



FACULTAD DE INFORMÁTICA

Herencia y sobrecarga de operadores

Baltasar Fernández Manjón

<http://www.fdi.ucm.es/profesor/balta/>



Herencia

- La herencia permite reutilizar código de forma sencilla
 - Las clases derivadas no tienen que reimplementar la funcionalidad que ya se ha codificado en la clase base
 - Si no reimplementa un método se usa el de la clase base
 - Desde una clase derivada se puede invocar explícitamente un método de la clase base
 - ClaseBase::MetodoClaseBase()

ClaseBase

ClaseDerivada

Laboratorio de programación II (Facultad de Informática)

- 1



Ejemplo

```

class ClaseBase
{
public:
    void imprimir(ostream&) const;
    void otroMetodoClaseBase(ostream&) const;
};
// función miembro de la clase base
void ClaseBase::imprimir(ostream& out) const
{
    out << " Impresión en ClaseBase " << endl;
}
// función miembro de la clase base
void ClaseBase::otroMetodoClaseBase(ostream& out) const
{
    out << " Impresión en otroMetodoClaseBase " << endl;
}

```

- 2



Ejemplo

```

class ClaseDerivada: public ClaseBase
{
public:
    void imprimir(ostream&) const;
};
// función miembro de la clase derivada
// implementada en función de la clase base
void ClaseDerivada::imprimir(ostream& out) const
{
    ClaseBase::imprimir(out);
    out << " Impresión en metodoClaseBase " << endl;
}

```

```

int main(int argc, char* argv[])
{
    ClaseBase objBase;
    ClaseDerivada objDerivada;

    objBase.imprimir(cout);
    objDerivada.imprimir(cout);

    objBase.otroMetodoClaseBase(cout);
    objDerivada.otroMetodoClaseBase(cout);
}

```

Laboratorio de programación II (Facultad de Informática)

- 3



Que se hereda de la clase base

- En principio se hereda todo menos
 - Constructor y destructor
 - El operador de asignación operator=()
 - Los métodos amigos
- Aunque no se heredan ni el constructor ni el destructor de la clase base, cuando se crea o se destruye un objeto de la clase derivada se invocan automáticamente los de la clase base. Se usa el constructor por defecto (es decir sin parámetros).



Constructores e inicialización

- Si se desea invocar a otro constructor sobrecargado se puede especificar en la definición del constructor de la clase derivada
 - NombreClaseDerivada (parámetros) : NombreClaseBase (parámetros) {}
- Principio general de la inicialización de clases derivadas
 - Se usa el constructor de la clase base para inicializar los miembros correspondientes de la clase base y se añade el código necesario para inicializar los nuevos miembros de la clase derivadas



Operadores que no pueden sobrecargarse

- Los siguientes operadores no se pueden sobrecargar
 - . El operador punto de acceso a clase
 - . * Operador de puntero a miembro
 - :: Operador de resolución de ámbito
 - ? : Operador de expresión condicional



Ejemplo

```
#include <iostream>
using namespace std;

class ClaseBase
{
public:
    virtual void imprimir(ostream&) const;
};

// función miembro de la clase base
void ClaseBase::imprimir(ostream& out) const
{
    out << " Impresión de ClaseBase ";
}

// Se define el operador pero no como función amiga.
// el trabajo se delega en un método de la clase (imprimir)
ostream& operator << (ostream& out, const ClaseBase& objetoBase)
{
    objetoBase.imprimir(out);
    return out;
}
```

```

graph BT
    ClaseDerivada --> ClaseBase
    OtraClaseDerivada --> ClaseBase
  
```



Ejemplo

```
class ClaseDerivada: public ClaseBase
{
public:
void imprimir(ostream&) const;
};
// función miembro de la clase derivada
void ClaseDerivada::imprimir(ostream& out) const
{
out << " Impresión de ClaseDerivada ";
}
// esta otra clase derivada no aporta implementación de imprimir
// por tanto se ejecutará ClaseBase::Imprimir
class OtraClaseDerivada: public ClaseBase
{
};
```

```
int main(int argc, char* argv[])
{
ClaseBase objBase;
ClaseDerivada objDerivada;
OtraClaseDerivada objOtraDerivada;

cout << objBase;
cout << objDerivada;
cout << objOtraDerivada;
}
```