BAĞLAÇLI LİSTELER
LINKED LISTS

Liste

Günlük yaşamda **listeler** pek çok yerde kullanılmaktadır: **Alışveriş listeleri**, **adres listeleri**, **davetli listeleri** gibi. Bilgisayar programlarında da listeler yararlı ve yaygın olarak kullanılan veri yapılarındandırlar.

Programlama açısından liste, aralarında doğrusal ilişki olan veriler topluluğu olarak görülebilir. Yığıt ve kuyrukların genişletilmesi yani üzerlerindeki sınırlamaların kaldırılması ile liste yapısına ulaşılır. Veri yapılarında değişik biçimlerde listeler kullanılmakta ve üzerlerinde değişik işlemler yapılmaktadır.

Listeler Üzerindeki Bazı İşlemler ve Tanımları

1. EmptyList(List) : returns Boolean
Listenin boş olup olmadığını belirleyen fonksiyon.
2. FullList(List) : returns Boolean
Listenin dolu olup olmadığını belirleyen fonksiyon.
3. LengthList(List) : returns integer
Listedeki eleman sayısını bulan fonksiyon.
4. InsertElement(List, NewElement)
Listeye yeni bir eleman ekleyen fonksiyon.
5. DeleteElement(List, Element)
Listeden bir elemanı arayarak çıkartan fonksiyon.
6. DestroyList(List)
Listedeki tüm elemanları silerek boş liste bırakan fonksiyon.
7. GetNextItem(List, Element)
Etkin elemandan bir sonrakini döndüren fonksiyon
8. RetrieveElement(List, Element, Found)
Elemanın listede olup olmadığını bulan ve döndüren fonksiyon

Bağlaçlı (Bağlı) Listeler

Kendi tipindeki bir yapıyı gösteren bir işaretçi üyesine sahip yapılara self-referential structures adı verilir.

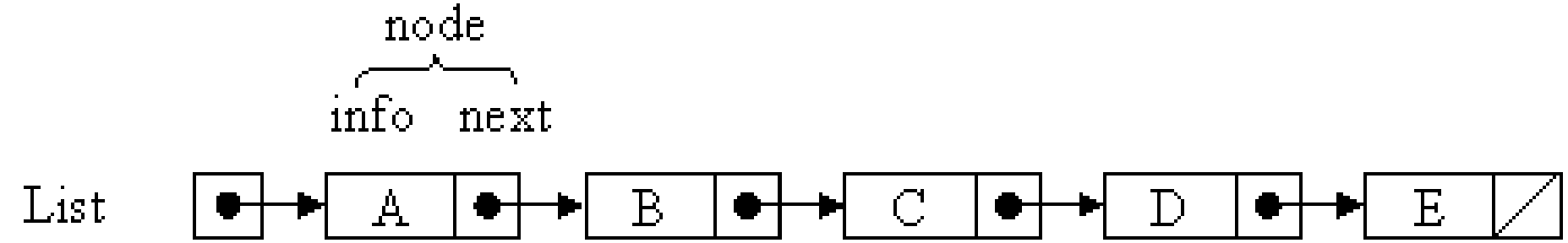
Örnek olarak :

```
struct node {
        char info;
        struct node *next; };
```

yapısı, info adlı karakter tipli bilgi elemanının yanında, bir düğüm yapısında bir bellek bölgesine işaret eden next işaretçisine sahiptir. Bu tür yapıların arka arkaya birbirine bağlanması mantığı listelerde, yığıtlarda, kuyruklarda ve ağaçlarda oldukça yararlıdır.

Doğrusal Bağlaçlı Liste

Listedeki her düğümde bir sonraki düğümün adresinin tutulduğu veri yapısı (doğrusal) bağlı liste olarak adlandırılır. Listenin her bir elemanına düğüm (node) adı verilir. Düğümler, bilgi ve bağ (adres) sahalarından oluşmaktadırlar. Bağ sahalarında işaretçiler kullanılmaktadır. Listenin ilk elemanına dışarıdan bir işaretçi (list) ile erişilmektedir. Diğer düğümlere de bağlar yardımı ile ulaşılabilmektedir. Son düğümün sonraki adres (next) sahası NULL değerini içerir. NULL bağı, liste sonunu belirtir. Elemanı olmayan liste boş liste olarak adlandırılır. Herhangi bir boyuta dinamik olarak genişletilip daraltılabilen yığıt ve kuyrukların gerçekleştirimi bağlı listeler üzerinde yapılmaktadır.



