Bağlı Listeler - Tek Yönlü Bağlı Liste

Doç. Dr. Fatih ÖZYURT

LİSTELER

- Günlük hayatta listeler; alışveriş listeleri, davetiye, telefon listeleri vs. kullanılır.
- Programlama açısından liste; aralarında doğrusal ilişki olan veriler topluluğu olarak görülebilir.
- Veri yapılarında değişik biçimlerde listeler kullanılmakta ve üzerlerinde değişik işlemler yapılmaktadır.

Doğrusal Listeler (Diziler)

- Diziler(arrays), doğrusal listeleri oluşturan yapılardır. Bu yapıların özellikleri şöyle sıralanabilir:

- Doğrusal listelerde süreklilik vardır. Dizi veri yapısını ele alırsak bu veri yapısında elemanlar aynı türden olup bellekte art arda saklanırlar.

- Dizi elemanları arasında başka elemanlar bulunamaz. Diziye eleman eklemek gerektiğinde (dizinin sonu hariç) dizi elemanlarının yer değiştirmesi gerekir.

- Dizi program başında tanımlanır ve ayrılacak bellek alanı belirtilir. Program çalışırken eleman sayısı arttırılamaz veya eksiltilemez.

Doğrusal Listeler (Diziler)

- Dizinin boyutu baştan çok büyük tanımlandığında kullanılmayan alanlar oluşabilir.

- Diziye eleman ekleme veya çıkarmada o elemandan sonraki tüm elemanların yerleri değişir. Bu işlem zaman kaybına neden olur.

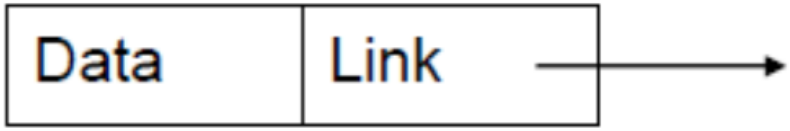
- Dizi sıralanmak istendiğinde de elemanlar yer değiştireceğinden karmaşıklık artabilir ve çalışma zamanı fazlalaşır.

BAĞLI LİSTELER

- Bellekte elemanları ardışık olarak bulunmayan listelere bağlı liste denir.

- Bağlı listelerde her eleman kendinden sonraki elemanın nerede olduğu bilgisini tutar. İlk elemanın yeri ise yapı türünden bir göstericide tutulur. Böylece bağlı listenin tüm elemanlarına ulaşılabilir.

- Bağlı liste dizisinin her elemanı bir yapı nesnesidir. Bu yapı nesnesinin bazı üyeleri bağlı liste elemanlarının değerlerini veya taşıyacakları diğer bilgileri tutarken, bir üyesi ise kendinden sonraki bağlı liste elemanı olan yapı nesnesinin adres bilgisini tutar.



BAĞLI LİSTELER

- Bağlantılı liste yapıları iki boyutlu dizi yapısına benzemektedir. Aynı zamanda bir boyutlu dizinin özelliklerini de taşımaktadır.

- Eleman eklemede de listenin boyutu yetmediğinde kapasiteyi arttırmak gerekmektedir. Bu durumda istenildiği zaman boyutun büyütülebilmesi ve eleman çıkarıldığında listenin boyutunun kendiliğinden küçülmesi için yeni bir veri yapısına ihtiyaç vardır.

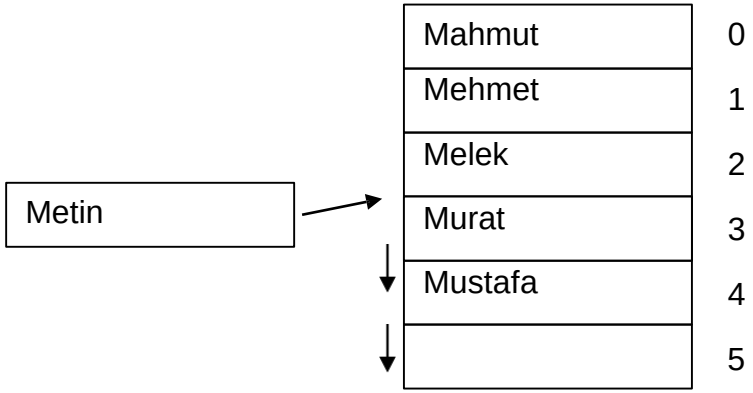
BAĞLI LİSTELER

- Doğrusal veri yapılarında dinamik bir yaklaşım yoktur. İstenildiğinde bellek alanı alınamaz ya da eldeki bellek alanları iade edilemez. Bağlantılı listeler dinamik veri yapılar olup yukarıdaki işlemlerin yapılmasına olanak verir. Bağlantılı listelerde düğüm ismi verilen bellek büyüklükleri kullanılır.

- Bağlantılı listeler çeşitli tiplerde kullanılmaktadır;
  - Tek yönlü doğrusal bağlı liste
  - İki yönlü doğrusal bağlı liste
  - Tek yönlü dairesel bağlı liste
  - İki yönlü dairesel bağlı liste

