

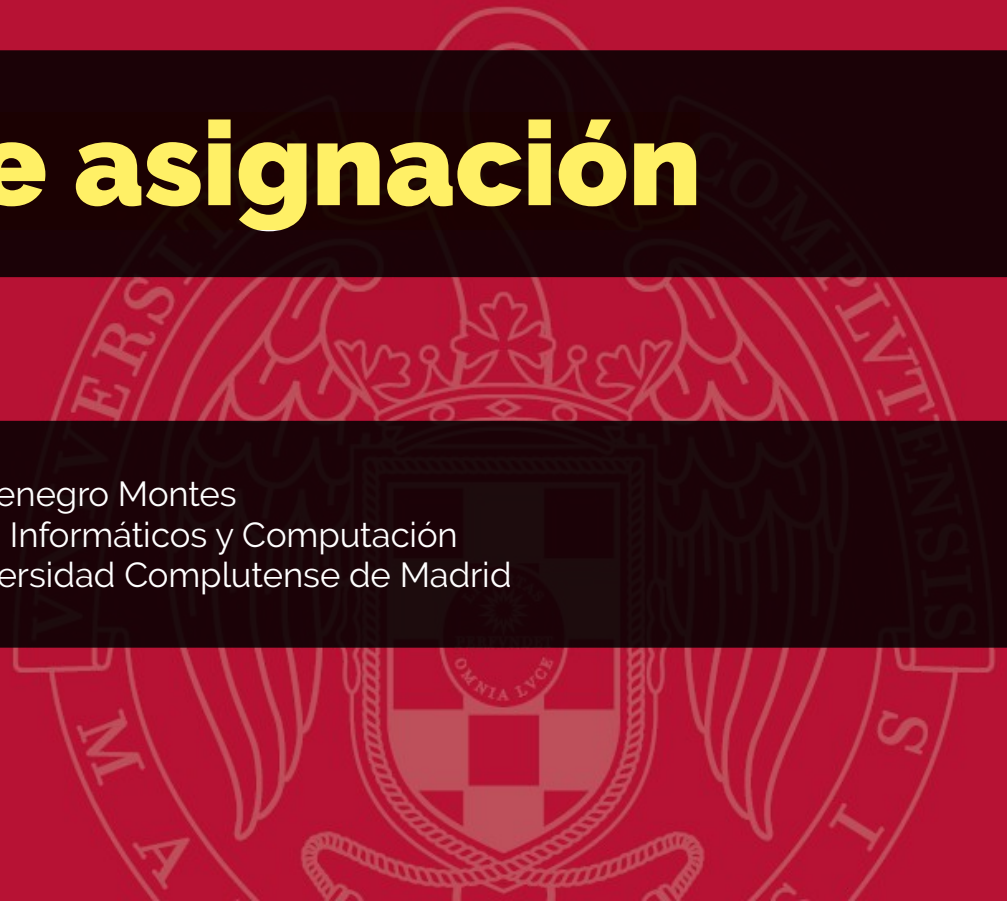
ESTRUCTURAS DE DATOS

NOTAS SOBRE C++

Operador de asignación

Manuel Montenegro Montes

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



Recordatorio: clase Fecha

Queremos sobrecargar el operador "=" que usamos para asignar valores a variables y demás.

```
class Fecha { Ya la hemos visto
public:
    Fecha(int dia, int mes, int anyo);
    Fecha(int anyo);
    Fecha();

    int get_dia() const;
    void set_dia(int dia);
    int get_mes() const;
    void set_mes(int mes);
    int get_anyo() const;
    void set_anyo(int anyo);
    void imprimir();

private:
    int dia;
    int mes;
    int anyo;
};
```



Asignar un objeto Fecha a otro

```
Fecha f1(10, 1, 2010);
```

```
Fecha f2(13, 2, 1993);
```

```
f2 = f1;
```

Copia campo a campo, atributo a atributo
los valores de f1 en f2.