Yığıt ve Kuyruk Gerçekleştiriminde Bağlaçlı Listeler

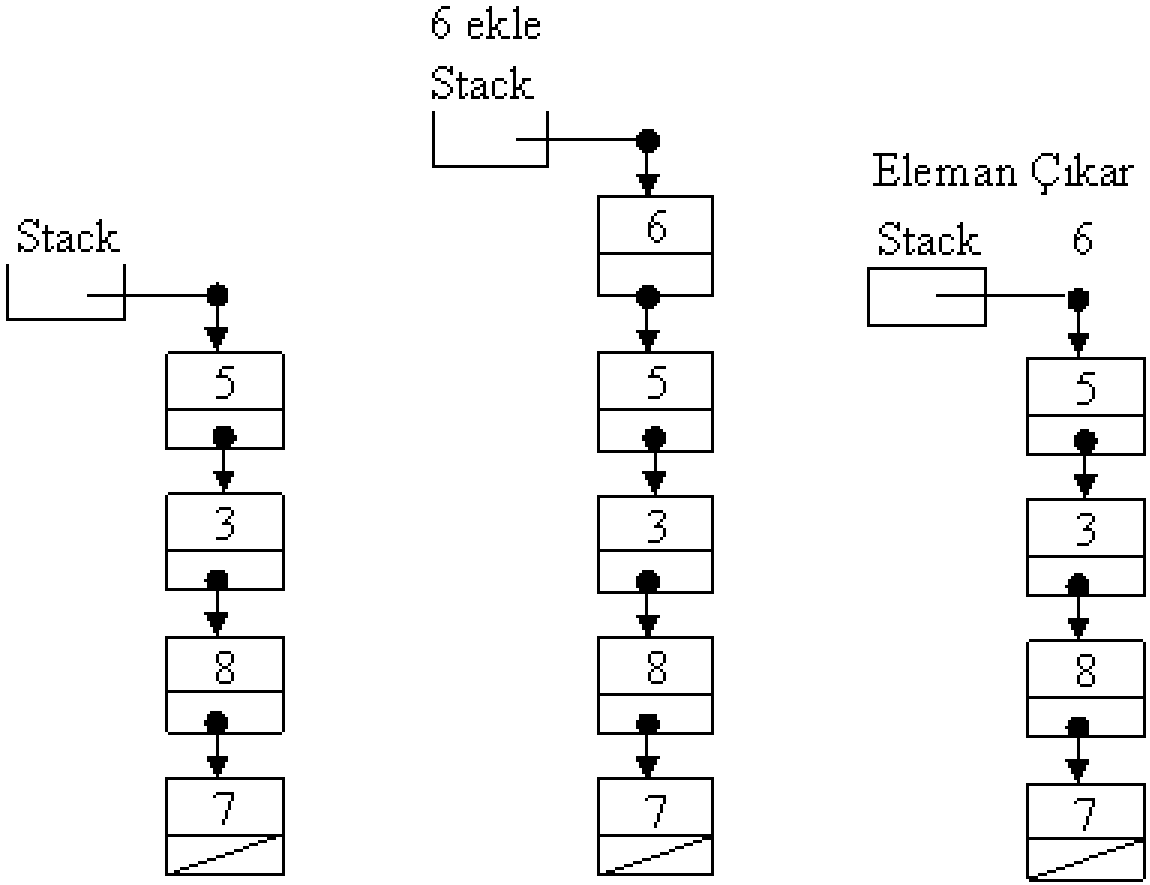
Yığıtlarda ve kuyrukların gerçekleştiriminde sıralı bellek kullanımının (dizi) en büyük dezavantajı, hiç kullanılmasa veya az kullanılsa bile sabit miktardaki belleğin bu yapılara ayrılmış olarak tutulmasıdır. Ayrıca sabit bellek miktarı aşıldığında da taşma oluşması ve eleman ekleme işleminin yapılamamasıdır. Bağlı listeler üzerinde gerçekleştirildiklerinde ise bu problemler ortadan kalkmaktadır. Bellekten sadece gerektiği kadar yer ayrılmakta ve bellek boyutu bitene kadar bu yapılara ekleme işlemi yapılabilmektedir.

