

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

```
class Fecha {  
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;
```

Pila

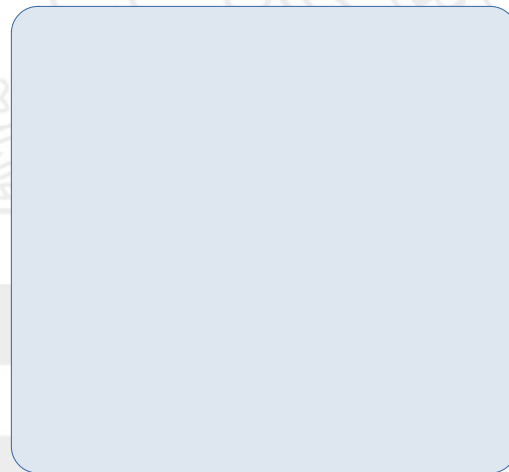
f1:

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;
```

Pila

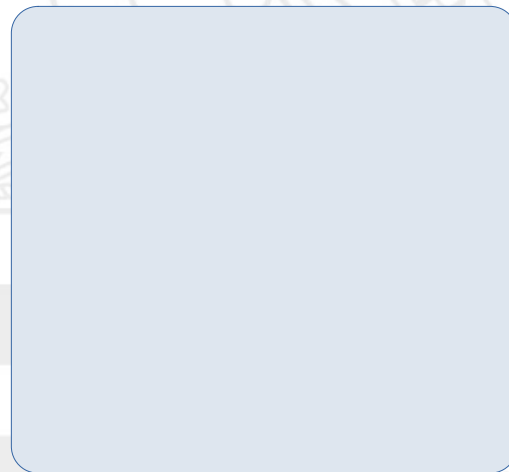
f1:

dia	10
mes	1
año	2010

f2:

dia	10
mes	1
año	2010

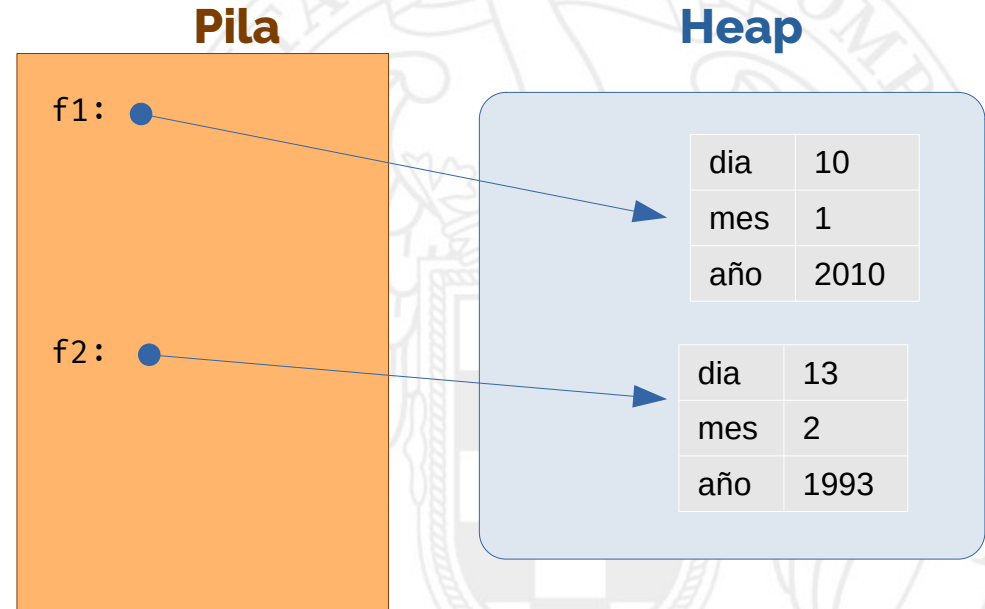
Heap



Asignar un objeto Fecha a otro

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

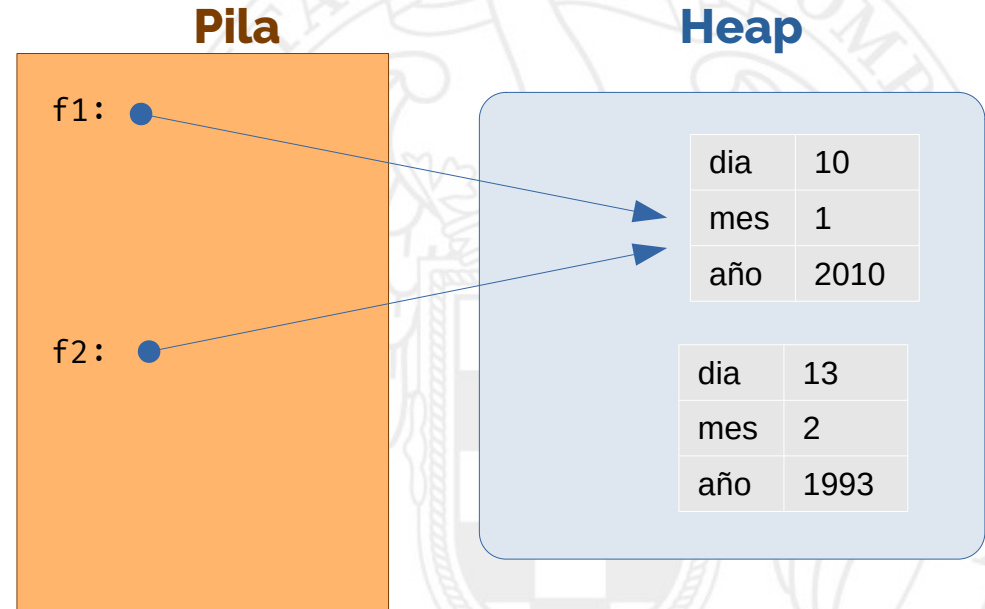
```
f2 = f1;
```



Asignar un objeto Fecha a otro

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