Pila

f1:

dia	10
mes	1
año	2010

f2:

dia	13
mes	2
año	1993

Heap

Asignar un objeto Fecha a otro

```
Fecha f1(10, 1, 2010);
```

```
Fecha f2(13, 2, 1993);
```

```
f2 = f1;
```

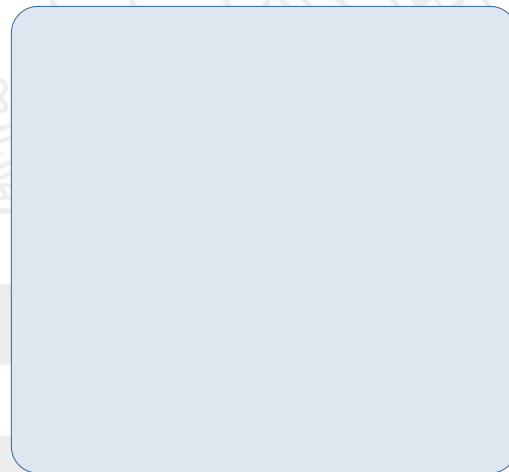
esto es razonable para las fechas.

Pila

f1:	dia	10
	mes	1
	año	2010

f2:	dia	10
	mes	1
	año	2010

Heap



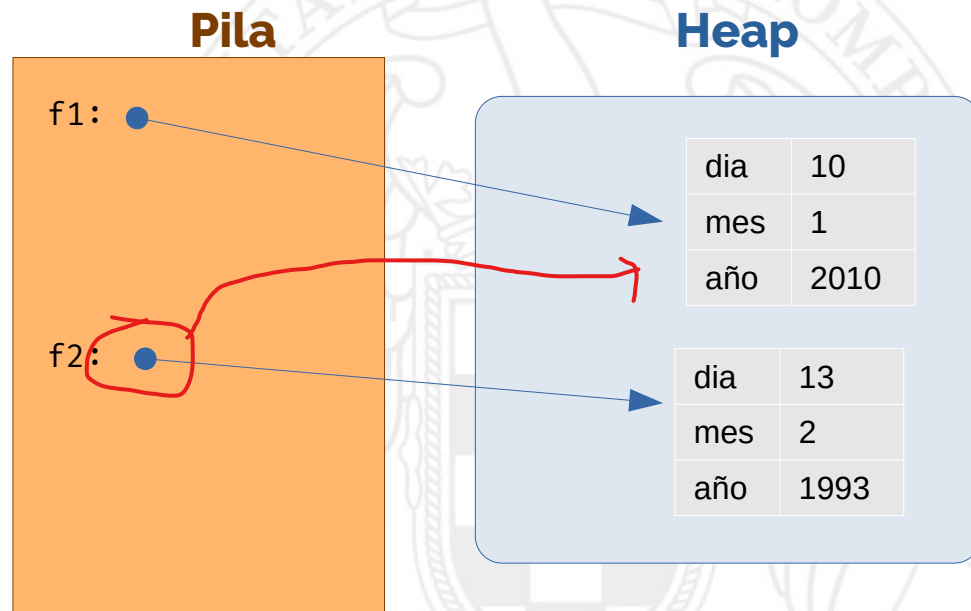
Asignar un objeto Fecha a otro

```
Fecha *f1 = new Fecha(10, 1, 2010);  
Fecha *f2 = new Fecha(13, 2, 1993);
```