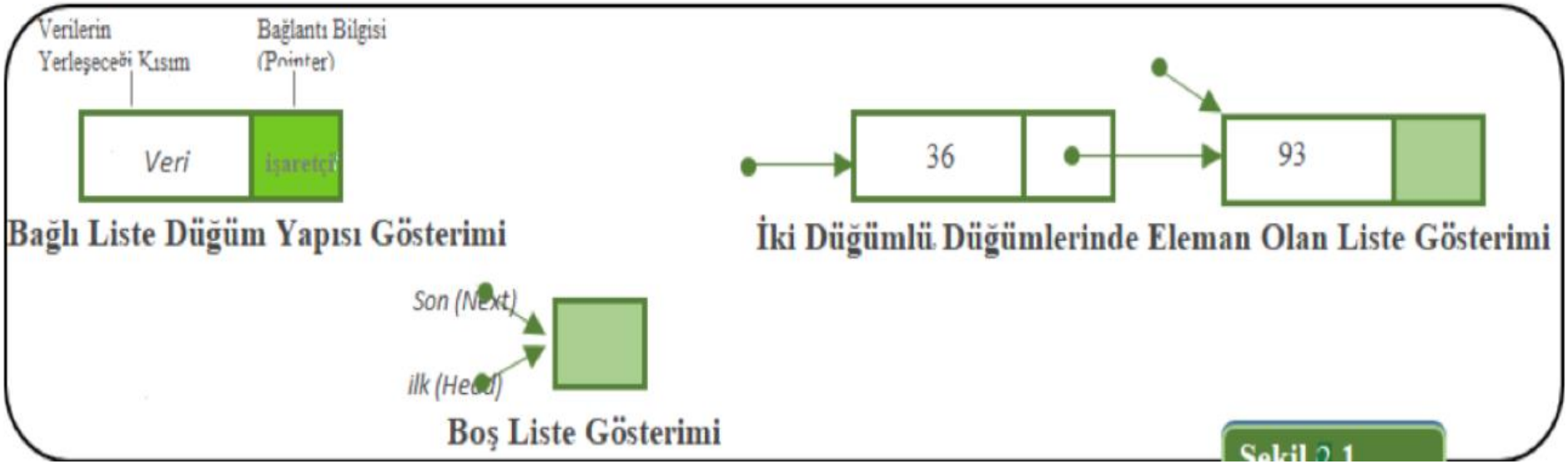
BAĞLI LİSTELER

- **Bağlı Listelerle Dizilerin Karşılaştırılması:**
- Diziler;
  - Boyut değiştirme zordur
  - Yeni bir eleman ekleme zordur
  - Bir elemanı silme zordur
  - Dizinin tüm elemanları için hafızada yer ayrılır
- Bağlı listeler ile bu problemler çözülebilir.



BAĞLI LİSTELER

- Ayrıca, bağlı listelerde
  - Her elaman için ayrı hafıza alanı ayrılır.
  - Bilgi kavramsal olarak sıralıdır ancak hafızada bulunduğu yer sıralı değildir.
  - Her bir eleman (node) bir sonrakini gösterir.



Diziler ve Bağlı Listeler Arasındaki Temel Farklılıklar

1. Bir dizi, benzer tipte veri elemanlarının bir koleksiyonunu içeren veri yapısıdır, buna karşılık bağlı listeler düğümler olarak bilinen sıralı olmayan bağlantılı elemanların bir koleksiyonunu içeren veri yapısıdır. A
2. Dizinin elemanlarına ulaşmak için dizinin adı ve ulaşılmak istenilen dizi elemanının sırasını kullanmak yeterlidir. Buna karşılık, bağlantılı bir listede, bir düğümdeki veri üzerinde işlem yapabilmek için her zaman baştaki düğümden okumaya başlayıp okunması gereken verinin düğümüne kadar ilerlemek gerekir. D
3. Bir dizideki bir elemana erişim hızlıdır, Bağlantılı listede bir düğüme ulaşmak doğrusal zaman alır, bu yüzden biraz daha yavaştır. D
4. Dizilerde ekleme ve silme gibi işlemler çok zaman alır. Öte yandan Bağlantılı listelerde bu işlemlerin performansı dizilere göre daha hızlıdır. A

Diziler ve Bağlı Listeler Arasındaki Temel Farklılıklar

5. Diziler sabit boyuttadır. Bağlı listelerin boyutu ise çalışma sırasında genişletip daraltılabilir. A

6. Bir dizi için bellek tahsisi derleme sırasında yapılırken, bağlı listeler için bellek tahsisi yürütme veya çalıştırma sırasında yapılır. A

7. Dizilerde öğeler (elemanlar) ardışık olarak saklanırken, bağlı listelerde öğeler rastgele saklanır.

8. Diziler bellekte ardışık adresleri kullandığı için dizi üzerinde işaretçi bilgisine ihtiyaç duyulmaz, Buna karşılık, bağlı listelerde, her düğümde verinin yanında bir sonraki düğüme işaret edecek işaretçi için yer ayrılması gerekir. D

Bağlı Listeler Üzerinde Yapılabilecek İşlemler

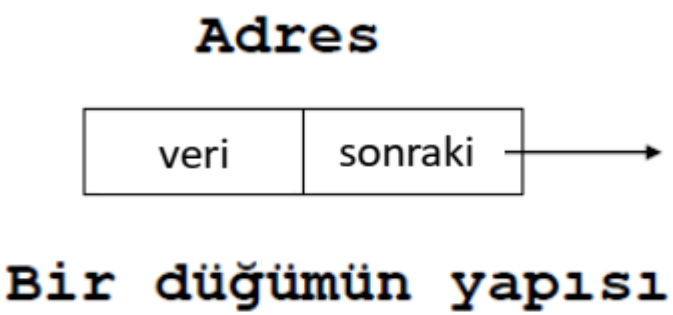
Bağıntılı Liste İşlemleri

- **Listeye eleman ekleme**
- Başa
- Sona
- Sıralı
- **Listeden eleman silme**
- Baştan
- Sondan
- Tümünü
- Belirlenen bilgiye sahip elemanı
- İstenen sıradaki elemanı
- **Arama**
- **Listeleme**
- **Kontrol**
- Boş liste
- Liste boyutu