Bağlaçlı Liste Veri Yapısı ve Avantajı

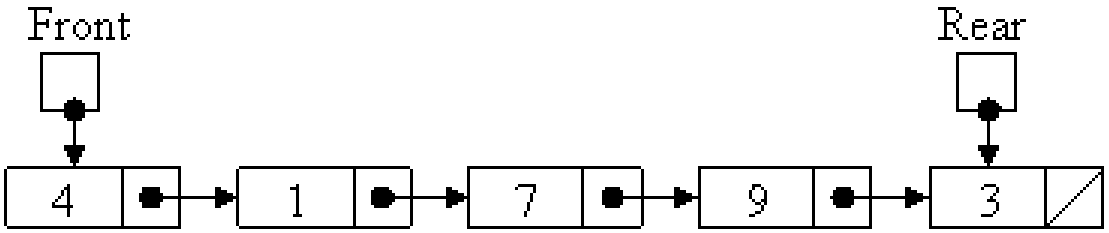
Bağlı listeler, başka veri yapılarının gerçekleştiriminde kullanılabildikleri gibi kendileri de veri yapısıdırlar. Bağlı listelerde elemanların eklenme ve çıkarılmasında bir sınırlama yoktur. Başa ve sona olduğu gibi araya da eleman eklenebilir; baştan ve sondan olduğu gibi ortadan da eleman çıkarılabilir. Bağlı liste dolaşılarak herhangi bir elemanına erişilebilir. Bir bağlı listenin n. elemanına erişmek için n tane işlem yapmak yani kendinden önceki (n-1) eleman üzerinden geçmek gerekmektedir. Elemanların bellekteki yerleri dizilerdeki gibi sıralı olmadığından elemanlar ve sıraları ile yerleştikleri bellek bölgeleri arasında bir ilişki yoktur.

Bağlı listelerin diziler üzerine avantajı, bir grup eleman arasına eleman eklemede ve bir grup eleman arasından eleman çıkarmada ortaya çıkar. Dizilerde bir eleman silerken arada boşluk kalmasını engellemek için ilerisindeki (sağındaki) tüm elemanları bir geriye (sola) kaydırmak gerekir. Eleman eklemede de yer açmak için konulacağı yerdeki ve ilerisindeki elemanları bir ileriye (sağa) kaydırmak gerekecektir. Kaç tane elemanın yer değiştireceği (birer kaydırılacağı) dizi boyutuna bağlı olarak ve eklenecek elemanın yerine bağlı olarak değişecektir. Bağlı listelerde ise eleman ekleme ve çıkarma için yapılan iş liste boyutundan bağımsızdır.

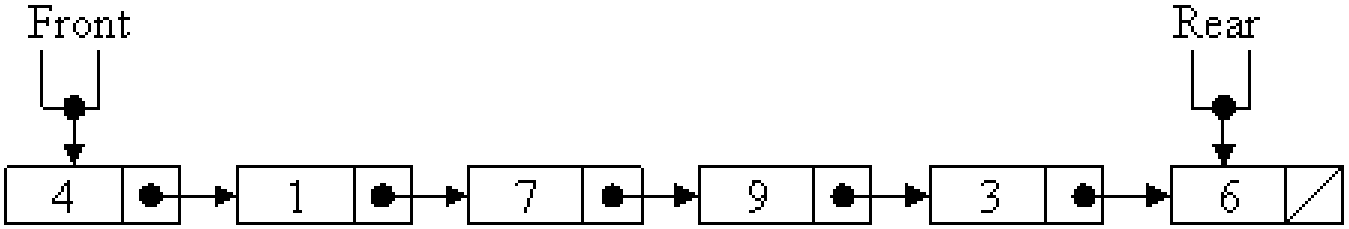
Yığıtın Bağlaçlı Liste Gösterimi Eleman Ekleme ve Çıkarma