```
f2 = f1;
```



Recordatorio: clase Persona

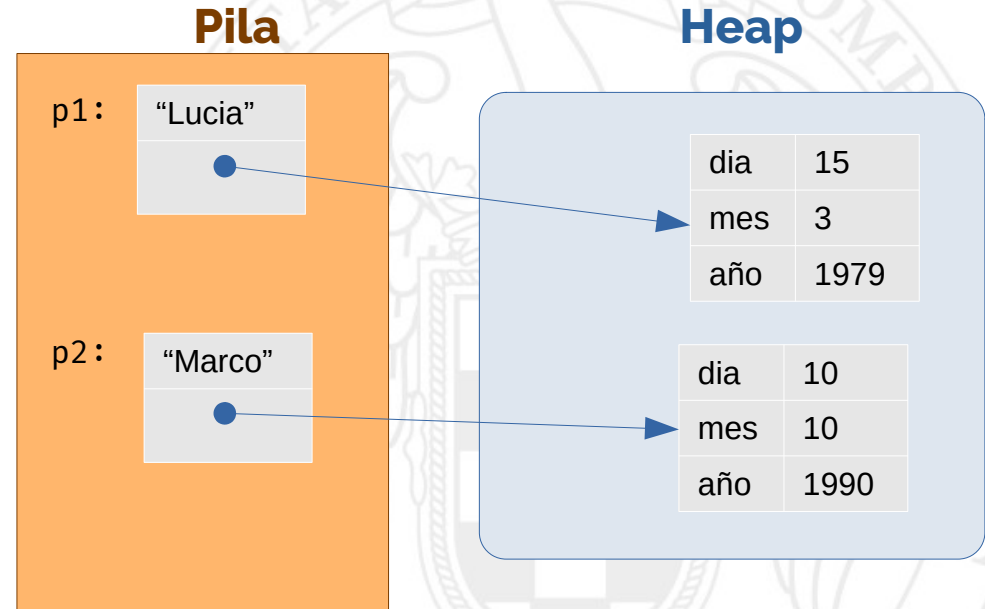
```
class Persona {  
public:  
    Persona(std::string nombre,  
            int dia,  
            int mes,  
            int anyo);  
  