Tek Yönlü Bağlı Listelerin Genel Yapısı

Örnek: C Dilinde Bağlantılı Liste Yapısı

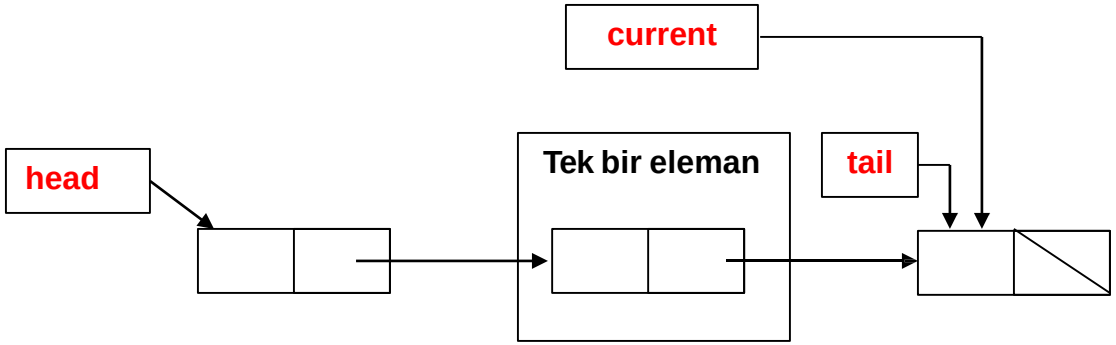
```
Struct dugum
{
int veri;
node *sonraki;
};
```



Listedeki her bir eleman **data (veri)** ve **link (bağlantı-sonraki)** kısmından oluşur. Data kısmı içerisinde saklanan bilgiyi ifade eder. Link kısmı ise kendisinden sonraki elamanı işaret eder

Tek Yönlü Bağlı Listelerin Genel Yapısı

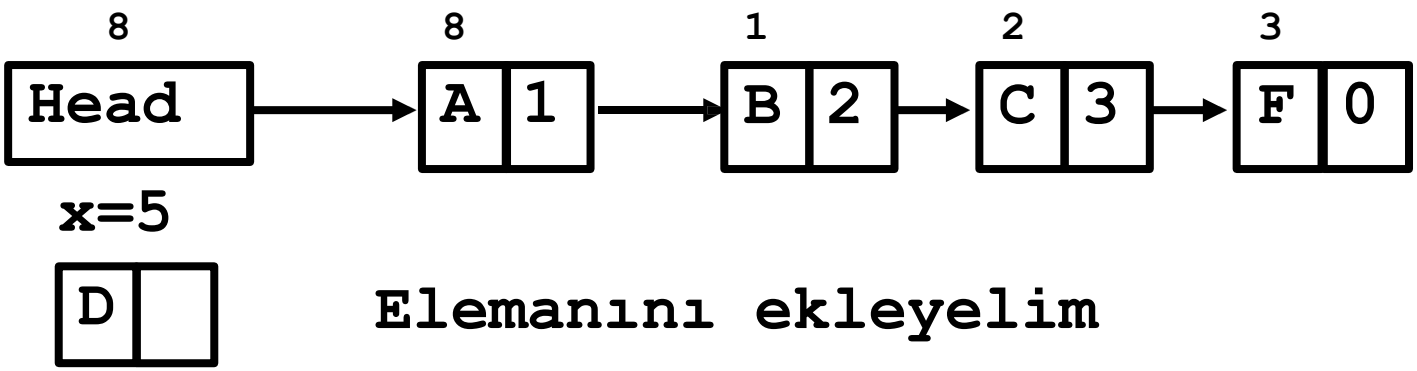
- Listede bir başlangıç (head) elemanı (listenin başlangıcını gösterir ve diğer düğümlere ulaşmak için bu düğüm ile başlamak **mecburidir**), birde sonuncu (tail) elamanı vardır.
- Listede aktif (current) eleman şu anda bilgilerine ulaşabileceğimiz elemandır.



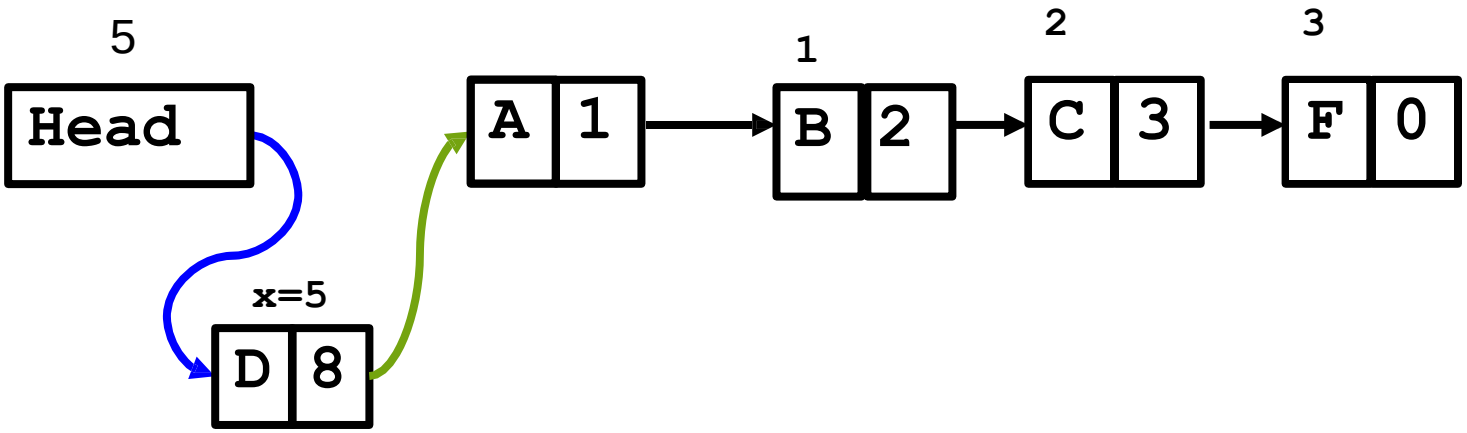
Tek Yönlü Bağlı Listeye Eleman Ekleme

```c
#include <stdio.h>
#include <stdlib.h>
struct dugum    // yapı oluşturma
{
    int veri;
    struct dugum * gosterici;
} ;
int main( ) {
    struct dugum * bir;     // düğüm oluşturma
    bir = (struct dugum  * ) malloc ( sizeof( struct dugum )) ;  // bellekte yer ayırma

    struct dugum * iki;
    iki = (struct dugum  * ) malloc ( sizeof( struct dugum )) ;

    struct dugum * uc;
    uc = (struct dugum  * ) malloc ( sizeof( struct dugum )) ;

    struct dugum * dort;
    dort = (struct dugum  * ) malloc ( sizeof( struct dugum )) ;

    bir->veri= 11;          // veri girişi yaptık
    bir->gosterici= iki;    // bir sonraki düğümü gösterdik

    iki->veri= 22;
    iki->gosterici= uc ;

    uc ->veri= 33;
    uc->gosterici= dort ;

    dort->veri= 44;
    dort->gosterici= NULL ;

    printf("%d  -  %d  -  %d  -  %d ", bir->veri, iki->veri, uc->veri, dort->veri  ) ;

    return 0;
}
```

