ESTRUCTURAS DE DATOS

NOTAS SOBRE C++

Constructores de copia

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

NOS ESTAMOS DANDO CUENTA DE QUE LA PROGRAMACIÓN EN C++ ES ALGO MÁS COMPLICADA QUE LA POO DE JAVA

Recordatorio: clases Fecha y Persona

```
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;
};
```

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

  void set_nombre(const std::string &nombre);
  void set_fecha_nacimiento(int dia,
                            int mes,
                            int anyo);

  void imprimir();

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

Puntero a una instancia de la clase fecha

Ejemplo

```
void modificar_copia(Persona p) {
  p.set_nombre("Berta");
  p.set_fecha_nacimiento(10, 10, 2010);
}


int main() {
  Persona david("David", 15, 3, 1979);
  david.imprimir();
  modificar_copia(david);
  david.imprimir();

  return 0;
}
```

Al pasarlo por valor, p va a ser una copia del objeto que yo le pase. En teoría no se debería de ver reflejado

```
Nombre: David
Fecha de nacimiento: 15/03/1979
```

```
Nombre: David
Fecha de nacimiento: 10/10/2010
```

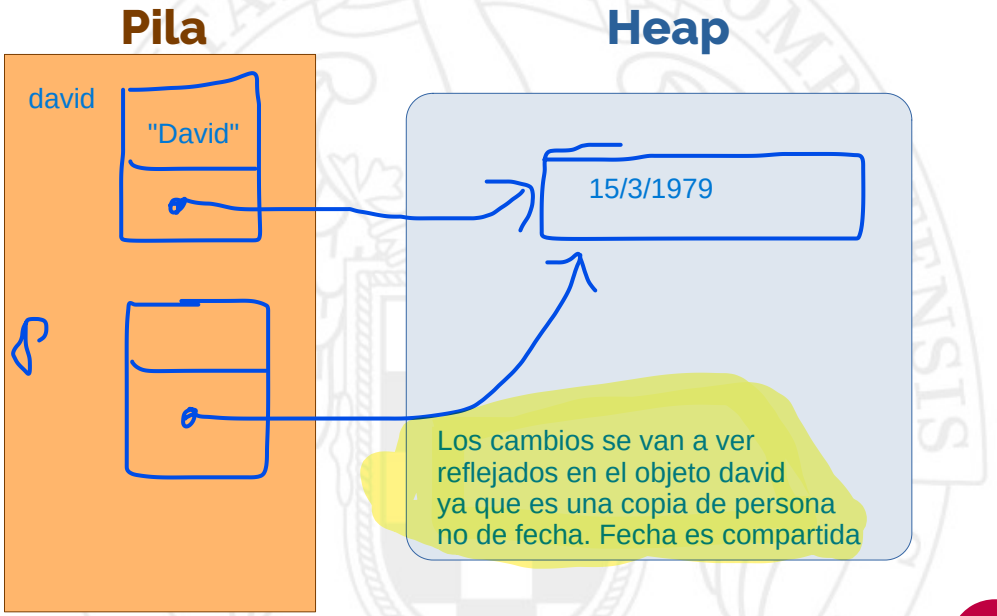
Berta no ha afectado al objeto David, pero si a su fecha de nacimiento

¿Qué ha pasado?

- Cuando se pasa una instancia a una función como parámetro **por valor**, se crea una copia de dicha instancia.
- **¿Cómo se realiza la copia?** Copiando el valor de cada uno de los atributos de la instancia "origen" a la instancia "destino".

```
void modificar_copia(Persona p) { … }


int main() {
  Persona david("David", 15, 3, 1979);
  david.imprimir();
  modificar_copia(david);
  david.imprimir();
   intenta liberar la fecha 2 veces lo cual es negativo y conflictivo
  return 0;
}
```



¿Qué ha pasado?

- Copiar uno a uno los atributos funciona bien en la mayoría de los casos.
- Pero cuando los atributos son punteros a arrays u otras estructuras, solamente se hace una copia del **puntero**, de modo que tanto el objeto original como la copia, **apuntan a la misma estructura**.
- Aún peor: los **destructores** de sendas instancias pueden intentar liberar la estructura compartida **dos veces**. Se harian dos deletes.

¿Puede alterarse el modo en el que se realiza la copia en estos casos?

Tipos de constructores

- **Constructor por defecto** (sin parámetros). ✔
- **Constructor paramétrico**. ✔
- **Constructor de copia**.
- **Constructor *move***.
- **Constructor de conversión.**

Ya hemos visto estos dos constructores.

Ahora vamos a ver este constructor

Constructor de copia

```
class Fecha {
public:
   ...
  Fecha(const Fecha &f);

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

En este caso fecha.

- Es un método con el mismo nombre que la clase.
- Recibe un único parámetro: una referencia constante a un objeto de la misma clase.
- No devuelve nada.

Constructor de copia

```
class Fecha {
public:
   ...
   Fecha(const Fecha &f)
    : dia(f.dia),
      mes(f.mes),
      anyo(f.anyo) { }

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

copio información de f a this

si no ponemos constructor de copia, este se generaría por defecto

Si va a hacer lo mismo, NO MERECE LA PENA DEFINIRLO

- En el caso de `Fecha`, el constructor de copia inicializa los atributos del objeto con los atributos correspondientes del objeto `f` pasado como parámetro.
- Este es el comportamiento por defecto.