    ~Persona() {  
        delete fecha_nacimiento;  
    }  
  
    ...  
  
private:  
    std::string nombre;  
    Fecha *fecha_nacimiento;  
};
```



Asignar un objeto Persona a otro

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

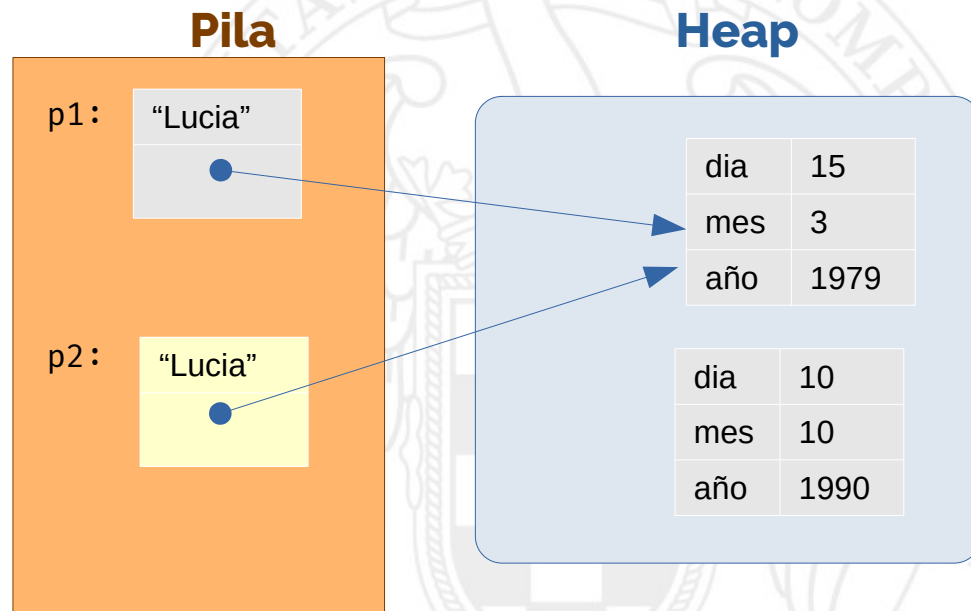
```
p2 = p1;
```



Asignar un objeto Persona a otro

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

```
p2 = p1;
```



Sobrecargando el operador de asignación

```
class Persona {  
public:
```

`p2 = p1`

equivale a

`p2.operator=(p1)`

...

```
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());  
}
```

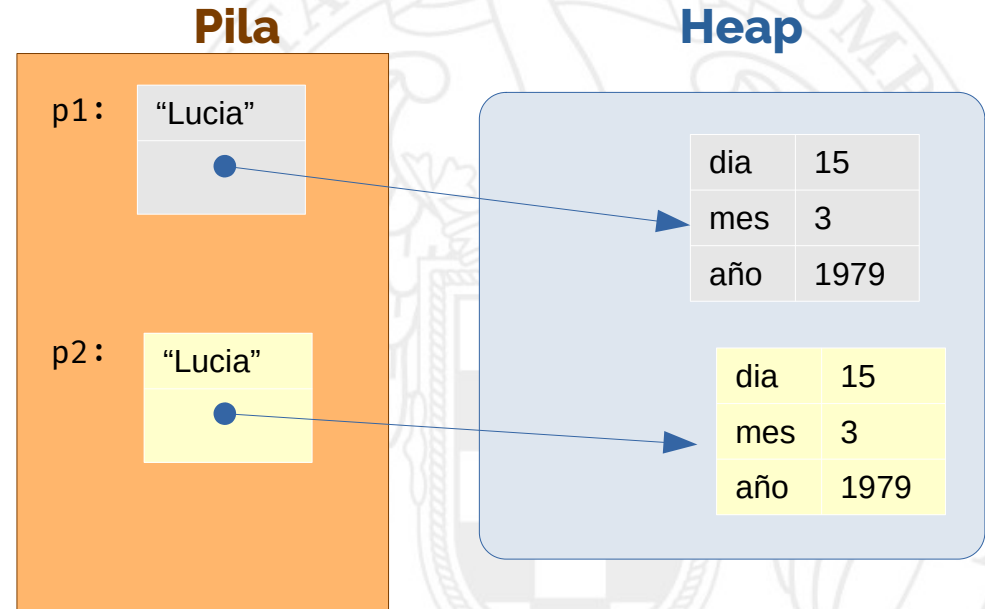
...

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

Asignar un objeto Persona a otro

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