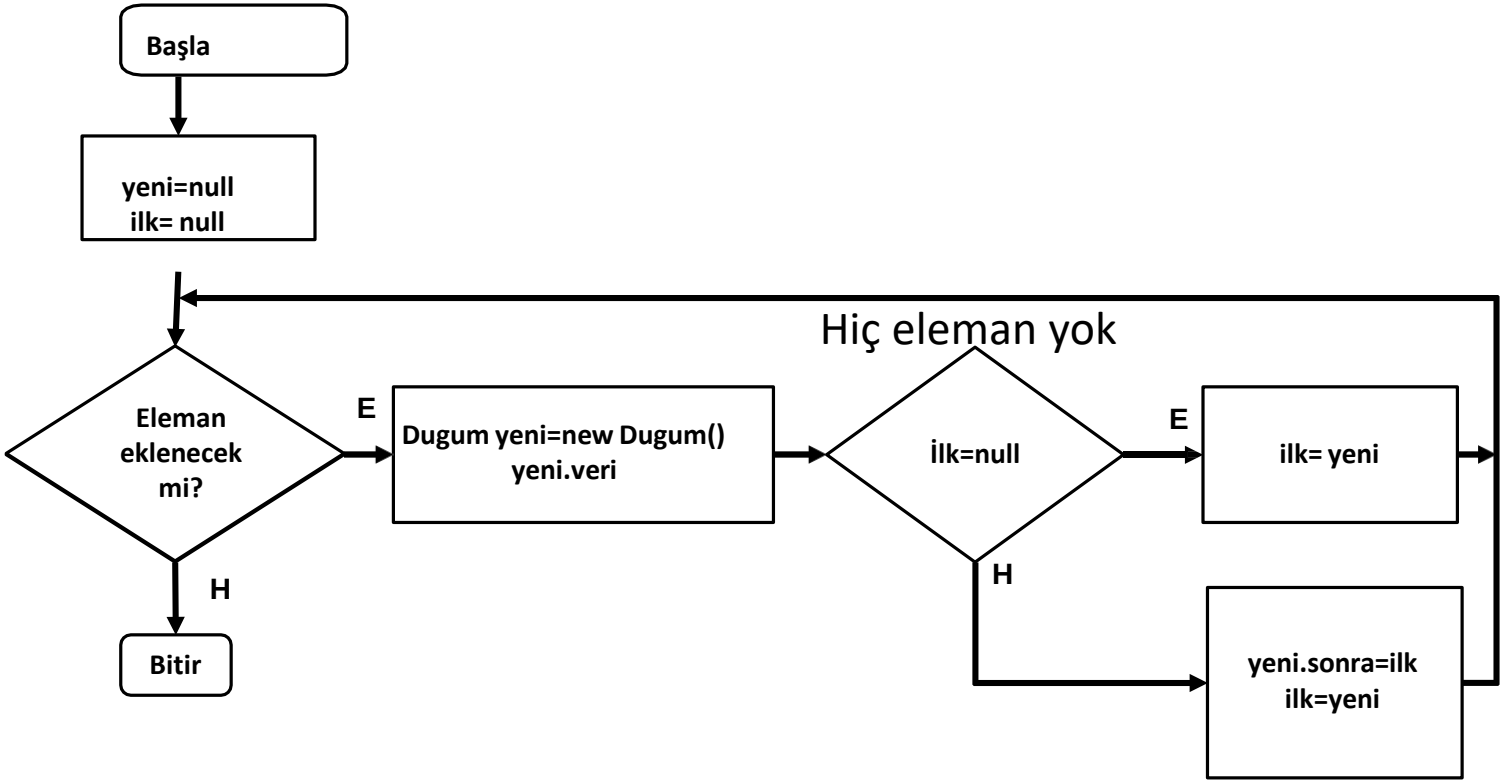
Tek Yönlü Bağlı Listenin Başına Elaman Ekleme



