# Programación Orientada a Objetos I

Prof. Andrés Arcia
Departamento de Computación
Escuela de Ingeniería de Sistemas
Facultad de Ingeniería
Universidad de Los Andes

# Qué es un paradigma de programación?

- Es un conjunto de conceptos que dirigen la manera en la que construimos un programa. Como se diseñe la solución depende de los mecanismos, conceptos y maneras que un paradigma nos diga.

# Algunos Paradigmas Conocidos

- Imperativo — C, Pascal, Basic, PHP, Perl.
- Funcional — Lisp, Scheme.
- Lógico — Prolog.
- Orientado a Objetos puros: Smalltalk, Eiffel.
- Orientado a Objetos hibridos: C++, Java.

Observe la distinción entre los lenguajes OxO puros e hibridos: En C++ o java, por ejemplo, existen constructos utilizados en la programación imperativa: while, for. Estos son modelados como objetos en los LOxO puros.

# Adaptabilidad del Lenguaje

- Es posible escribir código en un lenguaje a través de un paradigma que no implante ese lenguaje?
  - R: Si, por ejemplo, Ud. podría escribir un codigo OxO en C, sin embargo, reflexione acerca de la dificultad para escribir el codigo y los riesgos.

# Historia de la OxO

- Durante los años 80, persistia un problema en la programación: no se podía iniciar y finalizar valores en los tipos de datos y de forma automática.
- Qué se puede hacer durante una inicialización: abrir archivos, apartar memoria, copiar datos predefinidos.
- Qué se puede hacer en una finalización: cerrar archivos, liberar memoria.

# Historia de la OxO

- Entre otros problemas conocidos estaban: La ampliación sencilla de las operaciones sobre tipos de datos, reutilización de la interfaz de forma segura y sin riesgos de inconsistencia de tipos.
- Estos problemas se resolvieron con la abstracción *CLASE;* el bloque mínimo de construcción para la OxO.

# Justificación de la OxO

- La orientación por objetos es un paradigma que nace de la necesidad de poder expresar la solución de un problema en un lenguaje apegado al problema.
- Ofrece un mecanismo poderoso de abstracción de ideas, que pueden ser independizadas y relacionadas, mediante mecanismos sencillos.

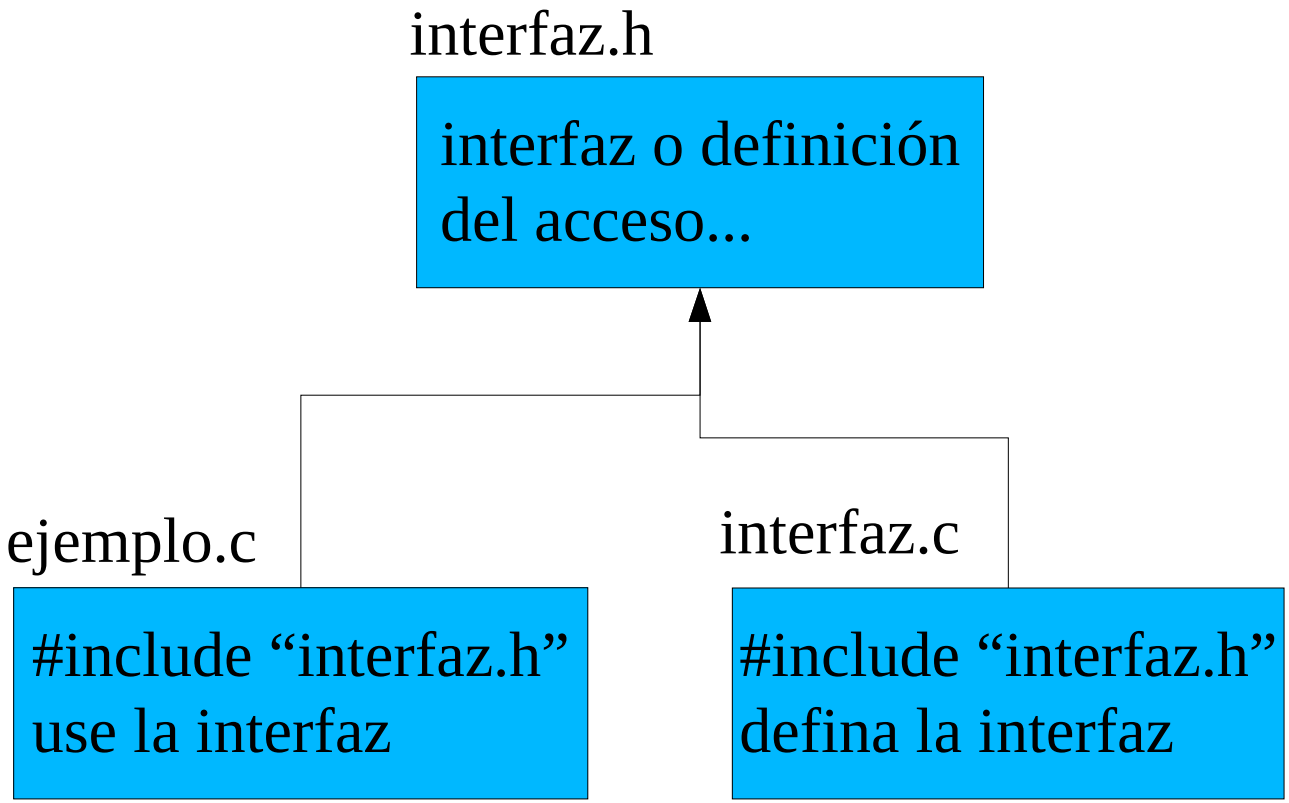
# ¿Cómo llegar a la OxO?

