```
#include <iostream>

#include "Nodo.h"

//----------------------------------------------Funciones----------------------------------------------

void insertarAlInicio(Nodo* &lista, ELEMENTO dato) {
    Nodo* n = new Nodo();

    n->dato = dato;
    n->siguiente = lista;
    lista = n;
}

void insertarAlFinal(Nodo*& lista, ELEMENTO dato) {
    Nodo* aux;
    Nodo* n = new Nodo();

    n->dato = dato;
    n->siguiente = NULL;

    if (lista == NULL) {
        lista = n;
    } else {
        aux = lista;
        while (aux->siguiente != NULL) {
            aux = aux->siguiente;
        }
        aux->siguiente = n;
    }
```

```
}

void insertarElementoEnPosicion(Nodo* &lista, int iPosicion, ELEMENTO dato) {
   bool bOperacion_Realizada = false;

   if (iPosicion == 0) {
      insertarAlInicio(lista, dato);
      bOperacion_Realizada = true;
   } else {
      Nodo* aux = lista;
      Nodo* n = new Nodo();

      n->dato = dato;

      for (int i = 0; aux != NULL; i++) {

         if (i == (iPosicion - 1)) {
            n->siguiente = aux->siguiente;
            aux->siguiente = n;
            bOperacion_Realizada = true;
         }

         aux = aux->siguiente;
      }

   }

   if (bOperacion_Realizada == false)
      std::cout << "Error... Posicion no encontrada..!" << std::endl;
}
```

```
int buscarElemento(Nodo* lista, ELEMENTO dato) {
    Nodo* aux = lista;
    int iContador = 0, iPosicion = NO_ENCONTRADO;
    bool bEncontrado = false;

    while (aux != NULL && !bEncontrado) {

        if (aux->dato == dato) {
            iPosicion = iContador;
            bEncontrado = true;
        }

        aux = aux->siguiente;
        iContador++;

    }

    return iPosicion;
}

void obtenerElemento(Nodo* lista, int iPosicion, ELEMENTO &dato) {
    Nodo* actual = lista;
    int iContador = 0;
    bool bEncontrado = false;

    if (iPosicion != NO_ENCONTRADO) {
        if (!listaVacia(lista)) {
            while (actual != NULL && !bEncontrado) {
```

```
            iContador++;
            if (iContador == iPosicion + 1) {
               dato = actual->dato;
               bEncontrado = true;
            }

            actual = actual->siguiente;

         }
      } else
         std::cout << "\nNo hay datos...\n";
   }

   if (!bEncontrado) {
      dato = NULL;
   }

}

void quitarElementoDelInicio(Nodo* &lista, ELEMENTO& dato) {

   Nodo* aux = lista;
   dato = aux->dato;
   lista = aux->siguiente;
   delete aux;

}

void quitarElementoDePosicion(Nodo* &lista, int iPosicion, ELEMENTO &dato) {
```

```
if (iPosicion != NO_ENCONTRADO) {

    if (!listaVacia(lista)) {

        if (iPosicion == 0) {
            quitarElementoDelInicio(lista, dato);
        } else {
            Nodo* anterior = NULL;
            Nodo* actual = lista;
            int iContador = 0;
            bool bEncontrado = false;

            while (actual != NULL && !bEncontrado) {

                iContador++;
                if (iContador == iPosicion + 1) {
                    dato = actual->dato;
                    bEncontrado = true;
                }

                if (bEncontrado) {
                    anterior->siguiente = actual->siguiente;
                    delete actual;
                    actual = NULL;
                } else {
                    anterior = actual;
                    actual = actual->siguiente;
                }
            }
```

```
            if (!bEncontrado) {
                std::cout << "No existe la posicion ingresa..." << std::endl;
                dato = NULL;
            }

        }

    }//cierre if lista vacia
    else
        std::cout << "\nLista vacia...\n";

  } //cierre if inicial

}

void vaciarLista(Nodo* &lista) {
  if (!listaVacia(lista)) {
    ELEMENTO aux;

    while (!listaVacia(lista))
        quitarElementoDePosicion(lista, 0, aux);

    delete lista;
    lista = NULL;
  }
}

bool listaVacia(Nodo* lista) {
  return (lista == NULL) ? true : false;
}
```