Juan Moreno Díez y Camilo Jené Conde

PREGUNTAS SOBRE LA PRÁCTICA

2. ¿Son correctas o no las implementaciones del enunciado?

(a) ==> El primer código es erróneo porque al reservar memoria a n1, no es posible que el ordenador sepa que datos hay dentro del TAD Node.

(b) ==> El segundo código es incorrecto ya que se envía como argumento 'n' y la variable inicializada es 'n1'.

(c) ==> El tercer código es incorrecto ya que se le envía la dirección de memoria de 'n', cuando en ningún momento se inicializa esa variable.

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

No se podría implementar la función de copia de nodos con el prototipo dado anteriormente ya que los argumentos que se le pasan no son de tipo puntero. Eso significa que si la utilizáramos en otro módulo, por ejemplo en el graph.c, con los argumentos pasados de esa forma, no se podría saber cual es la estructura de datos de Nodo ya que estamos implementando el código con la mayor abstracción posible. La forma correcta sería: **STATUS node_copy (Node *nDest, const Node *nOrigin);**

4. ¿Es imprescindible el puntero Node* en el prototipo de la función 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?

De la forma que hemos implementado nosotros la función que imprime el nodo, si que es imprescindible. La primera razón es porque utilizamos las funciones “getters” para obtener la información que queremos y cada una de esas funciones recibe como argumento un puntero a Node. La segunda razón que se me ocurre es muy parecida a la de la pregunta anterior. La función **node_printf**, la utilizamos en **graph.c** y por lo tanto para acceder a la estructura Node, necesitamos pasarle un puntero a la estructura.

5. ¿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.

Se tendría que implementar de la siguiente forma:

```
STATUS node_copy(const Node *nSource, Node *nDest){
  if ( nSource == NULL || nDest == NULL) {
      return ERROR;
  }

  nDest = node_ini();

  if ( nDest == NULL) {
```

```
    return ERROR;
  }

  nDest = node_setName(nDest, node_getName(nSource));
  nDest = node_setId(nDest, node_getId(nSource));
  nDest = node_setConnect(nDest, node_getConnect(nSource));

  return OK;
}
```

6. ¿Por que las funciones del apéndice 4 no deben ser funciones públicas? Justifica la respuesta.

No deben ser funciones públicas porque los prototipos de las funciones privadas dependen de la estructura de datos y solo se pueden utilizar en el módulo en el que estén implementadas.