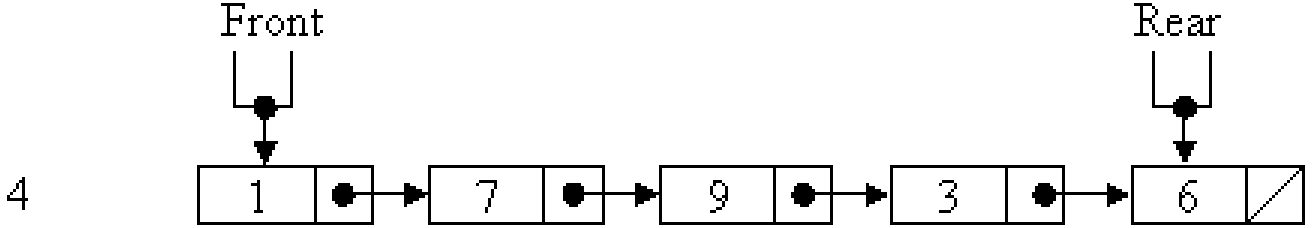
Kuyrukların Bağlaçlı Liste Gösterimi Eleman Ekleme ve Çıkarma



6 ekle :



Eleman Çıkar :



Öncelik Kuyruklarının Bağlaçlı Liste Gerçekleştirimi

- **Yöntem 1 : (Sıralı liste tutularak)** (Artan sıralı öncelik kuyruğunda)

Eleman ekleme, eklenecek elemanın listeyi sıralı tutacak şekilde liste üzerinde dolaşılarak araya eklenmesi şeklinde gerçekleştirilir. Eleman çıkarma da, listenin ilk elemanının (en küçük değer) çıkarılması ile gerçekleştirilir.

- **Yöntem 2 : (Sıralı olmayan liste)** (Artan sıralı öncelik kuyruğunda)

Eleman ekleme kuyruğun herhangi bir yerine yapılabilir. Eleman çıkarma ise eleman bulunana kadar tüm kuyruk boyunca dolaşılması ve elemanın listeden çıkarılması ile gerçekleştirilir.

Öncelik kuyruklarında listelerin sıralanarak kullanımı, sıralanmadan kullanımına göre daha etkindir. **Nedenlerini düşününüz!**

Diğer Bazı Liste Yapıları

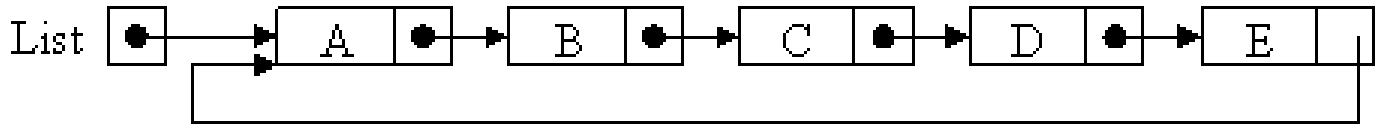
- Dairesel Bağlaçlı Listeler
  Circular Linked Lists

- Çift Bağlaçlı Listeler
  Doubly Linked Lists

- Dairesel Çift Bağlaçlı Listeler
  Circular Doubly Linked Lists

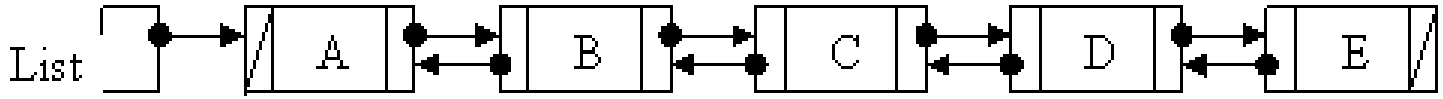
Dairesel Bağlaçlı Listeler

Tüm düğümlerin bir sonraki düğümü gösterdiği bağlı listelerdir. Son elemanın bağı NULL değildir; ilk elemanı gösterir. Böylece dairesel bir yapı oluşur.



