

Es C propiamente un lenguaje orientado a objetos? - Realizado por Aar3n Mizrachi en fecha 24/04/2004

Es C propiamente un lenguaje orientado a objetos?

Por Aar3n Mizrachi <unmanarc@gmail.com>, 24/04/2004 – 22:35

<http://www.unmanarc.com/v1/?q=node/6>

Revisi3n 1.1, 24/05/2009

Es C propiamente un lenguaje orientado a objetos? - Realizado por Aar3n Mizrachi en fecha 24/04/2004

C es un lenguaje muy extenso, y permite mucho el manejo de memoria, sin embargo, la pregunta original es si podemos transformar el lenguaje C en un lenguaje mas orientado a objetos sin la necesidad de modificar el compilador. Y la respuesta a nuestra duda es que si se puede. No tendr3amos todo el poder de C++, solo podr3amos crear clases interfaz y llenarlas a tiempo de corrida con alg3n m3todo constructor o alguna serie de par3metros que inicialicen la estructura, tampoco podr3amos especificar si un m3todo, subclase o variable, es de modo privado/publico/protegido.

Bueno, en mi af3n por explorar las posibilidades de este mundo de la programaci3n, y llevar al limite las capacidades de un lenguaje como C, me dedique a escribir un c3digo que nos permite transformar C en un lenguaje Pseudo-Orientado a Objetos

El m3todo que proponemos es inicialmente crear una clase que en programaci3n orientada a objetos ser3a una estructura interfaz.

```
typedef struct myclass
{
    int value;
    void (*method)(void *);
} my_class;
```

Inmediatamente, se debe crear ese "method" que aun no esta definido, pero como observaremos, en la compilaci3n del programa, ese m3todo puede servir solo para esta clase y solo tendr3a que ser asignado a cada una de las instancias con una misma referencia, de modo que el consumo de memoria se hace m3nimo entre varias instancias.

Veamos a continuaci3n una funci3n "com3n" que podr3a muy bien encajar con la pseudo-clase "my_class".

```
void some_function(void * local_ref)
{
    printf("Your number is: %d", ((my_class *)local_ref)->value );
}
```

Entonces tenemos una Pseudo-clase interfaz y un m3todo asociado. Ahora veamos como usarlos...

```
int _tmain(int argc, _TCHAR* argv[])
{
    //Is C an object oriented programming language?
    char stop[80];

    //Instance
    my_class objectx;
    objectx.value=21;
    objectx.method=&some_function;

    //Method call
    objectx.method(&objectx);

    //Stop!
```

Es C propiamente un lenguaje orientado a objetos? - Realizado por Aar3n Mizrachi en fecha 24/04/2004

```
fgets(stop,80,stdin);  
  
return 0;  
}
```

Es claro lo que vemos, una inicializaci3n de una pseudo-clase interfaz llamada `objectx`. En principio, le asignamos un valor, el 21, al valor que contiene la "clase", y el m3todo lo definimos como una funci3n, es decir como la posici3n de memoria de la funci3n `some_function`. Es as3 como instanciamos la clase y creamos un objeto con m3todos y valores, e incluso podr3a tener referencias a otras pseudo-clases similares.

Para llamarlo, debemos tambi3n incluir la referencia a la propia clase, esto debido a que a tiempo de corrida no hay forma de saber cuales son las variables de la clase dentro del m3todo. Quiz3 esto podr3amos ocultarlo a la hora de programar con alguna "macro" de C.