- Programación Procedural: decida que procedimientos desea, luego utilice los mejores algoritmos que encuentre.
- Programación Modular: Decida que módulos desea, luego parta el programa de tal forma que la data sea ocultada por los módulos.

# ¿Cómo llegar  a la OxO?

- El paradigma de la programación modular es también conocido como el *principio del ocultamiento de la data*. Como hacer:
  - Busque una interfaz para manejar la data.
  - Asegurese que la data sea solamente accesible a través de dicha interfaz.
  - Asegurese que dicha data sea inicializada siempre, antes de su primer uso.

# ¿Cómo llegar a la OxO?



# ¿Cómo llegar a la OxO?

- Abstracción de la data: la modularidad es el pilar de todos los programas de gran escala.

# ¿Cómo llegar a la OxO?

- Tipos definidos por el usuario:
  - C++ ataca este problema, permitiendo que un usuario pueda declarar los tipos casi tal como se hace con los propios tipos internos.
  - El paradigma es:

  Decida que tipos quiere, luego especifique cuales son las operaciones sobre esos tipos.

# Un Ejemplo ;-)

- ¿Recuerda qué es un número complejo?

```
class complex {
float real;
float img;
public:
complex (float r, float i) {real=r; img=i;}
complex (float r) {real=r; img=0;}
complex operator+(complex _a)
{ return complex(_a.real + real, _a.img + img); }
};
```

# Tipos Abstractos

- Un tipo abstracto, no está definido del todo. Especifica una interfaz y posteriormente otras clases podrán definir, concretamente la funcionalidad.
- En C++ un tipo de dato abstracto está compuesto de funciones virtuales puras. Esto le dice al programador, que son funciones que podrían ser especificadas luego, en clases derivadas.

# Tipos Abstractos. Ejemplo.

- Suponga que se desea construir una interfaz para tratar figuras geométricas.

```
class figGeom {
char * nombre[30];
public:
virtual float area( )=0;
virtual float rotar( )=0;
};
```

# Tipos Abstractos. Ejemplo.

```
class triangulo : public figGeom {
coordenada_punto p1, p2, p3;
public:
float area() {
// calcula area
}
...
};
```

El constructo “: public” se lee como deriva de, implementa a, hereda de, es un subtipo de.

# Funciones Virtuales

- Las funciones virtuales son alojadas en una tabla de funciones virtuales por cada *clase* derivada. Esto permite que en una función como la siguiente:

```
void calcula_areas(figGeom g[], int n)  {
 for (int i=0; i<n; i++)
    cout << g.nombre << g.area() << endl;
    }
```

Las areas correspondan a cada figura en particular.

# Gerarquía de Clases

- Fijese que figGeom es la clase base para triangulo y cuadrado. Otra forma de decirlo es que figGeom es una superclase y triangulo y cuadrado son subclases.
- Las clases derivadas heredan los miembros de las clases bases. Luego este acto se le denomina *herencia*.

# Gerarquía de Clases

- Este paradigma de programación es:
  - Decida que clases desea, provea a cada clase con un conjunto completo de funcionalidades y luego muestre lo que es común utilizando la herencia.
  - Encontrar lo que es común *no es fácil*.

# Programación Generica

- Con la programación generica se desea evitar hacer un algoritmo por tipo de dato, si el algoritmo puede llegar a ser el mismo.
- Por ejemplo, buscar, ordenar, intercambiar, etc. son funciones bien definidas de forma independiente del tipo de dato.
- En C++ el mecanismo para la programación genérica se le denomina *template* ó plantilla.

# Programación Generica. Ej. 1

- La siguiente es una función tipo plantilla para intercambiar valores:

```
template <class T> void swap(T & left, T & right)
{  T tmp = left;
   left = right;
   right = tmp; }
```

Para intercambiar un par de enteros sería:

```
int a = 5;
int b = 9;
swap<int>(a,b); // aqui sucede el intercambio...
```

# Programación Generica. Ej 2.

```
template <class In, class Out> copy (In * from, In * finish, Out * to)
{
   for (In * o=from, Out * d=to; o!=finish; o++, d++)
        d=o;
  d=o;
}
```

# ¿Existe algún esquema para desarrollar progs. OxO?

# Abstracción de los actores (objetos). ¿Cómo se hace?

- Identifique los datos, las operaciones y luego la interfaz para operar sobre esos datos.