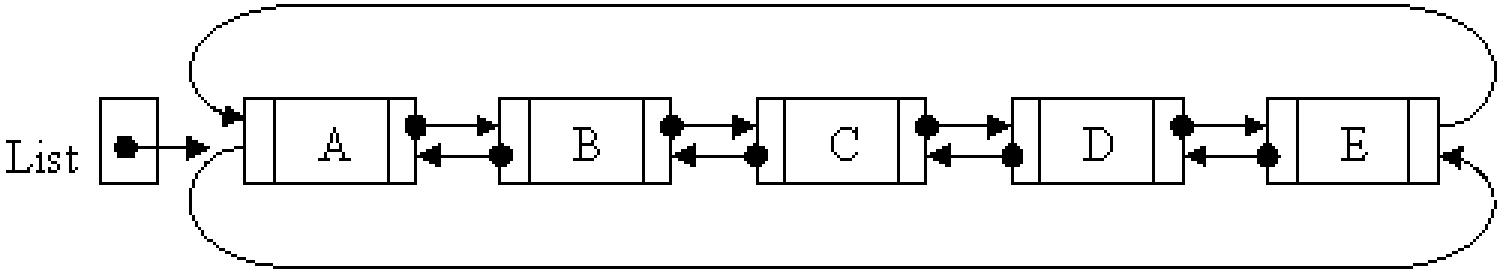
Çift Bağlaçlı Listeler

Her düğümü iki bağ içerdiği bağlı listelerdir. İlk bağ kendinden önceki düğümü gösterirken ikincisi de kendinden sonraki düğümü gösterir. Çift bağlı listelerde, tek bağlı listelerdeki geriye doğru listeleme ve dolaşmadaki zorluklar ortadan kalkar.



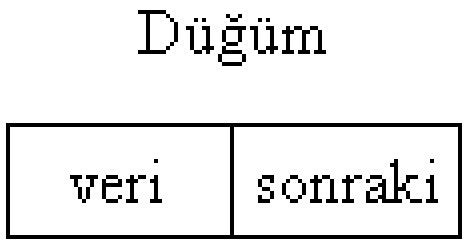
Dairesel Çift Bağlaçlı Listeler

Hem dairesellik hem de çift bağlılık özelliklerine sahip listelerdir. İlk düğümden önceki düğüm son, son düğümden sonraki düğüm de ilk düğümdür.



C# Programlama Dilinde Bağlaçlı Liste Örneği ve Kullanımı

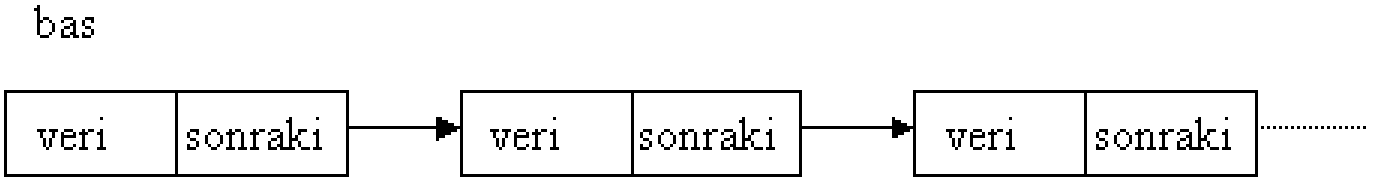
C# Programlama Dilinde Bağlaçlı Liste Düğüm Sınıfı




```
class Dugum
{
  public int veri; // değişik tiplerde çoğaltılabilir
  public Dugum sonraki; // sonraki düğümün referansı (adresi)

  public Dugum(int gelenVeri) // Yapıcı metot
  { veri = gelenVeri; } // Düğüm oluşturulurken değerini
                        // aktarır
  public void yazdir() // Düğümün verisini yazdırır
  { Console.Write(" " + veri); }
}
```

C# Programlama Dilinde Bağlaçlı Liste Sınıfı




```
class BListe
{
// Listenin ilk düğümünün adresini tutar
private Dugum bas;

public BListe() // Bir BListe nesnesi
{ bas = null; } // boş liste olarak
açılır.
```

```
public void listele()
{ // Bağlı Listeyi Baştan Sona Listeler
  Console.WriteLine();
  Console.Write("Bastan Sona Liste : ");
  Dugum etkin = bas;
  while (etkin != null)
  { etkin.yazdir(); etkin = etkin.sonraki;}
}
```

Diğer Bağlaçlı Liste Metotları

```
//Listede anahtar değerini bulur
public Dugum bul(int anahtar)
{
  Dugum etkin = bas;
  while (etkin.veri != anahtar)
  {
    if (etkin.sonraki == null)
      return null;
    else
      etkin = etkin.sonraki;
  };
  return etkin;
}

public void basaEkle(int yeniEleman)
{ // Liste başına eleman ekler
  Dugum yeniDugum = new Dugum(yeniEleman);
  yeniDugum.sonraki = bas;
  bas = yeniDugum;
}
```

```
public Dugum sil(int anahtar)
{ // Verilen anahtar değerindeki düğümü siler
  Dugum etkin = bas;
  Dugum onceki = bas;

  while (etkin.veri != anahtar)
  {
    if (etkin.sonraki == null)
      return null;
    else
      { onceki = etkin; etkin = etkin.sonraki; }
   }

  if (etkin == bas)
    bas = bas.sonraki;
  else
    onceki.sonraki = etkin.sonraki;

  return etkin;
}
```

