# Práctica 8
# Polimorfismo

(Rev 1.0)

# 1 Objetivos

Que el alumno comprenda la herencia, la jerarquía de herencia y los aspectos básicos de su implementación en C++.

# 2 Pre-laboratorio

## 2.1 Cuestionario

1. ¿Para qué se utiliza la palabra reservada `auto`?

2. ¿Qué es el range-`for` (o for con rango)?

3. ¿Para qué se utiliza la palabra clave `override`?

4. Cuando se utilizan métodos virtuales se aconseja declarar y definir al destructor como virtual en todas las clases, aunque no lo use, ¿porqué?

## 2.2 Actividades

1.

Entregables:

# 3 Teoría

# 4 Actividades

## Actividad 1: Probando el polimorfismo

1. Compile, ejecute y escriba la salida del archivo **polimorfismo1.cpp**.
2. Elimine la palabra `virtual` y `override` de todos los lugares donde aparezcan de este mismo programa. Compile, ejecute y escriba la salida del programa. ¿Qué sucedió? Explique.

## Actividad 2: Creciendo la jerarquía Estudiante

1. Compile y ejecute el programa del archivo **polimorfismo2.cpp**.
2. Mueva el atributo `promedio` a la clase base `Estudiante`. En esta misma escriba el setter/getter correspondiente. Modifique el código de las clases derivadas para que reflejen este cambio. Compile y ejecute.
3. Agregue la clase `BajaTemporal` que se deriva de la clase base `Estudiante`. Esta clase tendrá un atributo `tiempoDeBaja` del tipo entero que representará el número de semestres que el alumno estará en baja temporal. Este atributo se establecerá en el constructor con argumentos y su valor mínimo es de 1 semestre.
4. En el driver program agregue un objeto tipo `BajaTemporal`, agregue una entrada en el menú para procesar este nuevo objeto, y agregue su caso correspondiente en el `switch`. Compile y ejecute.

## Actividad 3: Usando referencias en lugar de apuntadores

El código de la actividad anterior utilizó un apuntador a la clase base como objeto polimórfico. En esta actividad Ud. usará referencias.

1. Guarde el archivo anterior con el nombre **polimorfismo_ref.cpp**.
2. Agregue en la clase base `Estudiante` un método virtual `Iam()`, del tipo `void`, de sólo lectura y sin argumentos. El cuerpo de este método únicamente imprimirá el nombre de su clase. Todas las clases derivadas deberán redefinirlo. No olvide marcar a estas redefiniciones como `virtual` y `override`.
3. Cambie la firma de la función `Imprimir()` para que reciba como argumento una referencia a un objeto de tipo `Estudiante`. En el cuerpo refleje este cambio.
4. Escriba una función `YoSoy()` que reciba como argumento una referencia a un objeto de tipo `Estudiante`. Dentro del cuerpo mande llamar al método `Iam()`.
5. Modifique los casos del `switch` para que cada uno mande llamar en secuencia a las funciones `YoSoy()` e `Imprimir()` y con los argumentos correctos. Borre la línea `Imprimir(p)`.que está terminando el `switch`. Compile y ejecute.
6. Explique porqué los casos del `switch` tuvieron que cambiar ahora que está usando referencias en lugar de apuntadores.

## Actividad 4: Colecciones de objetos polimórficos

1. En el archivo **polimorfismo3.cpp** modifique el driver program agregando un ciclo `for` tradicional que haga lo mismo que el range-`for` actual.

¿Qué forma se le hizo más fácil, el range-`for` o el `for` tradicional, y porqué?