Estructuras de Datos y algoritmos 1

Práctico 3

Tema: Punteros - Listas y árboles

Escucho y olvido, veo y recuerdo, hago y aprendo.

Confucio

Objetivos

- Familiarizarse con el manejo de estructuras dinámicas lineales y arborescentes.
- Comprender el papel de los punteros, valores y referencias en el desarrollo de funciones y procedimientos, recursivos e iterativos, en C y C++

Listas

NOTAS:

I) En los siguientes ejercicios se debe buscar una solución iterativa y otra recursiva.

II) Considere en cada caso la posibilidad de implementar una función o un procedimiento (salvo que se indique lo contrario).

1) Resolver los ejercicios pendientes del teórico.

2) Dada una lista lineal encadenada de enteros, implementar las siguientes operaciones:
   a) Insertar un elemento al principio.
   b) Eliminar el elemento de una lista no vacía.
   c) Insertar un elemento en la posición K.
   d) Eliminar el elemento que ocupa la posición K
   e) Retornar el mínimo (máximo) de la lista (una referencia al nodo donde se encuentra).
   f) Eliminar el mínimo (máximo) de la misma.
   g) Eliminar todas las ocurrencias de un elemento e.

3) Hacer una función que imprima una lista en sentido directo e inverso recorriendo explícitamente la lista solo una vez.

4) Asumiendo que trabajamos con listas ordenadas:
   a) La inserción de un elemento.
   b) La baja de un elemento.

5) Considerar a la lista con un puntero al principio y otro al final. Implementar operaciones que permitan:
   a) Insertar un elemento al principio de la lista.
   b) Insertar un elemento al final de la lista.
   c) Eliminar el primer elemento de la lista.

d) Eliminar el último elemento de la lista.

6) Dadas dos listas encadenadas de enteros ordenadas, se pide:
   a) Hacer una función que retorne una tercera lista ordenada, intersección de las dos primeras.
   b) Hacer una función que retorne una tercera lista ordenada, unión de las dos primeras.
   c) Hacer una función que retorne una tercera lista ordenada, diferencia de las dos primeras.

7) Implementar las siguientes operaciones en una lista es doblemente encadenada:
   a) Insertar un elemento al principio de la lista.
   b) Eliminar el primero elemento de la lista.
   c) Insertar un elemento en la posición K.
   d) Eliminar el elemento de la posición K
   e) Insertar un elemento al final de la lista.
   f) Eliminar el último elemento de la lista.

   (*) Para esta dos operaciones finales considerar la posibilidad de tener un puntero al final de la lista. Estudiar la eficiencia que se obtiene.

8) Implementar el merge sort con listas

9) Hacer una función que reciba como parámetro de entrada una lista lineal encadenada de enteros y retorne una nueva lista de pares de enteros en la que cada nodo contiene el entero correspondiente de la lista original y la distancia del mismo al máximo valor almacenado en dicha lista.

Por ejemplo:

Si se recibe la lista: **{3, 8, 7, 2, 10, 15, 0, 12, 4}**

Se debe retornar una nueva lista de pares de enteros que contenga:

{(3,-12), (8, -7), (7,-8), (2, -13), (10, -5), (15, 0) , (0, -15), (12, -3), (4, -11)} ya que el máximo es **15**.

Arboles (binarios)

1) Implemente una función que retorne la cantidad de nodos que tiene un árbol binario recibido como parámetro.

2) Implemente una función que retorne la altura que tiene un árbol binario recibido como parámetro.

3) Implemente una función que retorne la cantidad de hojas que tiene un árbol binario recibido como parámetro.

4) Implemente una función que reciba dos árboles binarios y retorne *true* si ambos tienen la misma forma, *false* en caso contrario.

5) Implemente una función que reciba dos árboles binarios y retorne *true* si ambos son iguales, *false* en caso contrario.

6) Implemente una función que busque un elemento “e” en un árbol binario retornando un puntero al elemento encontrado o NULL en caso de no encontrarlo.

7) Implemente una función que busque un elemento “e” en un árbol binario de búsqueda retornando un puntero al elemento encontrado o NULL en caso de no encontrarlo.

8) Implemente una función que inserte un elemento “e” en un árbol binario de búsqueda.

9) Implemente una función que de baja un elemento “e” de un árbol binario de búsqueda.

*****AGREGAR EJERCICIOS DE PARCIAL Y EXAMEN

Arboles (generales)

1) Defina una estructura de datos que permita representar árboles generales.

2) Considere árboles generales no vacíos y sin elementos repetidos. Implemente las siguientes operaciones:

   a) ArbolHoja: Dado un elemento genera un árbol que sólo contiene dicho elemento (como una hoja).

   b) Insertar: Dados un árbol y dos elementos v y w, inserta a v como el primer hijo de w en el árbol (hijo más a la izquierda), siempre que w pertenezca al árbol y v no pertenezca al árbol. En caso contrario, la operación no tiene efecto.

   c) EsArbolHoja: Dado un árbol, retorna true si, y sólo si, el árbol es un árbol hoja (con un sólo elemento).

   d) Pertenece: Dados un árbol y un elemento, retorna true si y sólo si el elemento pertenece al árbol.

   e) Borrar: Dados un árbol y un elemento, elimina al elemento del árbol siempre que éste pertenezca al árbol, no sea la raíz del mismo y no tenga ningún hijo. En caso contrario, la operación no tiene efecto.

*** agregar ejercicios de árboles generales de parcial y examen