Emilio Gomez Breschi

19300100

4D1

Sonia Erika Ibáñez de la Torre

Desarrollo de Software

26/Octubre/2021

Programación Orientada a Objetos

¿Que es la herencia simple y para que es utilizada ?

En la herencia, las clases derivadas “heredan” los datos y las funciones miembro de las clases base, pudiendo las clases derivadas redefinir estos comportamientos y añadir comportamientos nuevos propios de las clases derivadas. Para no romper el principio de encapsulamiento, se proporciona un nuevo modo de visibilidad de los datos/funciones: “protected”. Cualquier cosa que tenga visibilidad protected se comportará como pública en la clase Base y en las que componen la jerarquía de herencia, y como privada en las clases que NO sean de la jerarquía de la herencia.

La herencia permite que una clase tenga el mismo comportamiento que otra clase y amplíe o adapte ese comportamiento para brindar una acción especial para necesidades específicas. La herencia puede ayudar a representar objetos que tengan algunas diferencias y algunas similitudes en la forma en que funcionan. (CDMYPE-Matías, 2012)

Sintaxis de la herencia simple de la clase heredada y el constructor de la clase heredada.

```
package
{
        import A;
        public class B extends A
        {
                public function B(n:uint)
                {
                        super(n); //ejecuta el contructor de la clase A, enviandole el
                parámetro
                }
        }
}
```

Ejemplo Completo

```
#include <iostream>

using namespace std;

class Animal {
  private:
    int age;
```

```
  public:
     Animal(int a = 1){
        age = a;
     };
     int getAge() const {return age;}
     void setAge(int a = 9){
        age = a;
     }
};

class Dog:public Animal { };

class Cat:public Animal {
  private:
     float weight;
  public:
     Cat(int a = 2, float w = 3.2):Animal(a){
        weight = w;
     }
     float getWeight() const {return weight;}
};

int main(){
    Dog mastin;
    cout << "____DOG____" << endl;
    cout << "Before: " << mastin.getAge() << endl;
    mastin.setAge(2);
    cout << "After: " << mastin.getAge() << endl;
    Cat miau(3, 4.1);
    cout << "____CAT____" << endl;
    cout << miau.getAge() << " " << miau.getWeight() << endl;
}
```

(Zguillez, 2009)

Que es la herencia múltiple y para que es utilizada, así como la diferencia con la herencia simple ?

Consiste en el ensamblando una nueva clase con los elementos de varias clases-base. C++ permite crear clases derivadas que heredan los miembros de una o más clases antecesoras. Es clásico señalar el ejemplo de un coche, que tiene un motor; cuatro ruedas; cuatro amortiguadores, etc. Elementos estos pertenecientes a la clase de los motores, de las ruedas, los amortiguadores, etc.

Como en el caso de la herencia simple, aparte de los miembros heredados de cada clase antecesora, la nueva clase también puede tener miembros privativos.

Sintaxis de la herencia múltiple en la clase heredada y el constructor de la clase heredada.

```
class Base1
{  // Declaración de atributos y métodos de la clase Base1
};

class Base2
{  // Declaración de atributos y métodos de la clase Base2
};

class BaseN
{  // Declaración de atributos y métodos de la clase BaseN
};
  // Relación de herencia múltiple entre las clases Base1, Base2, BaseN y Derivada
class Derivada : public Base1, public Base2, public BaseN
{  // Declaración de atributos y métodos de la clase Derivada
};
Derivada :: Derivada (parámetros) : Base1 (parámetros propios de la clase Base1),  Base2 (parámetros propios de la clase Base2), …, BaseN (parámetros propios de la clase BaseN)
{
// Cuerpo del constructor de la clase Derivada
}
```

Ejemplo Completo

```
#include <iostream>
#include <iomanip>      /* Contiene prototipos de función para manipuladores de
flujo que
          dan formato a flujos de datos.  Permitirá formatear y organizar
          la salida de datos. */

using namespace std;

class Persona              // Clase Base Persona
{    protected:
        char nombre[40];
        int edad;
public:
        // Funciones Miembro
        Persona( ) { };             // constructor por defecto
        void leerdatos( );
        void imprimirdatos( );
};

// Declaración del Método para dar valor a los atributos de la clase Persona
void Persona :: leerdatos( )
{    cout << “Digitar el Nombre:" << endl;
     gets(nombre);
     cout << "Digitar la Edad: " << endl;
     cin >> edad;
}

// Método que despliega los valores de los atributos de una persona
void Persona :: imprimirdatos( )
{    cout << endl << endl;
     cout << "----- Imprimir los Datos del Empleado -----" << endl;
     cout << "Nombre : " << nombre << endl;
     cout << "Edad : " << edad << " a";
     printf("%c",164);         // Para mostrar la letra "ñ"
     cout << "os" << endl;
}

/* Definición de la Clase Empleado como clase derivada de la clase Persona. Se
```

```
utiliza
   herencia pública */
class Empleado : public Persona
{    protected:
       float salarioanual;
       char cargo[60];

     public:
       Empleado( ){ };              // constructor por defecto
       void leeremp( );
       void imprimiremp( );
};

// Declaración del Método para dar valor a los atributos de la clase Empleado
void Empleado :: leeremp( )
{    Persona::leerdatos();
     cout << "Introducir Cargo:" << endl;
     cin >> cargo;
     cout << "Introducir Sueldo:" << endl;
     cin >> salarioanual;
}// Método que despliega los valores de los atributos de un empleado
void Empleado :: imprimiremp( )
{   Persona :: imprimirdatos( );
    cout << "Cargo del empleado: " << cargo << endl;
    cout << "Sueldo anual empleado: $ " << fixed << showpoint << setprecision(2)
         <<  salarioanual << endl;
}

int main( )
{   Empleado Employee1;              // Asociar variable con clases
    Employee1.leeremp( );            // Obtener Información de Empleado
    Employee1.imprimiremp( );        // Imprimir Información Empleado

    system("pause>nul");
    return 0;
}
```

(Peña, 2018)

Referencias

CDMYPE-Matías. (Octubre de 2012). *cpp*. Recuperado el 26 de Octubre de 2021, de https://inicioscpp.wordpress.com/herencia-simple/

Peña, V. N. (2018). *StuDocu*. Recuperado el 26 de Octubre de 2021, de https://www.studocu.com/bo/document/universidad-mayor-real-y-pontificia-san-francisco-xavier-de-chuquisaca/programacion/guia-9-herencia-simple-y-multiple-en-c/5956506

Zguillez. (07 de Febrero de 2009). *Cristalab*. Recuperado el 26 de Octubre de 2021, de http://www.cristalab.com/tutoriales/programacion-orientada-a-objetos-herencia-de-clases-c261l/