Elemanını ekleyelim



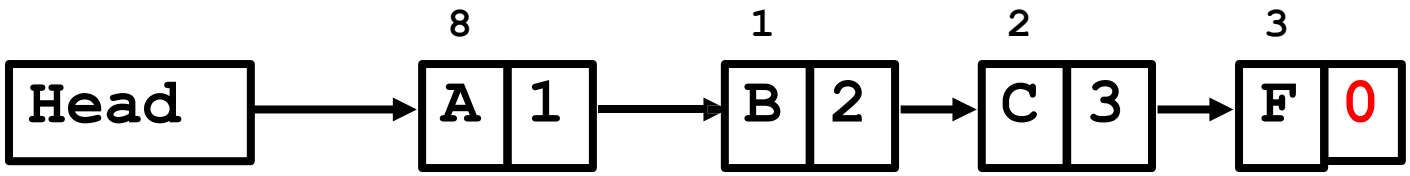
Tek Yönlü Bağlı Liste Başa Ekleme

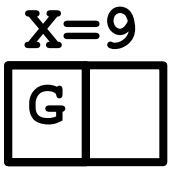


Tek Yönlü Bağlı Liste Başa Ekleme

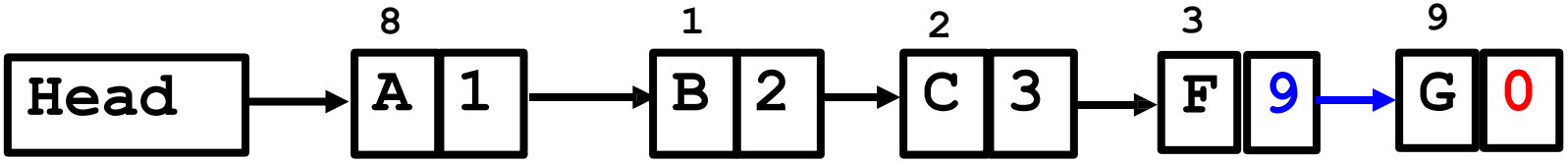
```c
#include <stdio.h>
#include <stdlib.h>

/* run this program using the console pauser or add your own getch, system("pause") or input loop */
struct node {
    int data;
    struct node * next; /* önceki düğüm*/
    struct node * prev; /* sonraki düğüm*/
};

struct node * firstNode=NULL;
struct node * next=NULL;
struct node * prev= NULL;
struct node * temp=NULL;  /* gecici node */

void addFirst (int sayi)
{
    struct node * yenidugum=(struct node *) malloc(sizeof(struct node));  /* hafızada bir düğümlük yer açma*/
    yenidugum->prev=NULL;
    yenidugum->data=sayi;
    if(firstNode==NULL)
    {
        firstNode= yenidugum;   /* ilk düğüm yenidugum oluyor (ilk eklenen düğüm)*/
        firstNode->next=NULL; /* tek düğüm olduğu için*/
    }
    else
    {
        yenidugum->next=firstNode; /* yeni eklenen yenidugum başa geliyor baştaki düğüm bir sonraki düğüm oluyor*/
        firstNode=yenidugum;

    }
}
```