Punteros a fechas, como ocurriría en Java

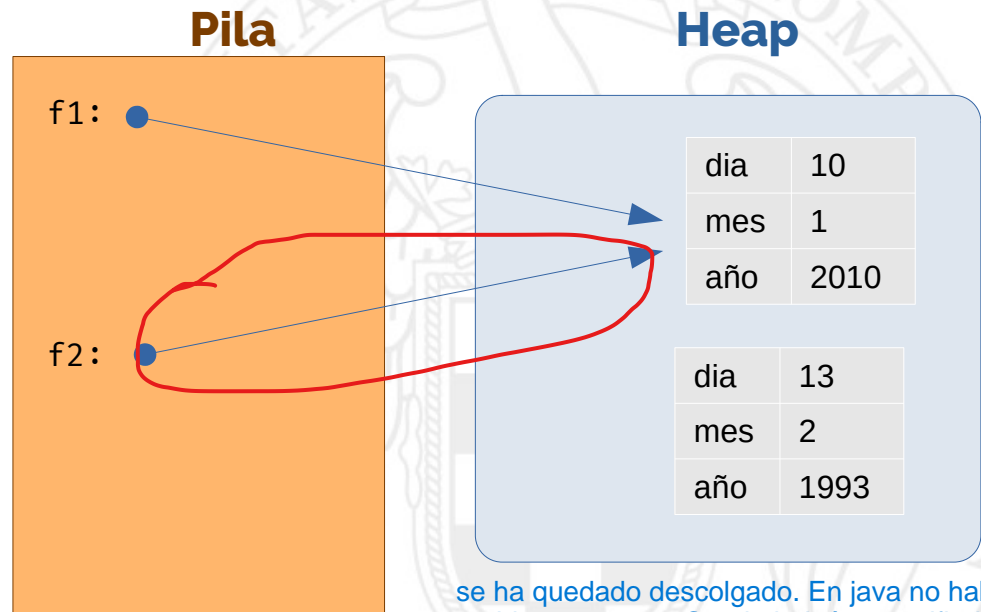
```
f2 = f1;
```

Cambiaría a dónde apunta f2, es decir, f2 apuntaría a donde apunta f1



Asignar un objeto Fecha a otro

```
Fecha *f1 = new Fecha(10, 1, 2010);  
Fecha *f2 = new Fecha(13, 2, 1993);  
delete f2; Esto es lo que teníamos que haber hecho  
f2 = f1;
```