BListe sınıfını kullanarak bir bağlaçlı liste oluşturan ve test eden sınıf

```
class Program
{
  static void Main(string[] args)
  {
     // liste adlı bir bağlı liste nesnesi oluşturur
     BListe liste = new BListe();

     liste.basaEkle(9);
     for (int i = 8; i >= 1; --i) liste.basaEkle(i);
       liste.listele();

     int deger = 5;
     Dugum d = liste.bul(deger);
     if (d == null)
        Console.WriteLine("\n" + deger + " Listede Yok");
     else
        Console.WriteLine("\n" + deger + " Bulundu");

      Dugum s = liste.sil(5);
      liste.listele();
  }
}
```

Ekran Çıktısı :

Bastan Sona Liste : 1 2 3 4 5 6 7 8 9
5 Bulundu

Bastan Sona Liste : 1 2 3 4 6 7 8 9

Alıştırmalar

- Tamsayı elemanlardan oluşan bir bağlaçlı liste sınıfına, elemanları küçükten büyüğe sıralı tutacak şekilde eleman ekleyen “ekle” metodunu yazınız.

- Aynı metodun listeyi büyükten küçüğe sıralı tutacak sürümünü yazınız.

- Tamsayı elemanlardan oluşan bir çift bağlaçlı liste sınıfı ile metotlarını (listele, ara, ekle, sil, güncelle) yazınız.

- Yolcu (ad, soyad, yas) nesnelerinden oluşan, ada göre sıralı bir tek bağlaçlı liste sınıfını ve listele, ara (ada göre), ekle, sil ve güncelle metotlarını yazınız.

C# Collections
Hazır Linked List Sınıfı

```
LinkedList<int> liste = new LinkedList<int>();

// Listeye Eleman Ekleme
liste.AddLast(5);
liste.AddLast(7);
liste.AddLast(3);
liste.AddLast(4);
liste.AddFirst(10);

Console.WriteLine(liste.ElementAt(0)); // [] erişimi LinkedList'te yok!
// ElementAt() metodu ile erişim var.

LinkedList<int>.Enumerator e = liste.GetEnumerator();
Write(e);

LinkedListNode<int> node = liste.Find(7);
liste.AddAfter(node, 15);
e = liste.GetEnumerator();
Write(e);
```

```
static void Write(IEnumerator<int> e)
{ // Dolaşma
  while (e.MoveNext())
  {
    int value = e.Current;
    Console.Write(value+" ");
  }
  Console.WriteLine();
}
```

Ekran Çıktısı :

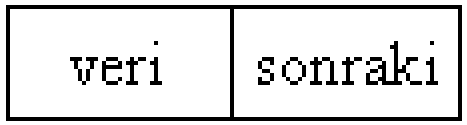
10
10 5 7 3 4
10 5 7 15 3 4

Eleman Silme

- liste.Remove(15); // 15 değerine sahip düğümü siler.

Java Programlama Dilinde Bağlaçlı Liste Örneği ve Kullanımı

Bağlaçlı Liste Düğüm Yapısı ve Sınıfı

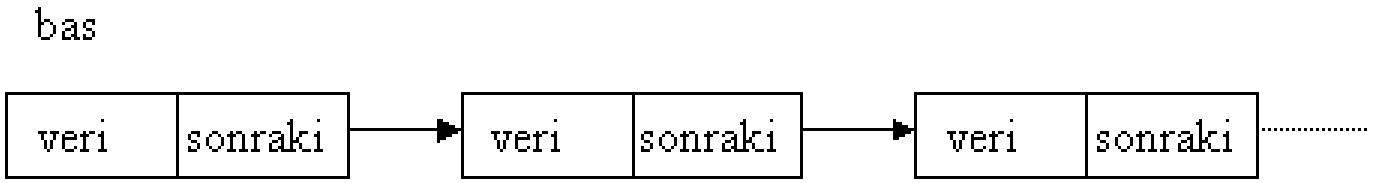



```
class Dugum
{
  public int veri; // değişik tiplerde çoğaltılabilir
  public Dugum sonraki; // sonraki düğümün referansı (adresi)

  public Dugum(int gelenVeri)      // Yapıcı metot
  { veri = gelenVeri; } // Düğüm yaratılırken değerini
                        // aktarır

  public void yazdir() // Düğümün verisini yazdırır
  { System.out.print(" "+veri); }
}
```

