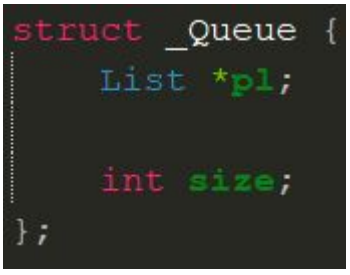
Junco de las Heras, Joaquin Murcia y Borja Velasco

Programación II - Grupo 210

PRÁCTICA Nº3: MEMORIA

1. PREGUNTAS SOBRE LA PRÁCTICA

1.Diferencias entre usar pilas o colas para recorrer un grafo:
La diferencia, básicamente, la explica el apartado c) del primer ejercicio. Al recorrer el grafo usando pilas, la filosofía de trabajo que usamos es LIFO (Last In First Out), mientras que en colas usamos FIFO (First In First Out). Si se utiliza una pila entonces el algoritmo que se implementa es DFS. Este encuentra un camino, pero no garantiza que es más corto. Si se utiliza una cola entonces el algoritmo es BFS, y encuentra el camino más corto asumiendo que las aristas tienen coste 1 (Para encontrar el camino más corto con las aristas de diferentes pesos se tendría que hacer una modificación con una pila de prioridad y pasaría a convertirse en un Dijkstra).



2a).Cambios en ficheros del 3a)
Hemos tenido que cambiar, primeramente, queue.c. Necesitábamos cambiar la estructura de datos de Cola para poder implementarla usando una lista. De este fichero también hemos tenido que cambiar la implementación de todas las funciones, ya que era más económico a nivel de mantenimiento y de cantidad de código hacer llamadas a las funciones de Lista dentro de cada función de Cola.

También hemos añadido list.h y list.c, que no estaban en el ejercicio 1, para poder hacer la implementación con Lista.

2b).Cambios en ficheros del 3b)
No hemos tenido que hacer ningún cambio, lo que muestra que el programa, realmente, funciona independientemente de la implementación que nosotros demos a las funciones fuera del main (luego habría que comprobar cuál de las dos implementaciones es más conveniente).

2. EJERCICIOS

1.TAD Cola

La implementación del TAD Cola no supuso ninguna dificultad significativa, ya que se había visto anteriormente en clase y sabíamos resolverlo. El único elemento llamativo de esta parte sería, tal vez, el uso de los punteros a funciones de la forma aprendida al final de la práctica anterior.

```
struct _Queue {
    void* items [MAXQUEUE];
    int front;
    int rear;
    destroy_element_function_type destroy_element_function;
    copy_element_function_type copy_element_function;
    print_element_function_type print_element_function;
};

Queue* queue_ini(destroy_element_function_type f1, copy_element_function_type f2, print_element_function_type f3) {
    Queue *pq = NULL;
    int i = 0;

    if (!f1 || !f2 || !f3) return NULL;

    pq = (Queue *) malloc(sizeof (Queue));
    if (!pq) {
        fprintf(stderr, "%s", strerror(errno));
        return NULL;
    }

    for (i = 0; i < MAXQUEUE; i++) {
        pq->items[i] = NULL;
    }

    pq->destroy_element_function = f1;
    pq->copy_element_function = f2;
    pq->print_element_function = f3;

    pq->front = 0;
    pq->rear = 0;

    return pq;
}
```

La aplicación de la implementación con el programa de ejemplo tampoco nos dio ningún problema.