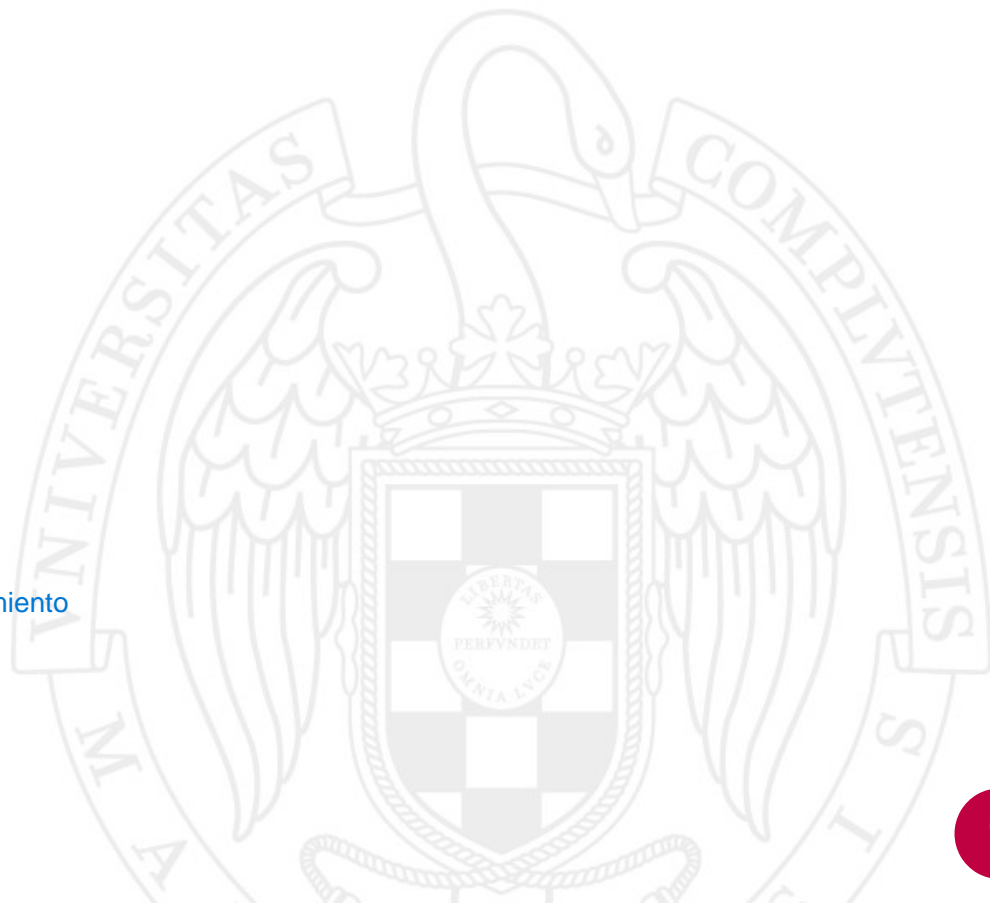
Recordatorio: clase Persona

```
class Persona { También laa vimos
public:
    Persona(std::string nombre,
            int dia,
            int mes,
            int anyo);

    ~Persona() {
        delete fecha_nacimiento;
    }

    ...

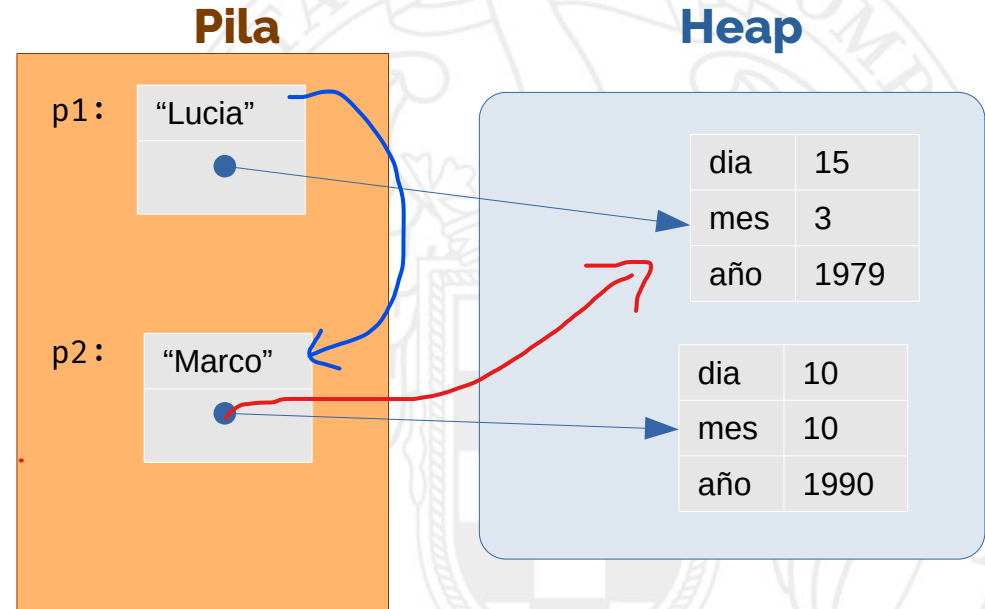
private:
    std::string nombre;
    Fecha *fecha_nacimiento; Puntero a una fecha de nacimiento
};
```



Asignar un objeto Persona a otro

```
Persona p1("Lucía", 15, 3, 1979);  
Persona p2("Marco", 10, 10, 1990);
```

```
p2 = p1;
```



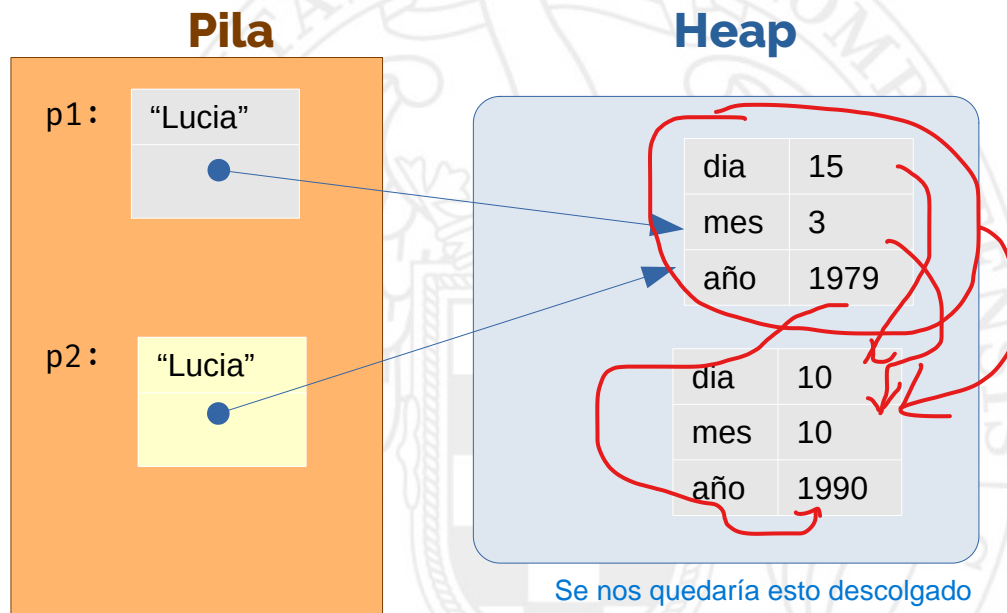
Asignar un objeto Persona a otro

```
Persona p1("Lucía", 15, 3, 1979);  
Persona p2("Marco", 10, 10, 1990);
```

```
p2 = p1;
```

