**Programación II, 2015-2016**

**Escuela Politécnica Superior, UAM**

**Práctica 1: Estructuras de Datos y Tipos Abstractos de Datos**

**David Cabornero Pascual**

**Alejandro Santorum Varela**

PARTE 2: PREGUNTAS SOBRE LA PRÁCTICA

**1.** ¿Sería posible implementar la función de copia de nodos empleando el siguiente prototipo **STATUS node\_copy(Node nDest, const Node nOrigin);** ? ¿Por qué?

*No, porque el usuario del TAD desconoce como está implementada tanto la estructura de datos del TAD como la propia función "node\_copy" por lo que deberá pasar por referencia la dirección(puntero) del nodo destino y la dirección(puntero) del nodo origen. Un prototipo viable muy parecido al sugerido es* ***STATUS node\_copy(Node \* nDest, Node \* nOrigin).***

**2.** ¿Es imprescindible el puntero Node\* en **int node\_print(FILE \* pf, const Node\* n);** o podría ser **int node\_print(FILE \* pf, const Node p);** ?

Si la respuesta es sí: ¿Por qué?

Si la respuesta es no: ¿Por qué se utiliza, entonces?

*Sí que es imprescindible, porque nosotros como usuarios del TAD y no como desarrolladores desconocemos como está implementado el TAD Node y por consiguiente, no sabemos como está implementada su estructura de datos, cual es su tamaño y como están los datos almacenados, por lo que la función debe recibir direcciones (punteros).*

**3.** ¿Qué cambios habría que hacer en la función de copiar nodos si quisiéramos que recibiera un nodo como argumento donde hubiera que copiar la información? Es decir, ¿cómo se tendría que implementar si en lugar de **Node\* node\_copy(const Node\* nOrigin)**, se hubiera definido como **STATUS node\_copy(const Node\* nSource, Node\* nDest)**? ¿Lo siguiente sería válido: **STATUS node\_copy(const Node\* nSource, Node\*\* nDest)**? Discute las diferencias.

*STATUS node\_copy(const Node \* nSource, Node \* nDest){*

*/\*Suponemos que el nodo nDest ya viene inicializado del main u otra función\*/*

*if(nDest == NULL || nSource == NULL)*

*return ERROR;*

*strcpy(nDest->name, nSource->name);*

*nDest->id = nSource->id;*

*return OK;*

*}*

*Sí que serí válido, y se implementaría tal que así:*

*STATUS node\_copy(const Node \* nSource, Node \*\* nDest){*

*/\*Suponemos que el nodo nDest ya viene inicializado del main u otra función\*/*

*if(nSource==NULL || \*nDest==NULL || nDest==NULL)*

*return ERROR;*

*strcpy(\*nDest->name, nSource->name);*

*\*nDest->id = nSource->id;*

*return OK;*

*}*

**4.** Indica qué se tendría que cambiar en **graph.c/h** para tener grafos que pudieran almacenar cualquier estructura de datos, es decir, que no estuviera limitado a almacenar nodos definidos como el TAD Node.

*Tendríamos que modificar la estructura de datos del TAD Grafo y en ella introducir todos los TAD que deseemos almacenar en el TAD Grafo. A su vez, tendríamos que modoficar las funciones (tanto el prototipo en graph.h como la implementación de las funciones en graph.c) para que el usuario pueda utilizar la estructura de datos del TAD Grafo en su totalidad.*