



Unidad VI Herencia y plantillas

M.C. Juan Carlos Olivares Rojas



Agenda

- 6.1 Concepto de herencia
- 6.2 Herencia múltiple
- 6.3 Miembros privados y amigos
- 6.4 Empleo de plantillas de funciones
- 6.5 Empleo de plantilla de clase

6.1 Concepto de herencia

- La herencia es un tipo sutil de relación entre los diferentes objetos.
- La herencia hace que las clases puedan especializarse y puedan tener datos genéricos y comportamientos similares.
- Por ejemplo un canario hereda los atributos y propiedades de una ave

Herencia

- No debe confundirse la relación es un (is-a) de herencia con la relación (have-a) composición.
- En relaciones de composición, un objeto está compuesto por otro. Por ejemplo un automóvil está compuesto por 4 llantas, un volante, un motor, etc.

Herencia

- Para definir que una clase hereda de otra se antepone el operador : al nombre de la clase.

```
class padre
{
    protected:
        int p1,p2;
        float m1();
        float m2()
}
```

```
class hijo: padre
{
    private:
        int a;
    public
        short yy();
}
```

- hijo A = new hijo
- A.p1 = 10;
- A.m2();
- A.yy();

Herencia

- Los tipos de datos públicos se convierten en públicos siempre al heredar, al igual que los privado.
- Los tipos de datos protected se convierten en públicos al heredar.

6.2 Herencia múltiple

- La herencia múltiple es ampliamente utilizada en la vida real, una clase hija no sólo hereda de su clase padre, sino que también hereda de una clase madre.
- La herencia múltiple se hace al igual que la simple, sólo enlistando cada una de las clases de las cuales se hereda.

Herencia múltiple

- El problema de la herencia múltiple radica en cuando se comparten métodos y propiedades con el mismo nombre, el compilador sufre en batallar a que clase padre se invoca cada método y propiedad.
- Ejemplo: class hijo: padre, madre {...}
- La herencia se utiliza en el diseño de mucho software actualmente.

6.3 Miembros privados y amigos

- Los miembros privados de una clase sólo son visibles desde la clase en la que se definieron o desde alguna que haya heredado.
- Los miembros amigos declarados con la variable `friend clase_amiga` pueden ser accedidos por otros miembros.

6.4 Empleo de plantillas de funciones

- Las plantillas de funciones son muy utilizadas debido a que permiten hacer portable los algoritmos independientes del tipo de datos.
- Así por ejemplo una estructura de datos del tipo pila realiza las mismas instrucciones independiente si el tipo de datos almacenados es entero, flotante, etc.

Plantillas de funciones

- Existe una biblioteca estándar de plantillas en C++ que utilizan los algoritmos más comunes denominada STL (Standard Template Library)
- Las plantillas utilizan la palabra clave `<template>` T, para indicar un tipo de dato, posteriormente la T realiza las operaciones normales

Plantilla de funciones

- `<template> T class clase`
- `{`
- `<template> T dato;`
- `public:`
- `<template> T setDato (float a, <template>`
`T);`
- `}`

6.5 Empleo de plantilla de clase

- Las plantillas se definen de acuerdo al tipo de datos.

```
<int> T class  
{  
    <int> T dato;  
    ...  
}
```

¿Preguntas?