Tek Yönlü Bağlı Listeye Sona Ekleme

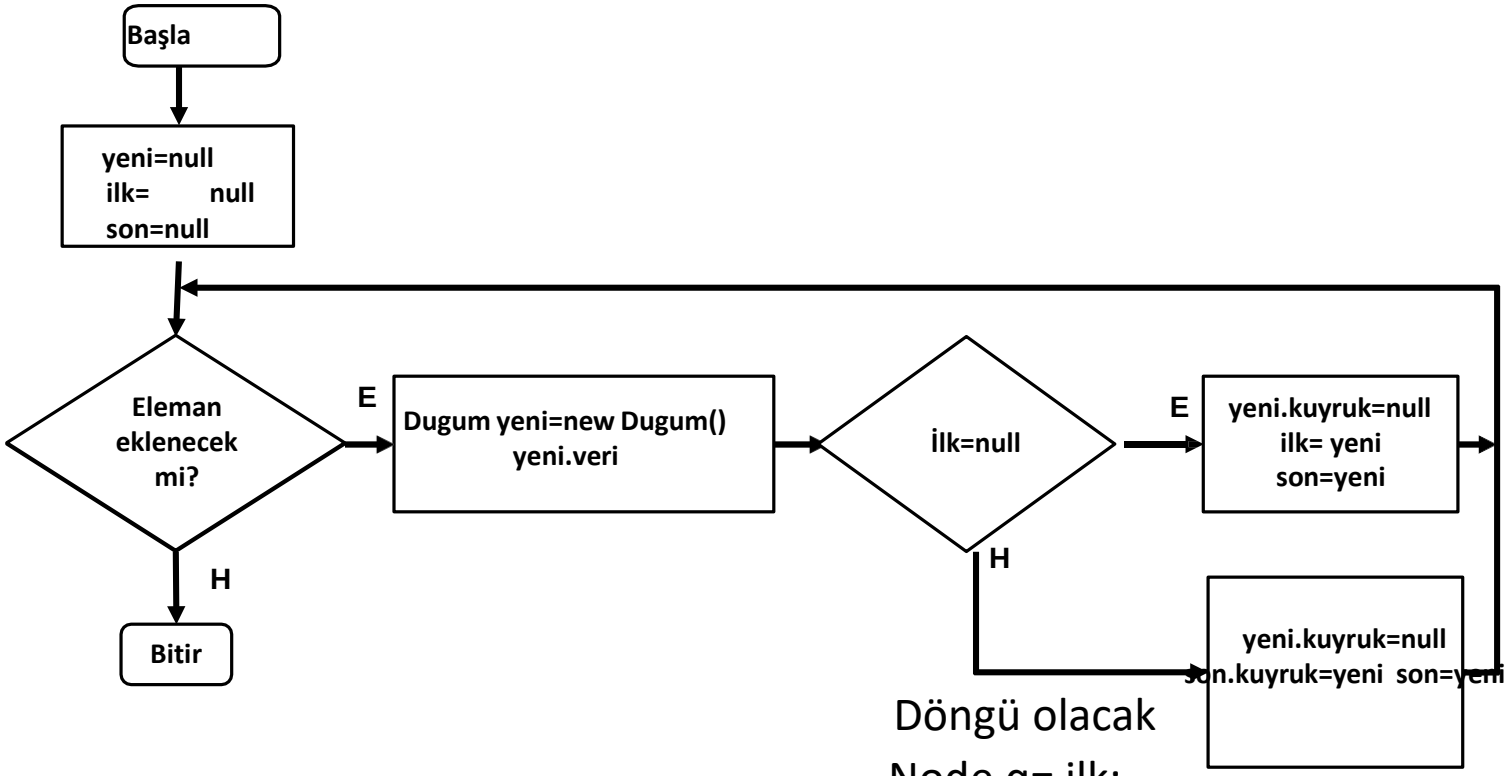




9->G



Tek Yönlü Bağlı Liste Sona Ekleme



Döngü olacak

Node q= ilk;

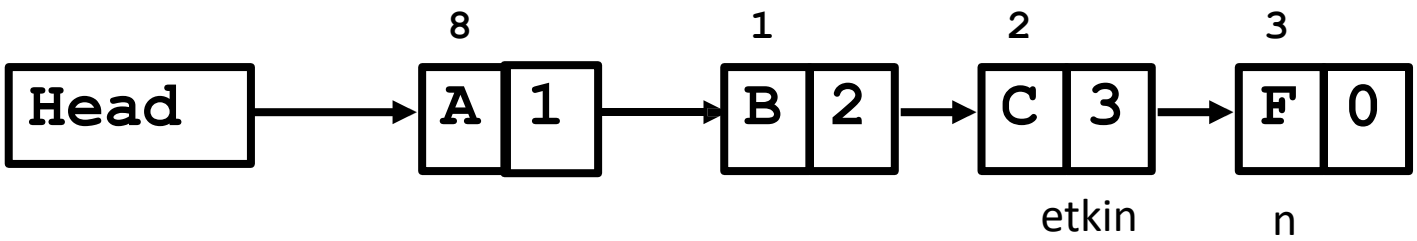
While(q.sonraki=!null)

{q=q.sonraki}

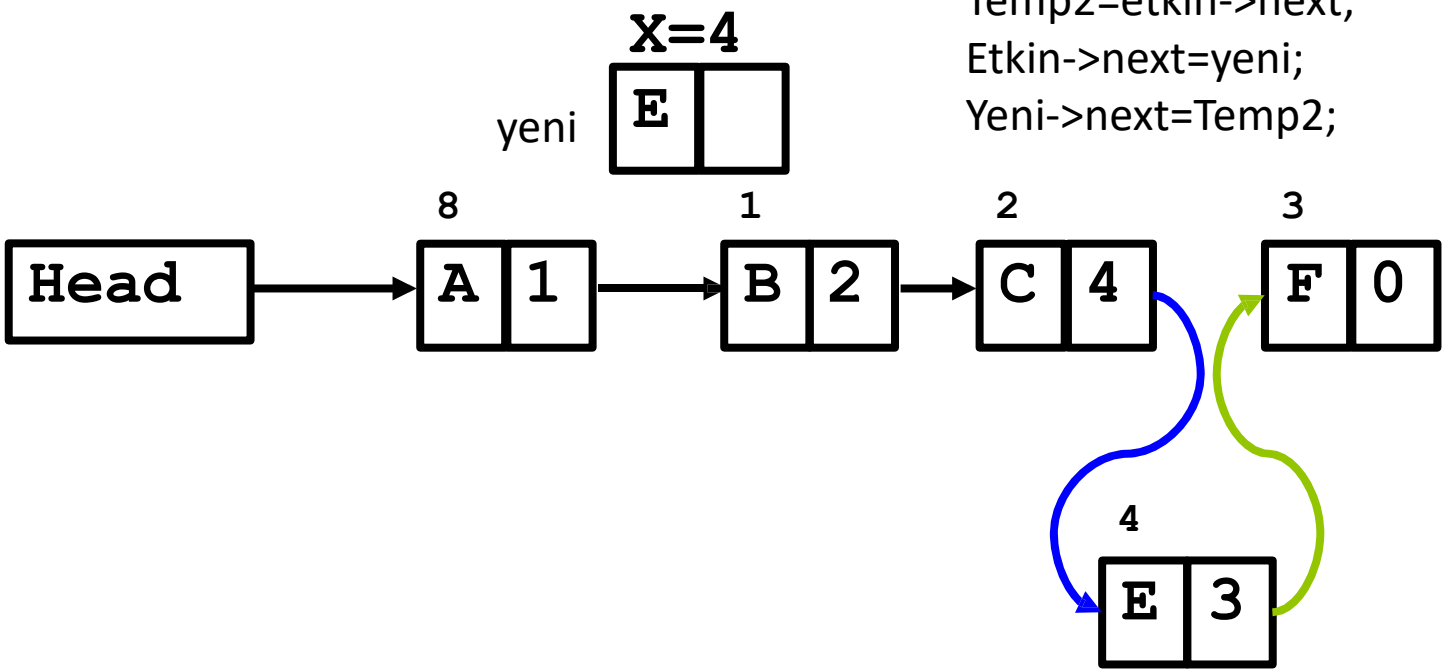
Tek Yönlü Bağlı Liste Sona Ekleme

```
void addLast (int sayi)
{
    struct node * yenidugum=(struct node *) malloc(sizeof(struct node));  /* hafızada bir düğümlük yer açma*/
    yenidugum->next=NULL;
    yenidugum->data=sayi;
    if(firstNode==NULL)
    {
        firstNode= yenidugum;   /* ilk düğüm yenidugum oluyor (ilk eklenen düğüm)*/
        firstNode->prev=NULL;
        firstNode->next=NULL; /* tek düğüm olduğu için*/
    }
    else
    {
        temp=firstNode;
        if(temp->next==NULL)
        {
            temp->next=yenidugum;
            yenidugum->prev=temp;
        }
        else
        while(temp->next!=NULL)
        {
            temp=temp->next;

        }
        temp->next=yenidugum; /* ensona ekledik*/
        yenidugum->prev=temp;
    }
}
```