```
p2 = p1;
```



Otra posibilidad

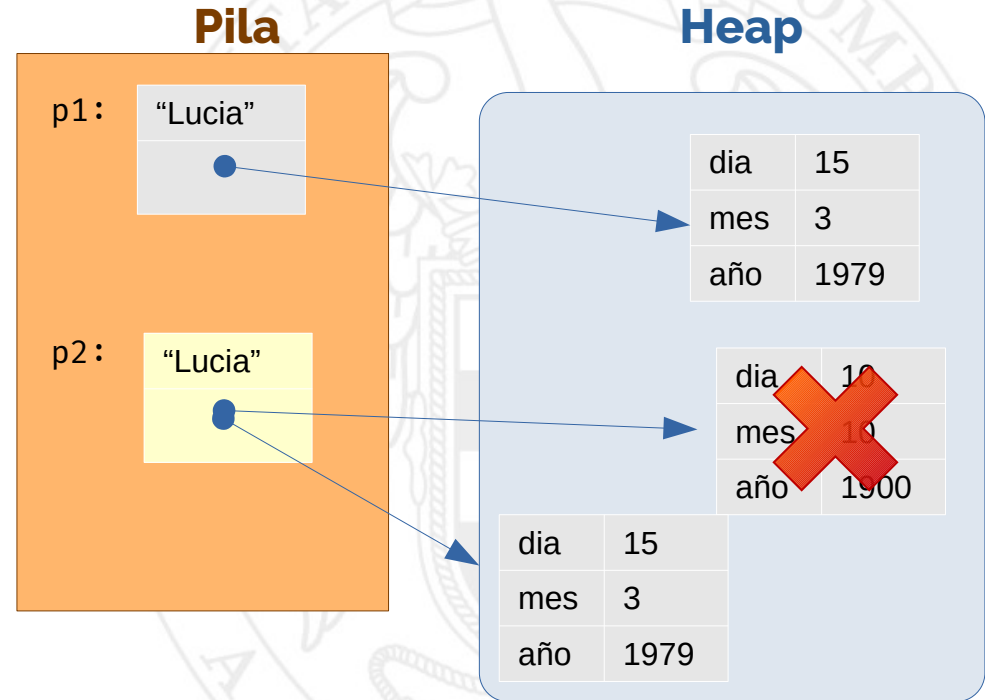
```
class Persona {  
public:  
  
...  
  
void operator=(const Persona &other) {  
    nombre = other.nombre;  
    delete fecha_nacimiento;  
    fecha_nacimiento = new Fecha(*other.fecha_nacimiento);  
}  
...  
  
