Curso C++ - Clase 3

Juan Antonio Zubimendi
azubimendi@lifia.info.unlp.edu.ar

LIFIA

17 de noviembre de 2010

Memoria dinámica

Existen dos tipos de memoria de dinámica disponibles en C++:

- *Stack*: Esta memoria se utiliza para las variables locales a las funciones y métodos. Este memoria será devuelta cuando la función finalice.
- *Heap*: Con esto podemos obtener memoria para variables que necesitemos más allá del alcance de una función.

Heap

Para obtener y liberar memoria dinámica (de la Heap) tenemos dos instrucciones:

- *new*: Con esta instrucción pedimos memoria dinámica al sistema.
- *delete*: Con esta instrucción devolvemos al sistema la memoria asignada a una variable

Podemos obtener memoria para:

- Para crear objetos
- Para tipos básicos u otro tipo de datos.

IMPORTANTE

La administración de la memoria pedida con los operadores *new* y *delete* es total responsabilidad del programador.

new

Con el operador *new* podemos pedir memoria para un objeto o tipo de datos.
El operador nos devuelve un puntero a la entidad recién construida.
De ser necesario se pueden pasar los parámetros necesarios para el constructor.

```
Casa *pCasa = new Casa;
```

delete

Con el operador *delete* liberamos la memoria apuntada por el puntero que recibe como parámetro. El destructor correspondiente a ese objeto es llamado y la memoria liberada.

Punteros

Un puntero nos permite un acceso indirecto a un objeto al cual esta apuntando. Dependiendo al tipo de objeto que apunte, será el tipo de puntero. Para definir un puntero usamos $*$ antes del nombre de la variable:

```
int * pEntero = new int;
```

¿Cómo accedemos al valor "apuntado"?
El acceso al valor apuntado se llama desreferenciar y se puede hacer con el operador $*$:

```
*pEntero = 5;
```

Punteros - continuación

new [] y delete []

Leaks

operador this

Este operador hace referencia a la instancia especifica de la clase y es válido solamente en los métodos de instancia de cualquier clase.

Herencia de clases

Herencia multiple.

Subclases

Para definir una subclase lo hacemos de la siguiente manera:

```
class Cuadrado : public Figura {

};
```

Clases abstractas

$= 0$ en los métodos.

Interfaces

- Podemos usar la herencia multiple para definir interfaces

Métodos virtuales

Que significa? Diferencias con otros lenguajes

Punteros y Polimorfismo