Bağlaçlı Liste Yapısı ve Sınıfı




```
class BListe
{
  // Listenin ilk düğümünün adresini tutar
  private Dugum bas;

  public BListe() // Bir BListe nesnesi
  { bas = null; } // boş liste olarak açılır.
```

```
public void listele()
  { // Bağlı Listeyi Baştan Sona
     Listeler
    System.out.println();
    System.out.print("Bastan Sona
     Liste : ");
    Dugum etkin = bas;
    while(etkin!=null)
    { etkin.yazdir();
     etkin=etkin.sonraki; }
  }
```

Bağlaçlı Liste Metotları

```
//Listede anahtar değerini bulur
 public Dugum bul(int anahtar)
 {
   Dugum etkin = bas;
   while(etkin.veri != anahtar)
   {
     if(etkin.sonraki==null)
       return null;
     else
       etkin = etkin.sonraki;
   };
   return etkin;
 }

 public void basaEkle(int yeniEleman)
 { // Liste başına eleman ekler
   Dugum yeniDugum = new
      Dugum(yeniEleman);
   yeniDugum.sonraki = bas;
   bas = yeniDugum;
 }
```

```
public Dugum sil(int anahtar)
{ // Verilen anahtar değerindeki düğümü siler
  Dugum etkin = bas;
  Dugum onceki = bas;

  while(etkin.veri!=anahtar)
  {
    if(etkin.sonraki==null)
      return null;
    else
      { onceki = etkin; etkin = etkin.sonraki; }
  }

  if(etkin==bas)
    bas = bas.sonraki;
  else
    onceki.sonraki = etkin.sonraki;

  return etkin;
 }
}
```

BListe sınıfını kullanarak bir bağlaçlı liste oluşturan ve test eden sınıf

```
class BListeTest
{
  public static void main(String args[])
  {
      // liste adlı bir bağlı liste nesnesi yaratır
    BListe liste = new BListe();

    liste.basaEkle(9);
    for(int i=8; i>=1; --i) liste.basaEkle(i);
    liste.listele();

    int deger = 5;
    Dugum d = liste.bul(deger);
    if(d==null)
      System.out.println("\n"+deger+" Listede Yok");
    else
      System.out.println("\n"+deger+" Bulundu");

    Dugum s = liste.sil(5);
    liste.listele();

  }
}
```

Ekran Çıktısı :

```
// Bastan Sona Liste :  1 2 3 4 5 6 7 8 9
// 5 Bulundu
// Bastan Sona Liste :  1 2 3 4 6 7 8 9
```

Java'daki Hazır LinkedList sınıfının kullanımı

```
import java.util.*;

public class LinkedListExample
{
	public static void main(String args[])
	{

	LinkedList <Integer> liste = new
	LinkedList<Integer>();

	int size;
	Iterator iterator;

	// Listeye Eleman Ekleme
	liste.add(5);
	liste.add(7);
	liste.add(3);
	liste.add(4);
	size = liste.size();
	// Listenin dolaşılması
	System.out.print( "Elemanlar: ");
	iterator = liste.iterator();
	while (iterator.hasNext()){
		System.out.print(iterator.next()+" ");
	}
	System.out.println();
	// İlk eleman olarak 15 değerini ekleme
	liste.addFirst(15); listele(liste);
	// Üçüncü Konuma (2. indise) 25 değerinin eklenmesi
	liste.add(2,25); listele(liste);

	}

	public static void listele(LinkedList<Integer> liste)
	{
		Iterator iterator;
		iterator = liste.iterator();
		while (iterator.hasNext())
		{ System.out.print(iterator.next()+" "); }
		System.out.println();
	}

}
```

```
Elemanlar: 5 7 3 4
15 5 7 3 4
15 5 25 7 3 4
```

Java'daki Hazır LinkedList sınıfının kullanımı - Eleman Silme (Devam)

```
int sonEleman = liste.removeLast(); listele(liste);
System.out.println("Son Eleman : " + sonEleman);

int ikinci = liste.remove(1);
System.out.println("İkinci konumdan silinen eleman : " + ikinci);
```

```
Son Eleman : 4
¦kinci konumdan silinen eleman : 5
```