Nosotros queríamos traspasar la información de la fecha de p1 a p2 pero lo que está haciendo es cambiar a donde apunta la variable p2.

Si nosotros queremos hacer lo de arriba tendremos que sobrecargar el operador de asignación.



Sobrecargando el operador de asignación

```
class Persona {  
public:
```

DENTRO DE LA CLASE PERSONA.

...

```
void operator=(const Persona &other) {  
    nombre = other.nombre;  
    fecha_nacimiento->set_dia(other.fecha_nacimiento->get_dia());  
    fecha_nacimiento->set_mes(other.fecha_nacimiento->get_mes());  
    fecha_nacimiento->set_anyo(other.fecha_nacimiento->get_anyo());  
}
```

...

```
private:  
    std::string nombre;  
    Fecha *fecha_nacimiento;  
};
```

$p2 = p1$

equivale a

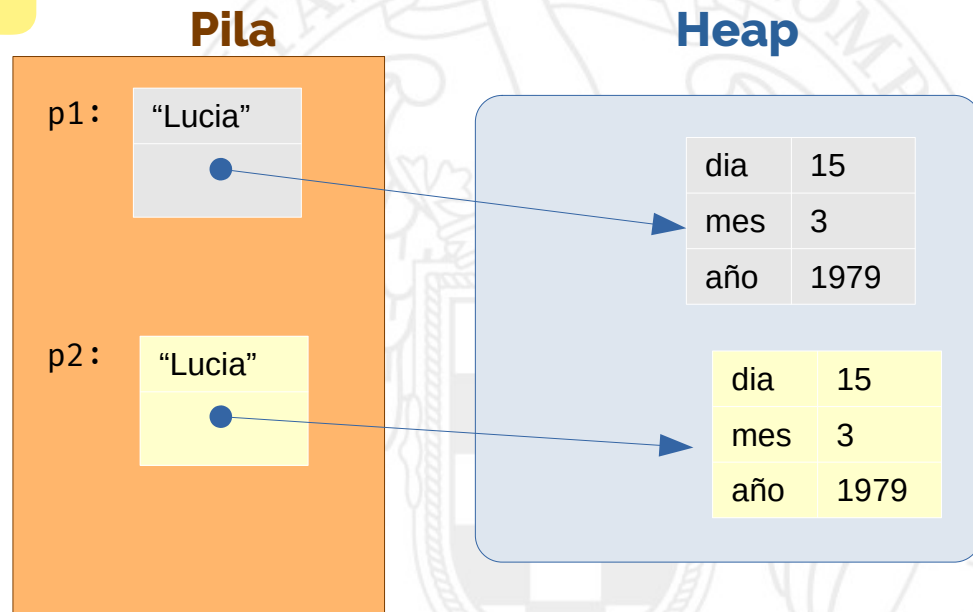
$p2.operator=(p1)$

Asignar un objeto Persona a otro

```
Persona p1("Lucía", 15, 3, 1979);  
Persona p2("Marco", 10, 10, 1990);
```

```
p2 = p1;
```

Con las modificaciones anteriores (sobrecarga del operador "=") conseguimos que siga apuntando al mismo objeto pero solo cambiamos su fecha de nacimiento.



