Capítulo 15. miembros y estáticos

Todos los objetos comparten el mismo espacio de almacenamiento static para ese atributo, constituyendo una forma de «comunicarse» entre ellos. Pero los datos static pertenecen a la clase; su nombre está restringido al interior de la clase y puede ser public, private o protected.

La definición debe realizarse fuera de la clase (no se permite el uso de la sentencia inline), y sólo está permitida una definición. Es por ello que habitualmente se incluye en el fichero de implementación de la clase. La sintaxis suele traer problemas, pero en realidad es bastante lógica. Por ejemplo, si crea un atributo estático dentro de una clase de la siguiente forma:

Deberá definir el almacenamiento para ese atributo estático en el fichero de definición de la siguiente manera:

class A {

static int i;

public:

//...

};

Usando static const de tipos enteros puede realizar las definiciones dentro de la clase, pero para cualquier otro tipo (incluyendo listas de enteros, incluso si estos son static const) debe realizar una única definición externa para el atributo. Estas definiciones tienen enlazado interno, por lo que pueden incluirse en ficheros de cabecera.

#### Variables estáticas locales

Lo más común es declarar una **variable** local **estática**:

|  |
| --- |
| void mi\_funcion\_o\_metodo(void)  {      static int var\_estatica = 4;  } |

**El valor de la variable** var\_estatica **no se pierde aunque ya no se encuentre dentro del alcance**, es decir, cuando el programa haya salido de la función en que la variable fue definida. Esto diferencia una variable estática de una variable no estática, cuyo valor se pierde al final de la función.

Debido a esta diferencia, las variables estaticas no se guardan en el stack sino en una memoria especial para variables estáticas.

En el código siguiente definimos una clase con una variable miembro estática:

|  |
| --- |
| class MiClase  {  public:      static int mi\_variable;  }; |

Esta variable se inicializa así:

|  |
| --- |
| int MiClase::mi\_variable = 5; |

Si la queremos usar sin instancia, entonces usamos el nombre cualificado:

|  |
| --- |
| void haz\_algo(void)  {      MiClase::miVariable = 3;  } |

#### Constantes estáticas

Variables estáticas se pueden declarar también const. Una **constante estática es una constante universal** del programa, **pero de alcance limitado**. Es una buena práctica utilizar constantes estáticas en lugar de macros #define ya que las constantes llevan un tipo: un 5 puede ser un int, un char o un float pero un dato

|  |
| --- |
| const static int var = 5; |

siempre es un int.

#### Métodos estáticos

**Declarar un método estático es una promesa que este método sólo utiliza miembros estáticos** de la clase. Este tipo de método no conoce el puntero this, por lo cual no se puede hacer directamente referencia a métodos y variables no estátitocs de la clase.