

ESTRUCTURAS DE DATOS

NOTAS SOBRE C++

# Destructores

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;  
};
```



# Recordatorio: clase Persona

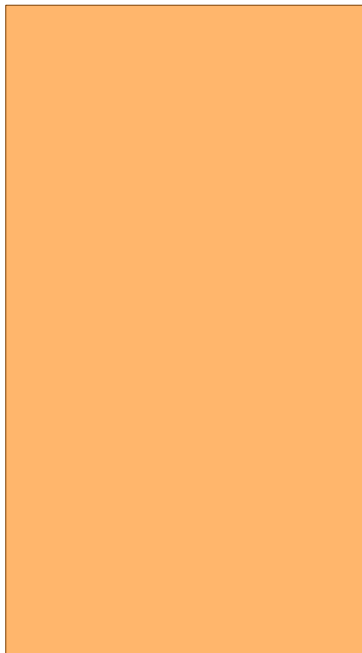
```
class Persona {  
public:  
    Persona(std::string nombre, int dia, int mes, int anyo)  
        : nombre(nombre), fecha_nacimiento(dia, mes, anyo) { }  
  
private:  
    std::string nombre;  
    Fecha fecha_nacimiento;  
};
```



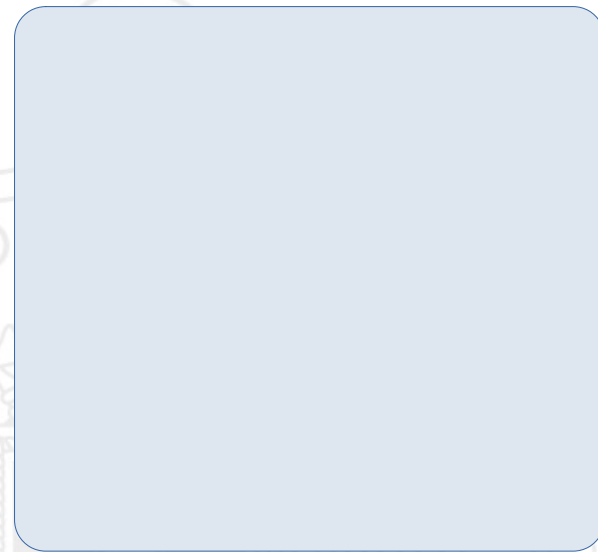
# Ejemplo de uso

```
void ejemplo() {  
    Persona p("David", 15, 3, 1979);  
}
```

**Pila**



**Heap**



# Cambio en la representación

```
class Persona {  
public:  
    Persona(std::string nombre, int dia, int mes, int anyo)  
        : nombre(nombre) {  
        this->fecha_nacimiento = new Fecha(dia, mes, anyo);  
    }  
  
private:  
    std::string nombre;  
    Fecha *fecha_nacimiento;  
};
```

# Cambio en la representación

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

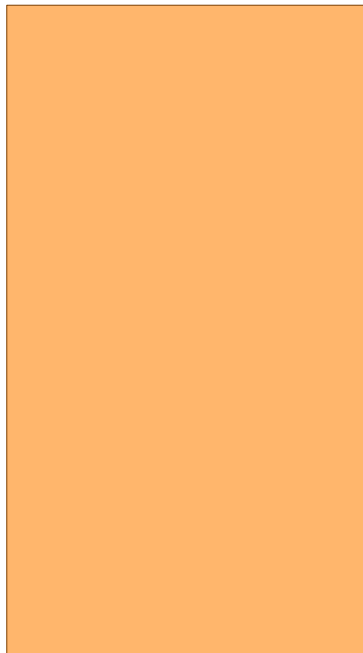


# Ejemplo de uso

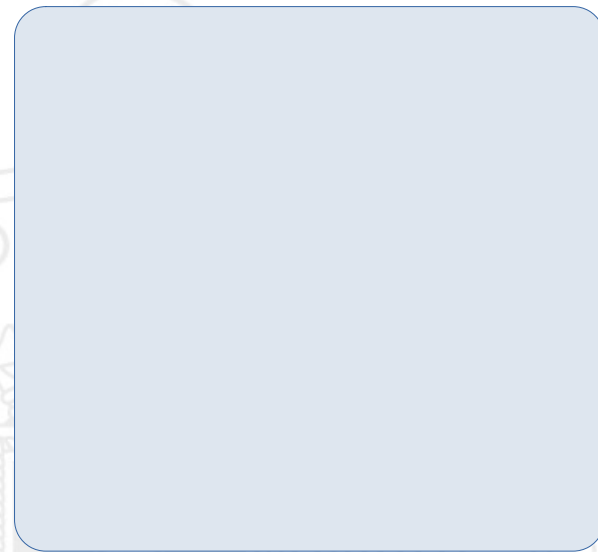
```
void ejemplo() {  
    Persona p("David", 15, 3, 1979);  
  