private:  
    std::string nombre;  
    Fecha *fecha_nacimiento;  
};
```



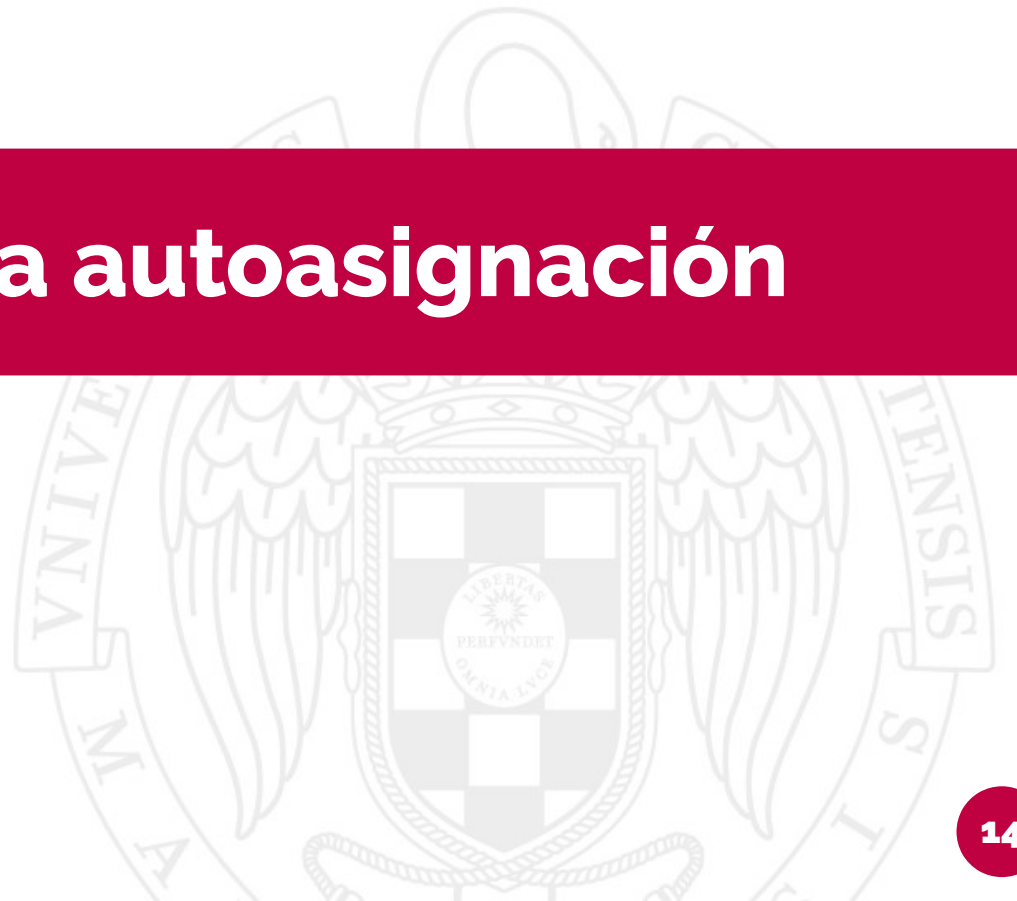
Asignar un objeto Persona a otro

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

```
p2 = p1;
```



El problema de la autoasignación

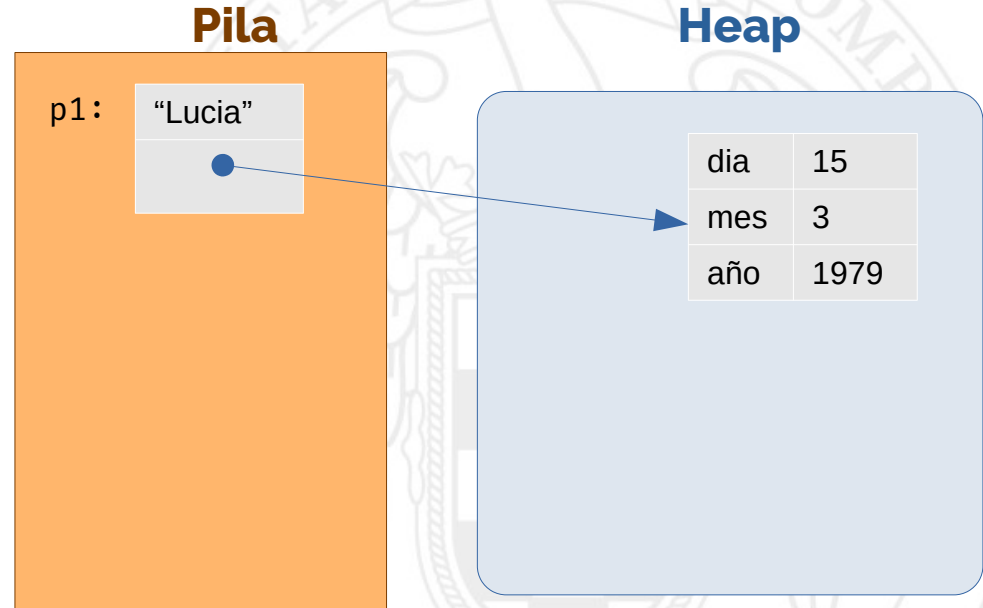


Asignar un objeto persona a sí mismo

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

