**TDA Lista doblemente Enlazada**

| **OPERACIÓN** | **DEFINICIÓN** | **EXPLICACIÓN GRÁFICA DE LA OPERACIÓN** |
| --- | --- | --- |
| **crearNodo( )** | Es una función la cual solicita por consola ingresar un numero entero y retorna en nodo con sus apuntadores y la información |  |
| **crearNodo(int dato)** | Es una función que es invocada por el método crearNodo( ) y recibe como parámetro un entero, esta función lo que hace es crear el nodo con su respectiva información y sus dos apuntadores retornando así el nodo para el método crearNodo( ) |  |
| **void borrarNodo(nodoLDE \*nodoBorrar)** | Es un procedimiento que recibe como parámetro un nodo, cuando recibe este nodo lo elimina y se libera memoria. |  |
| **TDAListaDoble \*BorrarElemento(TDAListaDoble \*c, nodoLDE \*nodBorrar)** | Es una función que recibe una lista de elementos doblemente enlazados y un nodo que desea eliminar, esta función evalúa donde se encuentra el nodo con el fin de eliminarlo y enlazar nuevamente la lista, retornando la lista enlazada actualizada sin el elemento que se indicó eliminar y se libera memoria. |  |
| **void InsertarPorCabeza(TDAListaDoble \*c, nodoLDE \*nodo)** | Es procedimiento que recibe una lista doblemente enlazada y un nodo, después de recibir los parámetros inserta el nodo por el inicio de la lista enlazándolo a la lista. |  |
| **void InsertarPorFinal(TDAListaDoble \*c, nodoLDE \*nodo)** | Es procedimiento que recibe una lista doblemente enlazada y un nodo, después de recibir los parámetros inserta el nodo por el final de la lista enlazándolo a la lista. |  |
| **void InsertarEntre(TDAListaDoble \*c, nodoLDE \*nuevo, int elemento)** | Es un procedimiento que recibe una lista doblemente enlazada, un nodo y la información del nodo que se encuentra en la lista, después de recibir los parámetros busca el nodo que contenga la información recibida si lo encuentra evalúa si lo inserta por el inicio, por el final o por el medio, en caso de que no encuentre el nodo lanza un mensaje de error. |  |
| **if (c->inicio==nodBorrar && c->fin==nodBorrar ){**  **c=Inicilizar();**  **}** | Si la lista está apuntando por su inicio y su final al mismo nodo, inicializa la lista creando una nueva lista donde su inicio y su fin apuntan a null, porque en la lista solo hay un elemento y al quedar eliminado queda vacía la lista y deben apuntar a null su inicio y su fin. |  |
| **else {**  **if(c->inicio!=nodBorrar && c->fin!=nodBorrar) {**  **nodBorrar->ant->sig = nodBorrar->sig;**  **nodBorrar->sig->ant = nodBorrar->ant;** | Después del else, el primer if evalúa si la lista en su inicio y su fin no apuntan al nodo que se necesita borrar, borran el nodo de la mitad (como si en la lista solo existen 3 nodos), por lo que faltaría aplicar la función nodoLDE \*buscarNodo(int elemento, TDAListaDoble \*c), para el caso en que haya más de tres nodos en la lista. |  |
| **} else if(c->inicio ==nodBorrar) {**  **c->inicio->ant=NULL;**  **c->inicio = c->inicio->sig;**  **nodBorrar->sig->ant = NULL;** | El segundo if después del else evalúa si el nodo se encuentra al inicio de la lista y lo elimina, enlazando nuevamente la lista con el siguiente nodo. |  |
| **else if(c->fin ==nodBorrar) {**  **c->fin->ant->sig=NULL;**  **c->fin=c->fin->ant;**  **nodBorrar->ant->sig = NULL;**  **}** | El tercer if después del else evalúa si el nodo se encuentra al final de la lista y lo elimina, enlazando nuevamente la lista con el anterior nodo. |  |
| **void imprimirInicio(TDAListaDoble \*c)** | Es un procedimiento que recibe como parámetro una lista y muestra la información del primer elemento de la lista |  |
| **void imprimirFin(TDAListaDoble \*c)** | Es un procedimiento que recibe como parámetro una lista y muestra la información del último elemento de la lista |  |
| **bool esVacia(TDAListaDoble \*c)** | Es una función que recibe como parámetro una lista y retorna true si está vacía o false si tiene nodos. |  |
| **nodoLDE \*buscarNodo(int elemento, TDAListaDoble \*c)** | Es una función que recibe como parámetros la información del nodo y una lista, con el propósito de primero evaluar si la lista está vacía, sino lo está recorre la lista hasta encontrar el nodo buscado y lo retorna. |  |

**Modificar Elemento**

**nodoLDE \*modificarElemento(nodoLDE \*nodoModificar){**

**int nuevoValor;**

**nodoLDE \*nodo;**

**nodo->info = nodoModificar->info;**

**cout<<"Ingrese el nuevo valor";**

**cin>>nuevoValor;**

**nodo->info = nuevoValor;**

**nodo->ant = nodoModificar->ant;**

**nodo->sig = nodoModificar->sig;**

**return nodo;**

**}**