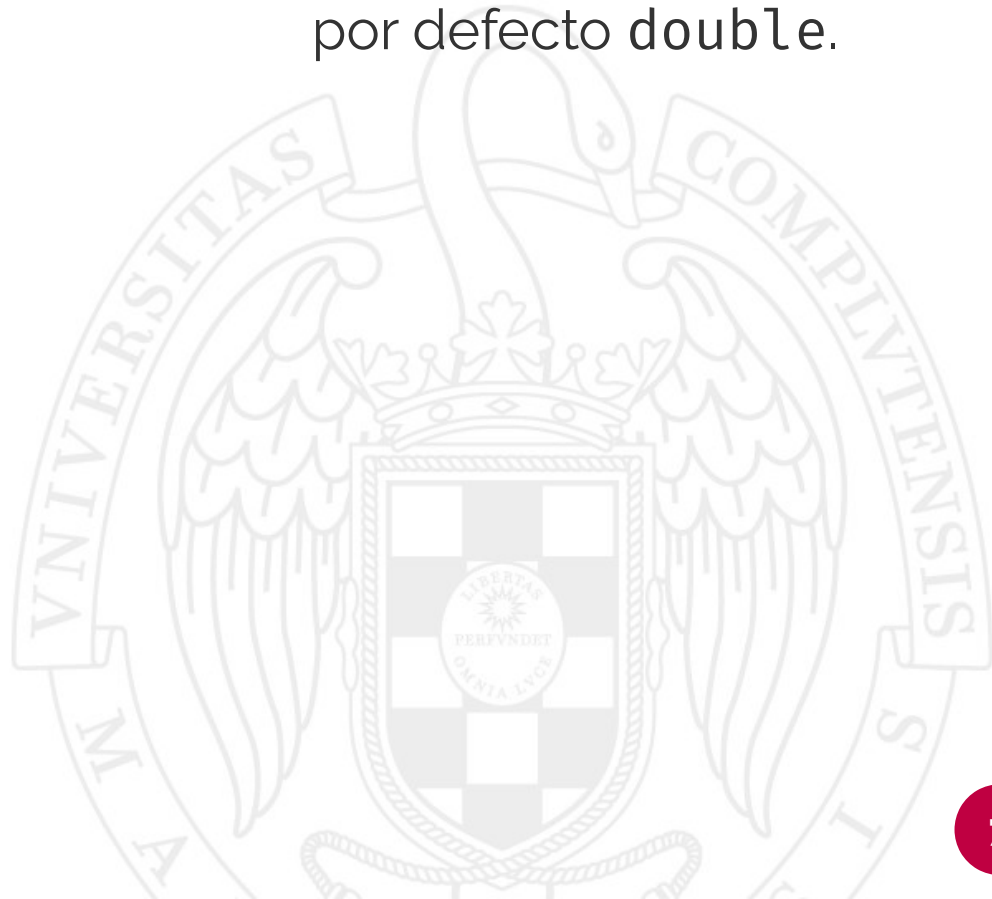
```
Complejo z4(2.0, -3.0); ❌
```

- ... aunque es posible indicar un tipo por defecto para la instancia.

Plantillas: argumentos por defecto

```
template<typename T = double>  
class Complejo {  
    ...  
};
```

- Si no se indica el tipo en la instanciación, se utilizará por defecto `double`.



Plantillas: argumentos por defecto

- Aún si queremos utilizar argumentos por defecto, es necesario indicar los delimitadores < y >, aunque no tengan nada en su interior.

Complejo<> z1(2.0, -3.0), z2(1.0, 0.0); ← **Correcto (= Complejo<double>)**

Complejo z1(2.0, -3.0), z2(1.0, 0.0); ← **Incorrecto**