```
p1 = p1;
```

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



Evitando la autoasignación

```
class Persona {
public:

    ...

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

    ...

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



Encadenar asignaciones



Encadenar asignaciones

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

x = y = z = 0;

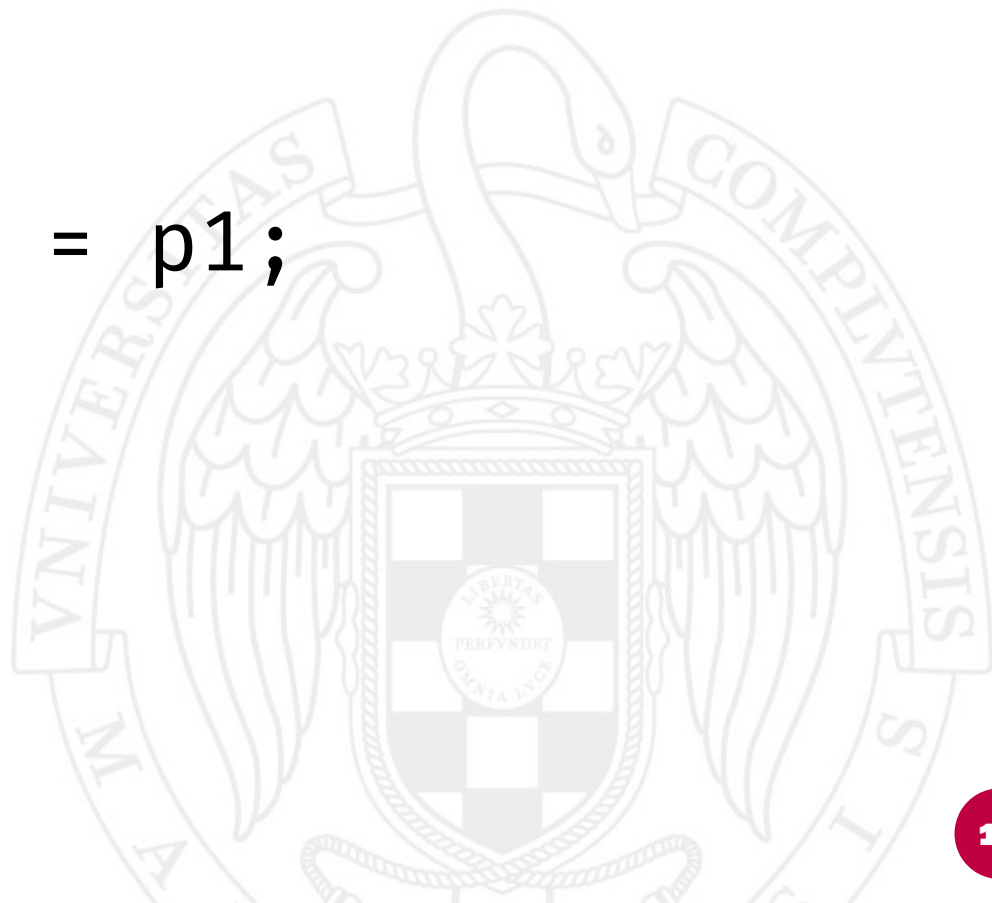


¿Podemos hacer lo mismo con objetos Persona?

