Carátula

Rios Matias.

Rychert Alan.

TDA Lista

Para implementar la lista se utilizan celdas simplemente enlazadas y un centinela, haciendo uso del concepto de posición indirecta, es decir, la posición se denota mediante un puntero a una celda que contiene un puntero a la celda que tiene el elemento deseado, y la lista es un puntero a una celda vacía que representa el centinela.

Dibujitos pongo después.

Operaciones del TDA:

void crear\_lista(tLista\* l):

Inicializa una lista vacía reservando y asignándole un lugar en la memoria dinámica. Como la lista está vacía, se inicializan los campos del struct en nulo.

tPosicion l\_primera(tLista l):

Recupera y retorna la primer posición de la lista. Si la lista está vacía, primera(l), ultima(l) y fin(l) son iguales. La primer posición es el resultado de castear a tPosicion la lista recibida por parámetro (el centinela).

tPosicion l\_ultima(tLista l):

Recupera y retorna la ultima posición de la lista. Si la lista está vacía, primera(l), ultima(l) y fin(l) son iguales. Para conseguir la ultima posición se recorren todas las posiciones de la lista hasta encontrarla.

void l\_destruir(tLista\* l,void (\*fEliminar)(tElemento)):

Destruye la lista l y elimina cada una de sus celdas y sus respectivos elementos. Los elementos son eliminados mediante la función fEliminar parametrizada, la cual debe ser implementada por el cliente de la lista.  
Al destruir la lista se liberan todos los espacios de la memoria dinámica que fueron reservados al momento de crear la lista o de insertar nuevos elementos.

void l\_insertar(tLista l, tPosicion p, tElemento e):

Inserta el elemento e en la lista l, en la posición p. Esto quiere decir que luego de insertar, el elemento contenido en la posición p ahora será e, y el elemento antes contenido en ella se encontrará en la siguiente de p (debido a que se trabaja con posición indirecta).

void l\_eliminar(tLista l, tPosicion p, void (\*fEliminar)(tElemento)):

Elimina la celda P de L, liberando el espacio en la memoria dinámica que había sido reservado al momento de insertar o crear la lista. El elemento almacenado en la posición P es eliminado mediante la función fElimina parametrizada, que debe ser implementada por el cliente de la lista. Si P es fin(l), finaliza indicando LST\_POSICION\_INVALIDA.

tElemento l\_recuperar(tLista l, tPosicion p):

Recupera y retorna el elemento contenido en la posición P (es decir, el elemento que se encuentra en la celda siguiente de la celda apuntada por P, debido a que se trabaja con posición indirecta). Si P es fin(l), finaliza indicando LST\_POSICION\_INVALIDA.

tPosicion l\_siguiente(tLista l, tPosicion p):

Recupera y retorna la posición siguiente a P en la lista, es decir, la celda siguiente a la celda apuntada por P. Si P es fin(l), finaliza indicando LST\_POSICION\_INVALIDA.

tPosicion l\_anterior(tLista l, tPosicion p):

Recupera y retorna la posición anterior a P en la lista, recorriendo la lista hasta encontrar la posición cuya siguiente posición es P. Si P es igual a l\_primera(l), finaliza indicando LST\_NO\_EXISTE\_ANTERIOR.

tPosicion l\_fin(tLista l):

Recupera y retorna la posición fin de l. Si la lista está vacía, l\_primera(l), l\_ultima(l) y l\_fin(l) son iguales. Para conseguir la posición fin, se recorre toda lista hasta encontrar una posición cuya siguiente posición sea nula.

int l\_longitud(tLista l):

Retorna la longitud actual de la lista, la cual es computada recorriendo toda la lista y aumentando un Contador por cada posición que la compone.

TDAArbol

Para la implementación del TDAArbol se utilizan “nodos” los cuales tienen un puntero hacia su elemento, su padre y a una lista que contiene todos sus hijos, además cada árbol contiene un puntero hacia su raíz.

Operaciones del TDAArbol:

void crear\_arbol(tArbol \* a);