FACULTAD DE INGENIERIA
Universidad de Buenos Aires

TDA 1 — Lista, Pila y Cola

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

Alumno:	Castillo, Carlos E.
Número de padrón:	108535
Email:	ccastillo@fi.uba.ar

Índice

1. Introducción

Muchos lenguajes de programación permiten al usuario extender los tipos de datos nativos del lenguaje con para facilitar la implementación de características y funcionalidades más complejas. De esta forma, el programador tiene la capacidad de combinar diferentes primitivas del lenguaje y así generar nuevos tipos compuestos para crear interfaces abstractas que faciliten el manejo y organización de la información mediante diferentes estructuras de datos.

El objetivo de este trabajo práctico es implementar una de las estructuras de datos más populares, la lista enlazada, así como dos de sus derivados, pila y cola.

2. Teoría

2.1. TDA Lista

Una lista es una colección lineal de datos cualquiera, en la que los elementos son colocados el uno al lado del otro. Una lista permite al usuario insertar nuevos elementos, ya sea al final de la lista o en una posición específica, borrar elementos en cualquier posición, buscar algún elemento en particular o revisar si la lista esta vacía. Además el usuario puede crear o destruir nuevas listas. A priori la funcionalidad de este tipo de dato se asemeja mucho a la funcionalidad de un arreglo.

Existen muchas implementaciones de lista como listas implementadas con vectores estáticos, listas implementadas con vectores dinámicos o listas circulares, pero en particular, una de las más interesantes es la implementación de lista enlazada compuesta por nodos, donde un nodo es otro tipo de dato abstracto que además de almacenar el elemento en cuestión, puede conocer al nodo que le sucede o que le antecede.

2.1.1. Lista Enlazada

La implementación de lista basada en nodos enlazados permite el almacenamiento no secuencial de los datos en memoria, lo que significa que cada uno de los elementos individuales puede ser almacenado en cualquier otra parte de la memoria. Para mantener el la estructura lineal y continua de la lista establecida por el usuario, se utiliza alguna de las referencias que cada nodo tiene con respecto a su antecesor o sucesor, así se puede recorrer la lista de nodo en nodo ordenadamente, avanzando al nodo siguiente en cada caso. En algunos tipos de lista enlazada, conocidas como listas doblemente enlazadas, cada nodo puede tener al mismo tiempo una referencia a su antecesor y sucesor, lo que permite tanto ”avanzar” como ”retroceder” en cualquier punto de la lista.

Esta forma de organizar los datos en memoria facilita muchas de las operaciones de la lista, principalmente la inserción y eliminación de elementos, ya que, como los elementos no están almacenados en bloques de memoria continuos, cuando se agrega o se quita un elemento en particular no es necesario que se le reasigne memoria a la lista ni que se tengan que reorganizar todos los demás elementos de la misma.

2.2. TDA Pila y Cola

De la estructura de datos lista se pueden derivar otras dos estructuras similares, conocidas como pila y cola, las cuales también pueden ser construidas a partir de nodos. Además de esta versión, existen otras variantes de estas estructuras, que pueden o no implementarse a partir de una lista enlazada, como lo son la pila con vector estático o dinámico, cola con vector estático, cola con vector dinámico y cola circular, creada con la intención de reutilizar un vector estático en una cola.

Las operaciones de pila y cola divergen levemente de las operaciones del tipo lista, aunque para una implementación basada en nodos, estas operaciones pueden ser consideradas casos particulares de las operaciones estándar de una lista enlazada.

La estructura de datos pila consiste en una colección de elementos organizados de manera lineal en las que solo se tiene acceso al elemento que se encuentra en el tope de la pila, es decir, el último

elemento agregado. De esta forma, al agregar un elemento a la pila, solamente se puede agregar sobre el tope actual de la misma. De igual manera, solamente se puede quitar el último elemento de la pila. Esto implica que para quitar un elemento en medio de la pila, primero se tienen que quitar todos los elementos que fueron agregados después de ese elemento, o los elementos que están encima. Este método para agregar y quitar elementos es conocido como **LIFO** (del inglés "Last In First Out"), que se traduce como "ultimo en entrar, primero en salir". Visto como caso particular de una lista enlazada, el agregar (o "apilar") un elemento en una pila equivale a agregar un elemento únicamente al final de la lista, y el quitar (o "desapilar") elemento es equivalente a quitar solamente el último elemento de la lista.

