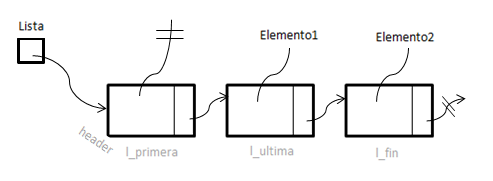
|  |
| --- |
| oRGANIZACIÓN DE COMPUTADORAS |
| TA-TE-TI |
| Proyecto N◦1 Programación en lenguaje C |
| **Segundo Cuatrimestre de 2019** |
|  |
|  |

|  |
| --- |
| Olaciregui Santiago – 121485 | Mayer Agustín – 121256 |

# TDA Lista

La lista está implementada mediante una estructura simplemente enlazada con celda centinela, utilizando el concepto de posición indirecta. En esta representación cada celda mantiene referencia a la celda siguiente y una referencia a un elemento genérico.  


**void crear\_lista(tLista \* l);** Inicializa una lista vacía. Una referencia a la lista creada es referenciada en \*l.

**void l\_insertar(tLista l, tPosicion p, tElemento e);** Inserta el elemento E, en la posición P, en L. Con L= A,B,C,D y la posición P direccionando C, luego: L' = A,B,E,C,D

**void l\_eliminar(tLista l, tPosicion p, void (\*fEliminar)(tElemento));**  
 Elimina la celda P de L. El elemento almacenado en la posición P es eliminado mediante la función fEliminar parametrizada. Si P es fin(L), finaliza indicando LST\_POSICION\_INVALIDA.

**void l\_destruir(tLista \* l, void (\*fEliminar)(tElemento));** Destruye la lista L, eliminando cada una de sus celdas. Los elementos almacenados en las celdas son eliminados mediante la función fEliminar parametrizada.

**tElemento l\_recuperar(tLista l, tPosicion p);**   
 Recupera y retorna el elemento en la posición P. Si P es fin(L), finaliza indicando LST\_POSICION\_INVALIDA.

**tPosicion l\_primera(tLista l);**  
 Recupera y retorna la primera posición de L. Si L es vacía, primera(L) = ultima(L) = fin(L).

**tPosicion l\_siguiente(tLista l, tPosicion p);**  
 Recupera y retorna la posición siguiente a P en L. Si P es fin(L), finaliza indicando LST\_NO\_EXISTE\_SIGUIENTE.

**tPosicion l\_anterior(tLista l, tPosicion p);**  
 Recupera y retorna la posición anterior a P en L. Si P es primera(L), finaliza indicando LST\_NO\_EXISTE\_ANTERIOR.

**tPosicion l\_ultima(tLista l);**  
 Recupera y retorna la última posición de L. Si L es vacía, primera(L) = ultima(L) = fin(L).

**tPosicion l\_fin(tLista l);**   
 Recupera y retorna la posición fin de L. Si L es vacía, primera(L) = ultima(L) = fin(L).

**int l\_longitud(tLista l);**  
 Retorna la longitud actual de la lista.

# TDA Árbol

El árbol está implementado mediante una estructura de nodos enlazados. En esta representación cada nodo mantiene referencia a otro nodo considerado padre del mismo dentro del árbol, una lista de nodos que representa los nodos hijos del mismo, y una referencia a un elemento genérico que representa el rótulo de dicho nodo.

# Juego Ta-Te-Ti

Se implementó una aplicación de consola que permite visualizar la dinámica de una partida del juego Ta-Te-Ti. El programa inicia solicitando los nombres de los dos jugadores, el modo de juego que desea el usuario, y la indicación de qué jugador comienza la partida. Luego, inicia una nueva partida de forma tal que, en cada turno, se imprime por consola el estado actual del tablero, las indicaciones de quién debe jugar, se consulta el movimiento a realizar por los usuarios y se modifica la partida a partir de la recopilación de estos datos.   
Se permiten dos modos de juego: Usuario vs. Usuario y Usuario vs. Agente IA. A la hora de indicar qué jugador comienza la partida se permiten tres opciones: jugador 1, jugador 2 o jugador al azar. Bajo esta última alternativa, el programa (de forma aleatoria) selecciona qué jugador comienza la partida.