IMPLEMENTACIÓN DE UNA LISTA CON SUS PRINCIPALES METODOS

CLASE NODO

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ListasEnlazadas
{
   internal class CNodo
   {
      public int dato;
      public CNodo siguiente;

      public CNodo(int valor)
      {
         dato = valor;
         siguiente = null;
      }
   }
}
```

CLASE LISTA

```
using System;

namespace ListasEnlazadas
{
   internal class ClistaEnlazada
   {
      private CNodo cabeza;

      public ClistaEnlazada()
      {
         cabeza = null;
      }
      // Método para insertar al inicio
      public void insertarInicio(int valor)
      {
         CNodo nuevoNodo = new CNodo(valor);
         nuevoNodo.siguiente = cabeza;
         cabeza = nuevoNodo;
         Console.WriteLine($"Insertado {valor} al inicio.");
      }
```

```
// Método para insertar al final
public void insertarFinal(int valor)
{
    CNodo nuevoNodo = new CNodo(valor);
    if (cabeza == null)
    {
        cabeza = nuevoNodo;
    }
    else
    {
        CNodo actual = cabeza;
        while (actual.siguiente != null)
        {
            actual = actual.siguiente;
        }
        actual.siguiente = nuevoNodo;
    }
    Console.WriteLine($"Insertado {valor} al final.");
}
// Método para eliminar el primer nodo
public void eliminarInicio()
{
    if (cabeza != null)
    {
        Console.WriteLine($"Eliminando {cabeza.dato} del inicio.");
        cabeza = cabeza.siguiente;
    }
    else
    {
        Console.WriteLine("La lista está vacía.");
    }
}
// Método para eliminar el último nodo
public void eliminarFinal()
{
    if (cabeza == null)
    {
        Console.WriteLine("La lista está vacía.");
        return;
    }

    if (cabeza.siguiente == null)
    {
        Console.WriteLine($"Eliminando {cabeza.dato} del final.");
        cabeza = null;
        return;
    }
```

```
    CNodo actual = cabeza;
    while (actual.siguiente.siguiente != null)
    {
        actual = actual.siguiente;
    }
    Console.WriteLine($"Eliminando {actual.siguiente.dato} del final.");
    actual.siguiente = null;
}
// Método para eliminar un nodo específico por su valor
public void eliminarNodo(int valor)
{
    if (cabeza == null)
    {
        Console.WriteLine("La lista está vacía.");
        return;
    }

    if (cabeza.dato == valor)
    {
        Console.WriteLine($"Eliminando {valor}.");
        cabeza = cabeza.siguiente;
        return;
    }

    CNodo actual = cabeza;
    while (actual.siguiente != null && actual.siguiente.dato != alor)
    {
        actual = actual.siguiente;
    }

    if (actual.siguiente != null)
    {
        Console.WriteLine($"Eliminando {valor}.");
        actual.siguiente = actual.siguiente.siguiente;
    }
    else
    {
        Console.WriteLine($"El valor {valor} no se encontró en la lista.");
    }
}
// Método para buscar un nodo
public bool buscar(int valor)
{
    CNodo actual = cabeza;
    while (actual != null)
    {
```

```
            if (actual.dato == valor)
            {
                Console.WriteLine($"Valor {valor} encontrado en la lista.");
                return true;
            }
            actual = actual.siguiente;
        }
        Console.WriteLine($"Valor {valor} no encontrado en la lista.");
        return false;
    }
    // Método para invertir la lista
    public void invertir()
    {
        CNodo anterior = null;
        CNodo actual = cabeza;
        CNodo siguiente = null;

        while (actual != null)
        {
            siguiente = actual.siguiente;
            actual.siguiente = anterior;
            anterior = actual;
            actual = siguiente;
        }
        cabeza = anterior;
        Console.WriteLine("La lista ha sido invertida.");
    }
    // Método para obtener el tamaño de la lista
    public int obtenerTamaño()
    {
        int tamaño = 0;
        CNodo actual = cabeza;
        while (actual != null)
        {
            tamaño++;
            actual = actual.siguiente;
        }
        Console.WriteLine($"El tamaño de la lista es: {tamaño}");
        return tamaño;
    }
    // Método para imprimir la lista
    public void imprimirLista()
    {
        if (cabeza == null)
        {
            Console.WriteLine("La lista está vacía.");
            return;
```

```
        }

        CNodo actual = cabeza;
        Console.WriteLine("Elementos de la lista:");
        while (actual != null)
        {
            Console.WriteLine(actual.dato);
            actual = actual.siguiente;
        }
    }
  }
}
```

IMPLEMENTACION DEL PROGRAMA PRINCIPAL

```
using System;

namespace ListasEnlazadas
{
  internal class Program
  {
    static void Main(string[] args)
    {
      ClistaEnlazada lista = new ClistaEnlazada();
      int opcion = -1;

      while (opcion != 0)
      {
        Console.WriteLine("\n--- Menú de Opciones ---");
        Console.WriteLine("1. Insertar al inicio");
        Console.WriteLine("2. Insertar al final");
        Console.WriteLine("3. Eliminar el primer nodo");
        Console.WriteLine("4. Eliminar el último nodo");
        Console.WriteLine("5. Eliminar un nodo por valor");
        Console.WriteLine("6. Buscar un valor en la lista");
        Console.WriteLine("7. Invertir la lista");
        Console.WriteLine("8. Mostrar tamaño de la lista");
        Console.WriteLine("9. Imprimir la lista");
        Console.WriteLine("0. Salir");
        Console.Write("Seleccione una opción: ");

        opcion = int.Parse(Console.ReadLine());
        int valor;

        switch (opcion)
        {
          case 1:
            Console.Write("Ingrese el valor a insertar al inicio: ");
```

```
            valor = int.Parse(Console.ReadLine());
            lista.insertarInicio(valor);
            break;

        case 2:
            Console.Write("Ingrese el valor a insertar al final: ");
            valor = int.Parse(Console.ReadLine());
            lista.insertarFinal(valor);
            break;

        case 3:
            lista.eliminarInicio();
            break;

        case 4:
            lista.eliminarFinal();
            break;

        case 5:
            Console.Write("Ingrese el valor a eliminar: ");
            valor = int.Parse(Console.ReadLine());
            lista.eliminarNodo(valor);
            break;

        case 6:
            Console.Write("Ingrese el valor a buscar: ");
            valor = int.Parse(Console.ReadLine());
            lista.buscar(valor);
            break;

        case 7:
            lista.invertir();
            break;

        case 8:
            lista.obtenerTamaño();
            break;

        case 9:
            lista.imprimirLista();
            break;

        case 0:
            Console.WriteLine("Saliendo...");
            break;

        default:
```

```
                Console.WriteLine("Opción no válida.");
                break;
            }
        }
    }
  }
}
```

EJERCICIOS PROPUESTOS:

Ejercicio 1: Ordenar e invertir

- Crea una lista con los siguientes valores: 8, 2, 5, 3, 10.
- Escribe un método adicional que ordene la lista en orden ascendente.
- Invierte la lista después de haberla ordenado y verifica si el orden es correcto imprimiendo los elementos.

Ejercicio 2: Duplicados

- Escribe un método que permita eliminar nodos duplicados de la lista.
- Crea una lista con valores duplicados y luego llama a este método para eliminar las repeticiones.

Ejercicio 3: Rotación de la lista

- Implementa un método llamado rotar (int n) que mueva los primeros n elementos de la lista al final.
- Prueba el método con una lista de 7 elementos y rota 3 posiciones.