}
```

Necesitamos una manera de eliminar el objeto Fecha justo antes de que p salga de ámbito


**Pila**



**Heap**



# Destruyores en C++

- Un **destructor** es un método especial que es invocado cada vez que el objeto correspondiente se libera.
  - Si el objeto está en la pila, el destructor es invocado cuando la variable que contiene dicho objeto sale de ámbito.
  - Si el objeto está en el *heap*, el destructor es invocado cuando se aplica `delete` sobre el objeto.
- El nombre del método destructor es el mismo que el de la clase en el que está definido, pero anteponiendo el símbolo `~`.  **AltGr + 4**
- El método destructor no tiene ni parámetros, ni tipo de retorno.



# Añadiendo un destructor a Persona

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

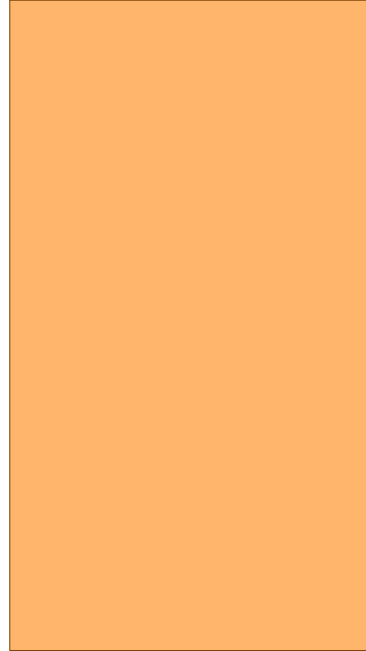


Método destructor

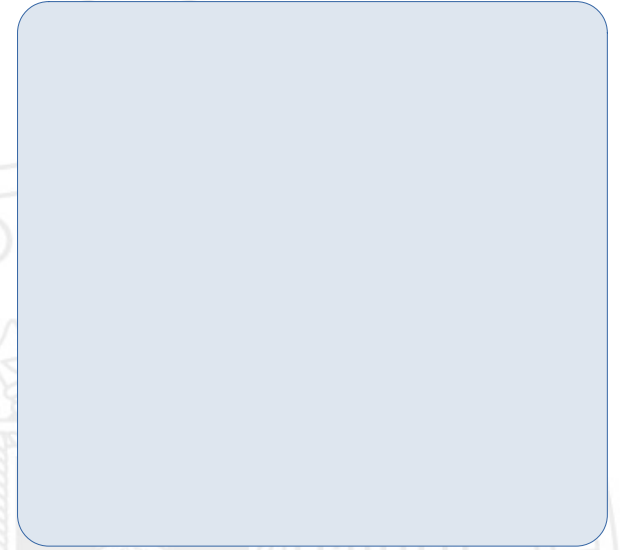
# Ejemplo de uso

```
void ejemplo() {  
    Persona p("David", 15, 3, 1979);  
  
}
```

**Pila**



**Heap**



# ***Resource acquisition is initialization (RAII)***

- En la gran mayoría de casos, la reserva de memoria (`new`) que se realice en el constructor debe tener asociada su liberación (`delete`) en el destructor.
- Excepciones:
  - La memoria reservada se ha liberado antes de invocar el destructor.
  - La memoria reservada está compartida entre varias instancias.
- El principio RAII no solo se aplica a memoria, sino también a otros recursos (apertura/cierre de ficheros, conexiones a bases de datos, etc.)