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

p3 = p2 = p1; ❌

p3 = p2 = p1;




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

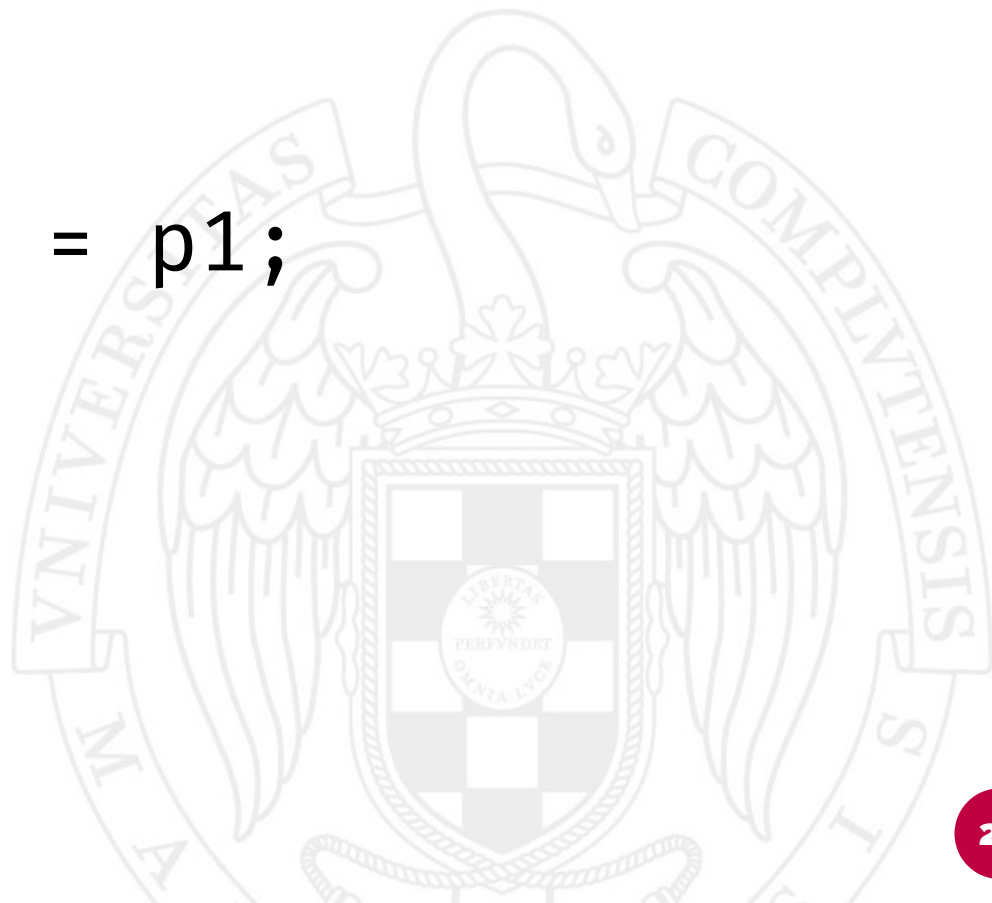


¿Podemos hacer lo mismo con objetos Persona?

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

p3 = p2 = p1; 

p3 = p2 = p1;



Constructor de copia vs. Operador asignación

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

```
Persona p1(...);  
Persona p2 = p1;
```

- No devuelve nada.
- No puede producirse autoasignación:

```
Persona p2 = p2;
```

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

```
Persona p1(...);  
Persona p2(...);  
p2 = p1;
```

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

```
p2 = p2;
```