Tek Yönlü Bağlı Liste-Araya Ekleme



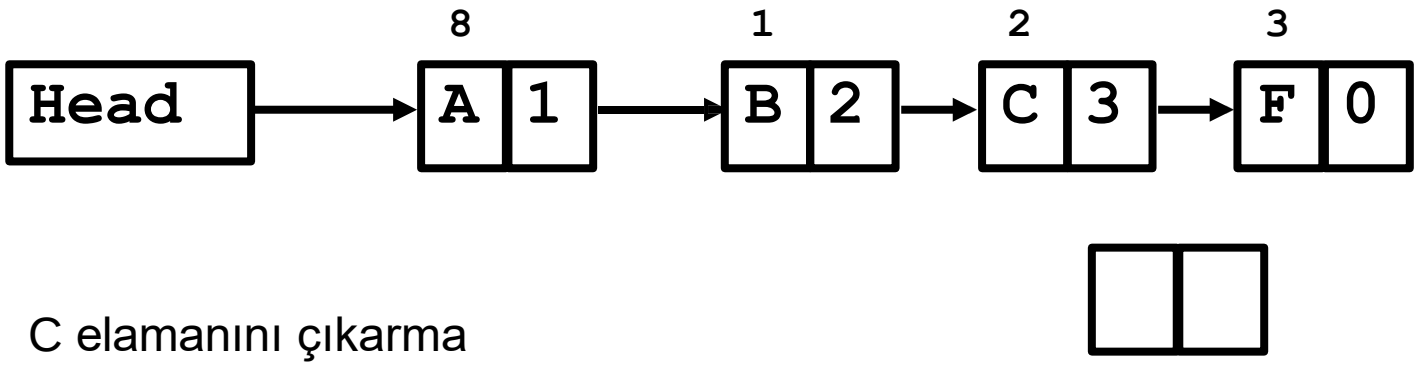
Temp2=etkin->next;
Etkin->next=yeni;
Yeni->next=Temp2;



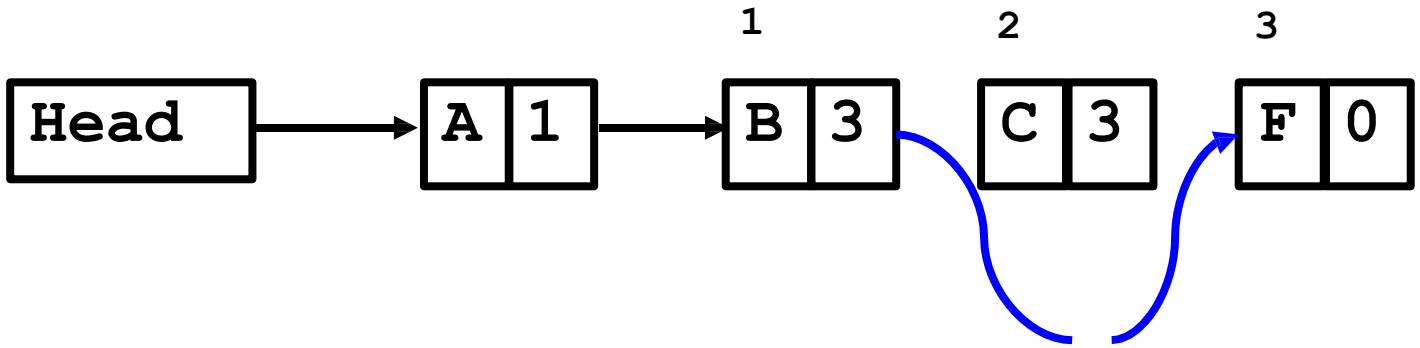
Tek Yönlü Bağlı Liste Sona Ekleme

```
void arayaEkle ( int n, int veri  )
{
    struct node * eleman ;
    eleman = (struct node *) malloc (sizeof(struct node )) ;
    eleman->data= veri;

    temp= start;
    while( temp->next->data != n  )
    {
        temp= temp->next;
    }

    struct node * temp2 ;
    temp2= (struct node * ) malloc ( sizeof(struct node ))  ;

    temp2= temp->next;
    temp->next =  eleman;
    eleman->next = temp2;

}
```