Otra posibilidad

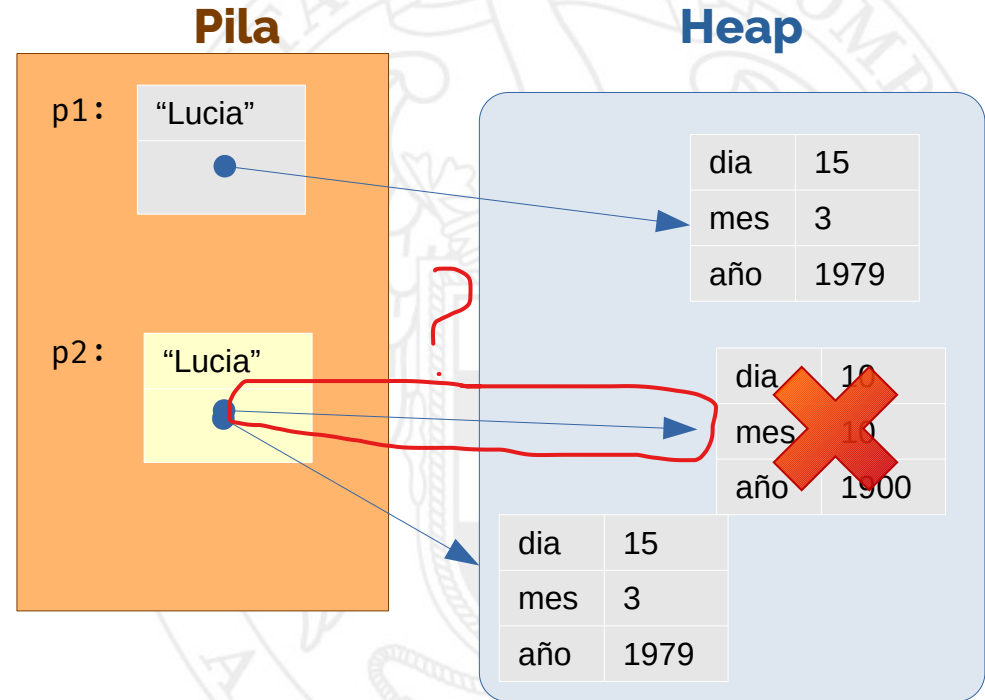
```
class Persona {  
public:  
  
...  
  
void operator=(const Persona &other) { Otra posibilidad de implementación.  
    nombre = other.nombre;  
    delete fecha_nacimiento;  
    fecha_nacimiento = new Fecha(*other.fecha_nacimiento);  
}  
...  
  
private:  
    std::string nombre;  
    Fecha *fecha_nacimiento;  
};
```



Asignar un objeto Persona a otro

```
Persona p1("Lucía", 15, 3, 1979);  
Persona p2("Marco", 10, 10, 1990);
```

```
p2 = p1;
```



El problema de la autoasignación



Asignar un objeto persona a sí mismo

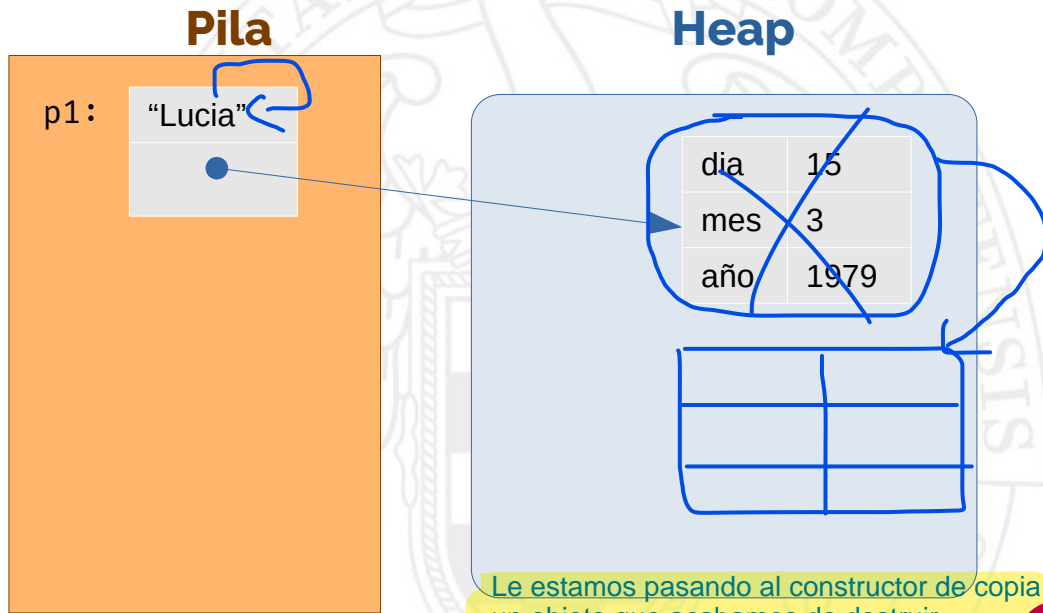
```
Persona p1("Lucía", 15, 3, 1979);
```

```
p1 = p1;
```