Por otra parte, el tipo de dato cola consiste también en una colección lineal de datos en el que solo se tiene permitido agregar (o "encolar") elementos al final de la cola, sin embargo, contrario a lo sucede en una pila, en una cola solo se tiene acceso al elemento que está al frente de la cola, es decir, al primer elemento insertado. De esta forma, si se desea sacar un (o "desencolar") elemento se tienen que remover el primer elemento de la cola. Este método para agregar y quitar elementos es conocido como **FIFO** (del inglés "First In First Out"), que se traduce como "primero en entrar, primero en salir".

3. Detalles de implementación

La implementación de estas tres estructuras de datos fue escrita en el lenguaje de programación C, siguiendo la especificación del lenguaje detallada en el estándar C99. Los archivos principales se encuentran dentro del directorio **src** ubicada en la raíz del repositorio.

Además se incluye un archivo **pruebas.c** en el que se encuentran diferentes pruebas automatizadas para detectar errores en la ejecución del programa o en la asignación, manejo y liberación de memoria dinámica. Para esto se utiliza **gcc** como compilador y **valgrind** como herramienta de análisis de memoria.

3.1. Inserción de elementos

Se proveen dos funciones para insertar elementos. La primera de estas funciones, lista_insertar, se encarga de insertar un elemento al final de la lista y lista_insertar_en_posicion se encarga de insertar un elemento en la posición especificada por el usuario. Para esta última función, en el caso de que el usuario quisiera agregar un elemento en una posición que no existe en la lista, el elemento se agregará al final de la misma.

3.2. Eliminación de elementos

Algún detalle de otra función.

4. Diagramas

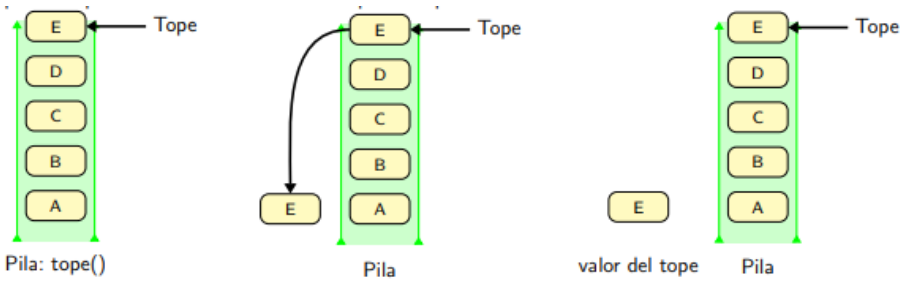

Figura 1: Ejemplo de un diagrama.

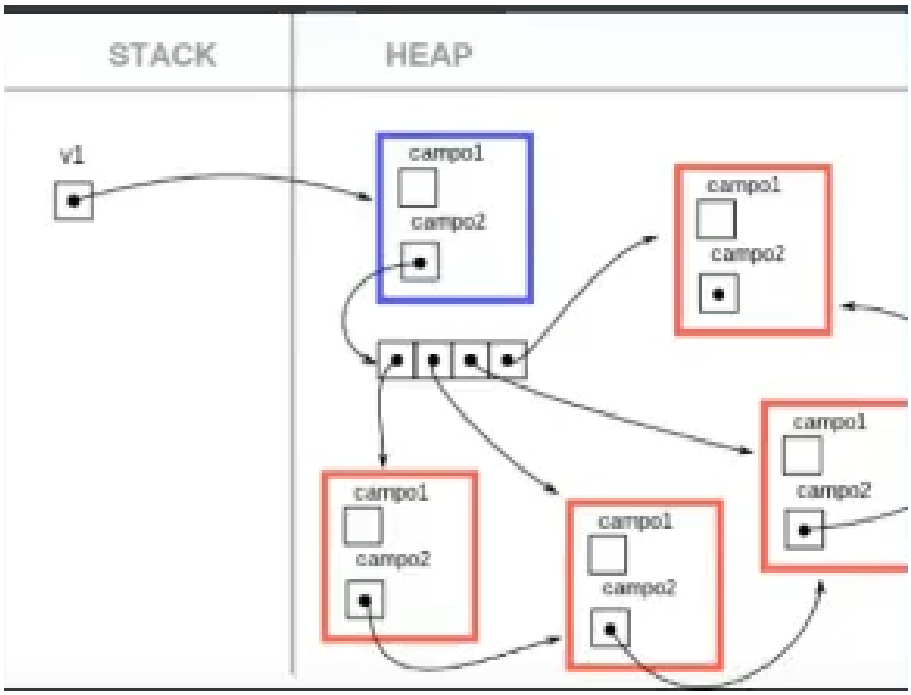

Figura 2: Otro ejemplo de un diagrama.