Constructor de copia

Aquí el comportamiento por defecto no me vale.

```
class Persona {
public:

  Persona(const Persona &p)
    : nombre(p.nombre) {
    fecha_nacimiento =
      new Fecha(
        p.fecha_nacimiento→get_dia(),
        p.fecha_nacimiento→get_mes(),
        p.fecha_nacimiento→get_anyo()
      );
  }

  ...
}
```

con la información del día mes y anyo.

- En el caso de `Persona`, el constructor de copia inicializa el atributo `fecha_nacimiento` creando un **nuevo** objeto `Fecha`, e inicializa los valores de este último con los de la fecha de `p`.

En vez de esto hacer llamada al constructor de copia de fecha.

Constructor de copia

```
class Persona {
public:

  Persona(const Persona &p)
    : nombre(p.nombre) {
    fecha_nacimiento =
      new Fecha(*p.fecha_nacimiento);
  }
  ...
}
```

constructor de copia de fecha

- También podría haberse llamado explícitamente al constructor de copia de Fecha.

Constructor de copia

```
class Persona {
public:
  Persona(const Persona &p)
    : nombre(p.nombre),
      fecha_nacimiento(
          new Fecha(*p.fecha_nacimiento)
      ) { }
  ...
}
```

- También podría haberse llamado explícitamente al constructor de copia de `Fecha`.

Pero en este caso se ha utilizado una lista de inicialización

¿Cuándo se llama al constructor de copia?

- Cuando se invoca explícitamente al crear un objeto.

```
Persona p1("David", 15, 3, 1979);
Persona p2(p1);
```

p2 tendría mismo info que p1

- Cuando se declara una variable y se inicializa desde otro objeto.

```
Persona p1("David", 15, 3, 1979);
Persona p2 = p1;
```

Aquí se llama al constructor de copia.

- Cuando se pasa un parámetro por valor.

```
bool es_navidad(Fecha f) { ... }
…
Fecha f1(15, 3, 1979);
if(es_navidad(f1)) { … }
```

Copia se genera a través del constructor de copia.

- Cuando se devuelve un objeto como resultado.

```
Fecha nochevieja(int anyo) {
  Fecha result(31, 12, anyo);
  return result;
}
```

Se copia la información devuelta

¿Cuándo NO se llama?

- Cuando se asigna un objeto a una variable inicializada previamente.

```
Persona p1("David", 15, 3, 1979);
Persona p2("Gerardo", 1, 2, 1983);
p2 = p1;
```

Asignación campo por campo

No se llama al constructor de copia

```
Persona p1("David", 15, 3, 1979);
Persona p2 = p1;
```

Ya que esto es inicialización.

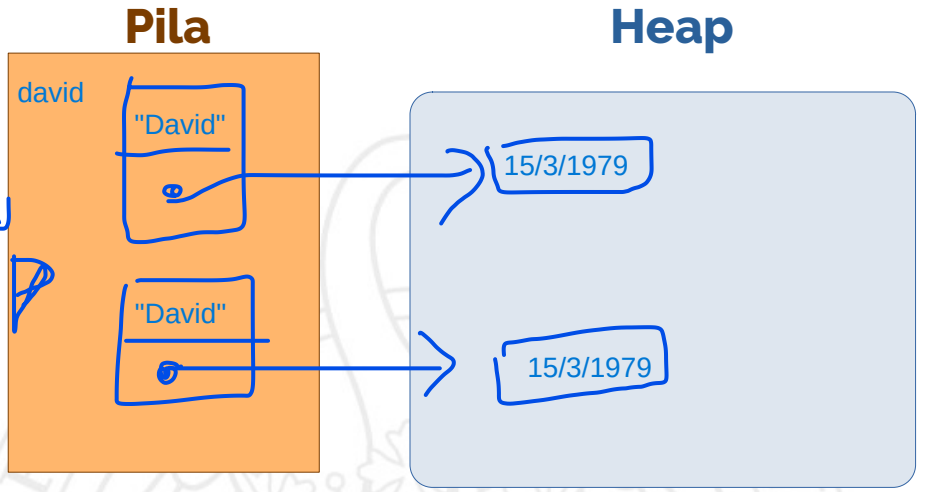
Sí se llama al constructor de copia

Volviendo a nuestro ejemplo...

```
void modificar_copia(Persona p) {
  p.set_nombre("Berta");
  p.set_fecha_nacimiento(10, 10, 2010);
}


int main() {
  Persona david("David", 15, 3, 1979);
  david.imprimir();
  modificar_copia(david);
  david.imprimir();

  return 0;
}
```




```
Nombre: David
Fecha de nacimiento: 15/03/1979
```

```
Nombre: David
Fecha de nacimiento: 15/03/1979
```

YA QUE EN ESTE CASO SI HAY CONSTRUCTOR DE COPIA