Para las asignaciones tenemos que tener en cuenta el caso particular de las autoasignaciones.

```
void operator=(const Persona &other) {  
    nombre = other.nombre;  
    delete fecha_nacimiento;  
    fecha_nacimiento = new Fecha(*other.fecha_nacimiento);  
}
```

Implementación del operador igual



Le estamos pasando al constructor de copia un objeto que acabamos de destruir.

Evitando la autoasignación

```
class Persona {  
public:
```

```
...
```

```
void operator=(const Persona &other) {  
    if (this  $\neq$  &other) { si se hace una autoasignación, por tanto, NO HACEMOS NADA.  
        nombre = other.nombre;  
        delete fecha_nacimiento;  
        fecha_nacimiento = new Fecha(*other.fecha_nacimiento);  
    }  
}
```

```
...
```

```
private:  
    std::string nombre;  
    Fecha *fecha_nacimiento;  
};
```

$p1 = p1$

NO HARÍA NADA.

Encadenar asignaciones



Encadenar asignaciones

```
int x, y, z;  
x = y = z = 0;
```

Tres variables, encadenamos asignaciones, todas al valor 0. El operador = asocia a la izquierda, lo que significa que le asignamos a la z 0, a la y 0 y a la x 0.

$$x = \left(y = \left(z = 0 \right) \right);$$

$$x = 0$$

¿Podemos hacer lo mismo con objetos Persona?

```
Persona p1("Lucía", 15, 3, 1979);  
Persona p2("Marco", 10, 10, 1990);  
Persona p3("Laura", 1, 3, 1980);
```

p3 = p2 = p1;  Esto no sería posible.

p3 = (p2 = p1);
leaving

Devolviendo referencia a this

```
class Persona {  
public:
```

```
...  
Persona & operator=(const Persona &other) {  
    if (this != &other) {  
        nombre = other.nombre;  
        delete fecha_nacimiento;  
        fecha_nacimiento = new Fecha(*other.fecha_nacimiento);  
    }  
    return *this;  
}  
...
```


Debe devolver una referencia a una persona.

devuelve el valor de this.

```
private:  
    std::string nombre;  
    Fecha *fecha_nacimiento;  
};
```

¿Podemos hacer lo mismo con objetos Persona?

```
Persona p1("Lucía", 15, 3, 1979);  
Persona p2("Marco", 10, 10, 1990);  
Persona p3("Laura", 1, 3, 1980);
```

p3 = p2 = p1; 

p3 = (p2 = p1);

p3 = p2

A su vez p2 = p1.

Constructor de copia

- Para crear un objeto nuevo con la misma información que otro existente.

```
Persona p1(...);
```

```
Persona p2 = p1;
```

Se llama al constructor de copia.

- No devuelve nada.
- No puede producirse autoasignación:

```
Persona p2 = p2;
```

vs.

Operador asignación

- Para copiar la información de un objeto existente a otro existente.

```
Persona p1(...);
```

```
Persona p2(...);
```

Sobre un objeto p2 que ya había sido creado

```
p2 = p1;
```

- Devuelve `*this`.
- Hay que tener en cuenta la autoasignación:

```
p2 = p2;
```