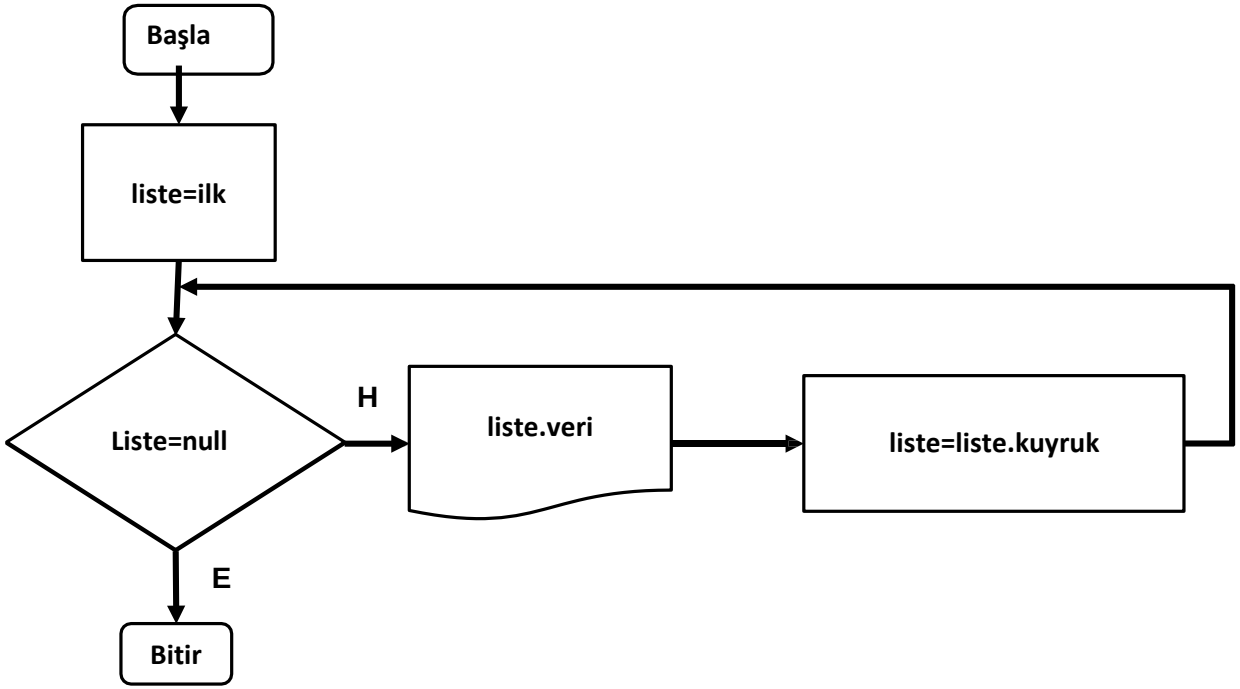
Tek Yönlü Bağlı Listeden Elaman Çıkarma



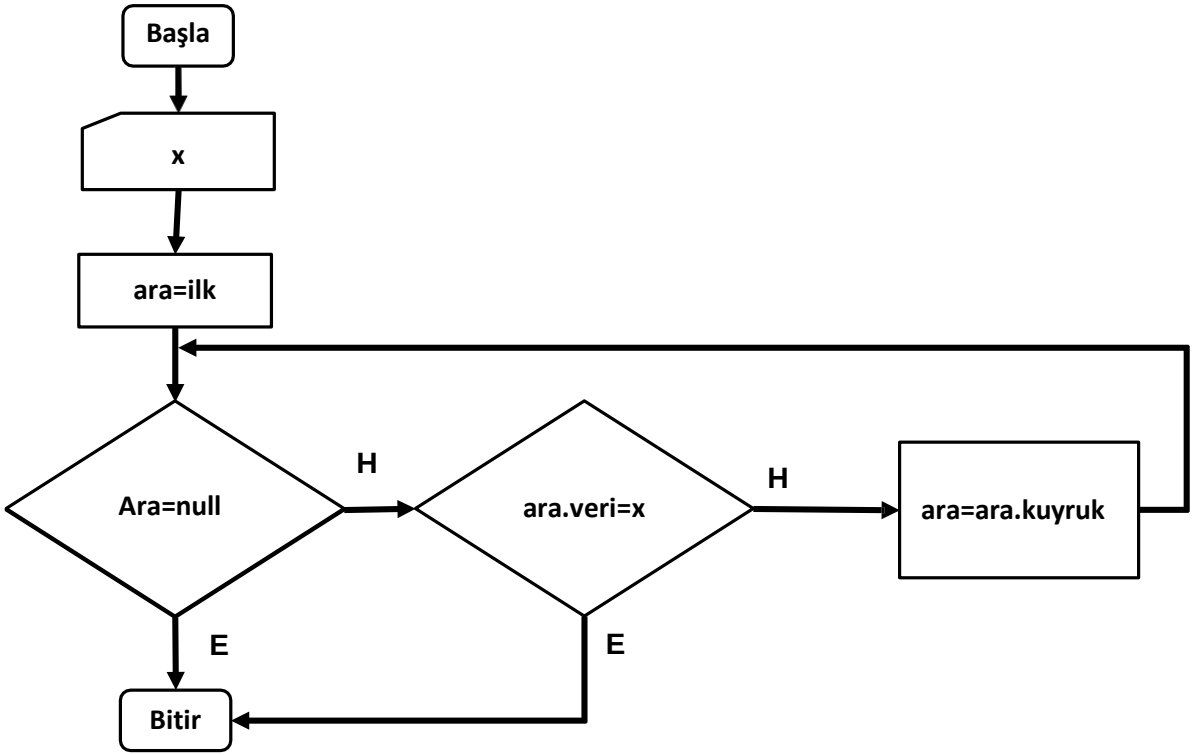
C elamanını çıkarma



Tek Yönlü Bağlı Liste- Listeleme



Tek Yönlü Bağlı Liste- Arama



Kaynaklar

Veri Yapıları ve Algoritmalar – Dr. Rifat ÇÖLKESEN, Papatya yayıncılık
Veri Yapıları ve Algoritmalar-Dr. Öğ. Üyesi Ömer ÇETİN
Veri Yapıları – Prof. Dr. Erkan TANYILDIZI