ÇİFT YÖNLÜ LİNKED LİST

ESMA BALIKÇI

24080410019

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Security.Cryptography.X509Certificates;
using System.Text;
using System.Threading.Tasks;

namespace ciftyonlubagliliste
{
    internal class Program
    {
        static void Main(string[] args)
        {

        }
        //Düğüm tanımlama
        public class Node
        {
            public int Veri;
            public Node Next;
            public Node Prev;
            public Node(int veri)
            {
                this.Veri = veri;
                this.Prev = null;
                this.Next = null;
            }
        }
        public class BagliListe
        {
            public Node head;
            public Node tail;

            public BagliListe()
            {
                this.head = null;
                this.tail = null;
            }
            // Başa ekleme
            public void BasaEkle(int veri)
            {
                Node newNode = new Node(veri);
                if (head == null)
                {
                    head = newNode;
                    tail = newNode;
                }
                else
                {
                    newNode.Next = head;
                    head.Prev = newNode;
                    head = newNode;
                    Console.WriteLine($"{veri} listenin başına eklendi.");
```

```
            }
        }
        // Sona Ekleme
        public void SonaEkle(int veri)
        {
            Node newNode = new Node(veri);
            if (head == null)
            {
                head = newNode;
                tail = newNode;
            }
            else
            {
                tail.Next = newNode;
                newNode.Prev = tail;
                tail = newNode;
                Console.WriteLine($"{veri} listenin sonuna eklendi.");
            }
        }
        //Araya herhangi bir veriden sonra ekleme
        public void SonrasınaEkleme(int hedefVeri, int eklenecekVeri)
        {
            Node newNode = new Node(eklenecekVeri);
            if (head == null)
            {
                Console.WriteLine("Liste boş araya ekleme yapılamaz.");
                return;
            }
            Node current = head;
            while (current != null && current.Veri != hedefVeri)
            {
                current = current.Next;
            }
            if (current == null)
            {
                Console.WriteLine("Aranan veri bulunamadı.");
                return;
            }
            if (current == tail)
            {
                SonaEkle(eklenecekVeri);
                return;
            }
            newNode.Next = current.Next;
            newNode.Prev = current;
            current.Next = newNode;
            current.Next.Prev = newNode;
            Console.WriteLine($"{eklenecekVeri} listeye {hedefVeri}
verisinden sonra eklendi");
        }
        //Araya herhangi bir veriden önce ekleme
        public void BasinaEkleme(int hedefVeri, int eklenecekVeri)
        {
            Node newNode = new Node(eklenecekVeri);
            if (head == null)
            {
                Console.WriteLine("Liste boş araya ekleme yapılamaz.");
                return;
            }
            Node current = head;
            while (current != null && current.Veri != hedefVeri)
            {
                current = current.Next;
```

```
                }
                if (current == null)
                {
                    Console.WriteLine("Aranan veri bulunamadı.");
                    return;
                }
                if (current == head)
                {
                    BasaEkle(eklenecekVeri);
                    return;
                }
                newNode.Next = current;
                newNode.Prev = current.Prev;
                current.Prev.Next = newNode;
                current.Prev = newNode;
                Console.WriteLine($"{eklenecekVeri} listeye {hedefVeri}verisinden
önce eklendi.");
            }
            //Baştan Silme
            public void BastanSilme()
            {
                if (head == null)
                {
                    Console.WriteLine("Liste boş silinecek eleman yok.");
                    return;
                }
                if (head.Next == null)
                {
                    head = null;
                    tail = null;
                    Console.WriteLine("Listede tek eleman vardı silindi. Liste
artık boş.");
                }
                else
                {
                    head = head.Next;
                    head.Prev = null;
                    Console.WriteLine("Baştaki eleman silindi.");
                }
            }
            // Sondan Silme
            public void SondanSilme()
            {
                if (head == null)
                {
                    Console.WriteLine("Liste boş silinecek eleman yok.");
                    return;
                }
                if (head == tail)
                {
                    head = null;
                    tail = null;
                    Console.WriteLine("Listede tek eleman vardı silindi. Liste
artık boş.");
                }
                else
                {
                    tail = tail.Prev;
                    tail.Next = null;
                    Console.WriteLine("Listenin son elemanı silindi.");
                }
            }
            // Aradan arayarak silme
```

```
        public void ArayarakSilme(int silinecekVeri)
        {
            if (head == null)
            {
                Console.WriteLine("Liste boş silinecek eleman yok.");
                return;
            }
            Node silinecekDugum = head;
            while (silinecekDugum != null && silinecekDugum.Veri !=
silinecekVeri)
            {
                silinecekDugum = silinecekDugum.Next;
            }
            if (silinecekDugum == null)
            {
                Console.WriteLine($"{silinecekVeri} listede bulunamadı.");
                return;
            }
            if (silinecekDugum == head)
            {
                BastanSilme();
                return;
            }
            if (silinecekDugum == tail)
            {
                SondanSilme();
                return;
            }
            silinecekDugum.Prev.Next = silinecekDugum.Next;
            silinecekDugum.Next.Prev = silinecekDugum.Prev;
        }
        // Arama
        public void Arama(int veri)
        {
            Node current = head;
            while (current != null)
            {
                if (current.Veri == veri)
                {
                    Console.WriteLine($"{veri} listenin başında bulundu.");
                    return;
                }
                current = current.Next;
            }
            return;
        }
        // Listeleme
        public void Listeleme()
        {
            if (head == null)
            {
                Console.WriteLine("Liste boş.");
                return;
            }
            Console.Write("Liste: ");
            Node current = head;
            while (current != null)
            {
                Console.WriteLine(current.Veri + "-->");
                current = current.Next;
            }
        }
        //Tümünü Silme
```

```
        public void TumunuSilme()
        {
            head = null;
            tail = null;
            Console.WriteLine("Liste silindi.");
        }
        public int Boyut
        {
            get
            {
                int sayac = 0;
                Node current = head;
                while (current != null)
                {
                    sayac++;
                    current = current.Next;
                }
                return sayac;
            }
        }
        // Tüm linked listi bir diziye atma
        public int[] DiziyeCevir()
        {
            int boyut = this.Boyut;

            if (boyut == 0)
            {
                return new int[0];
            }
            int[] dizi = new int[boyut];

            Node current = head;
            int index = 0;
            while (current != null)
            {
                dizi[index] = current.Veri;
                current = current.Next;
                index++;
            }

            return dizi;
        }
    }
}
```