.UBAfiuba
FACULTAD DE INGENIERÍA

Tda 1 — Listas

[7541/9515] Algoritmos y Programación II
Segundo cuatrimestre de 2021

Alumno:	de la Rosa, Agustín
Número de padrón:	102875
Email:	adelarosa@fi.uba.ar

Índice

1. Introducción

Un nodo es una estructura que contiene un puntero a un elemento y un puntero al siguiente nodo. De aquí en adelante se usara en todo el trabajo.

Se pide implementar una Lista utilizando nodos simplemente enlazados. Para ello se brindan las firmas de las funciones p´ublicas a implementar y se deja a criterio del alumno la creaci´on de las funciones privadas del TDA para el correcto funcionamiento de la Lista cumpliendo con las buenas pr´acticas de programaci´on. Adicionalmente se pide la creaci´on de un iterador interno y uno externo para la lista, como as´ı tambi´en reutilizar la implementaci´on de lista simplemente enlazada para implementar los TDAS pila y cola.

2. Teoría

Respuestas a las preguntas teóricas:

1. Explique qué es una lista y cómo funciona. Explique qué tipos diferentes de implementaciones conoce.

2. Explique qué es una pila y cómo funciona. Explique qué tipos diferentes de implementaciones conoce

3. Explique qué es una cola y cómo funciona. Explique qué tipos diferentes de implementaciones conoce

1. Una lista simplemente enlazada es una estructura de compuesta de nodos, que, para no llenar la memoria de cadenas de reallocs, se utilizan nodos simplemente enlazadas, los cuales están SOLO conectados a los siguientes nodos. Con esto entonces podemos ordenar datos del orden que quisieramos, a diferencia de las siguientes pilas y colas.

2. Una pila es una estructura compuesta de nodos, que se acumulan como si fueran precisamente, una pila de platos. Es una estructura que se usa SOLO para sacar desde el último ingresado hasta el inicio, en caso de ingresar uno se va para el fondo.

3. Una cola es una estructura compuesta de nodos, que se acumula como si fuera, precisamente, una cola del supermercado. Es una estrúctura que se usa SOLO donde el primero es el primero en sakir, por lo tanto el que entra último va para el final de la cola.

3. Detalles de implementación

Para compilar el programa se usa el makefile dado por la cátedra, donde la función main se encuentra en el ejemplo.c.

Luego, se va a hablar solo de las funciones complejas del trabajo.

3.1. Lista insertar(lista, elemento);

Función que recibe una lista y un elemento y lo inserta en el final, pero antes de poder hacer esto hay que comprobar unas condiciones. Primero si la lista no existe, entonces imposible proseguir, devuelve NULL.

Segundo si la lista está vacía, para saber que hay que referenciar el nodo al inicio y al final, que va a ser tanto el nodo a agregar.

En caso que sea una lista con elementos previos (no vacia) simplemente haces que el siguiente de nodo fin sea el nodo a agregar y luego sin perder la referencia al nodo anterior, le decimos al programa que el nodo a agregar es el nuevo final.

Sumamos la cantidad de la lista en 1.

3.2. Lista quitar(lista);

Sacamos el último elemento de la función. Lo importante de recalcar esta función es que, al no ser doblemente enlazada, se aumenta la complejidad algoritmica de esta función, porque hay que recorrer los n elementos de la lista para poder llegar al ANTEÚLTIMO, que es el que tiene la referencia hacía el nodo a quitar. Cuando lo recorremos hasta el anteúltimo dereferenciamos el último y dejamos al previo como el último.

Para liberar la memoria, no podemos simplemente usar free(nodo) porque la función devuelve el elemento del nodo, pero lo que no necesitamos es el espacio reservado para la estructura, por lo tanto, devolvemos un puntero al elemento y liberamos el espacio del nodo.

Resamos la cantidad de la lista en 1.

3.3. Lista vacia(lista);

Una función booleana que se encarga de la mayoria de condiciones del programa, esta basada en 3 condiciones principales, que exista la lista, que la variable lista cantidad sea 0, y que en caso de emergencia donde se haya roto la memoria en las funciones quitar y agregar, por lo tanto lista cantidad sea erronea, buscamos que lista nodo inicio apunte a NULL, por lo tanto la lista deberá estar vacia.

4. Diagramas

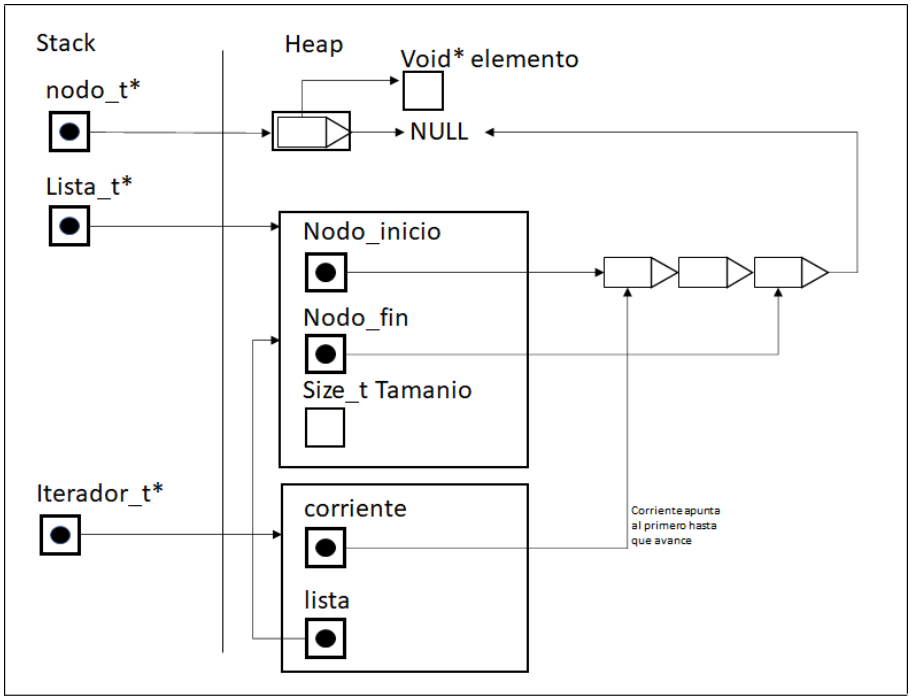

Figura 1: Diagrama de un nodo, una lista, y un iterador sobre esa lista.

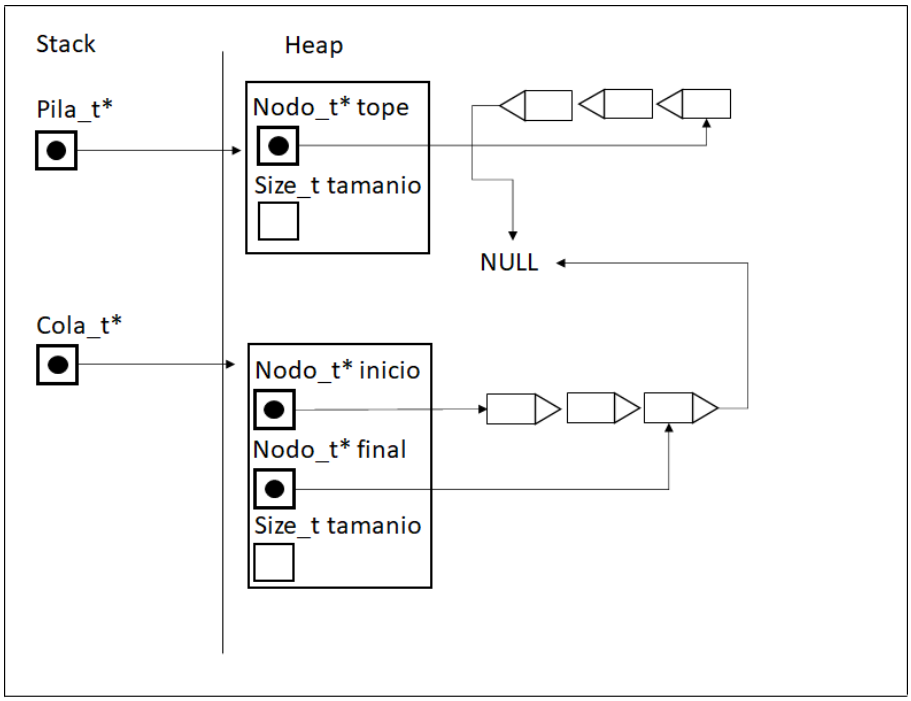

Figura 2: Diagrama de una pila y una cola, es importante recordar que apesar de ser la cola igual a la lista en los diagramas, la cola solo se puede sacar elementos desde el inicio y se agregan para el final.