BLM 112- Programlama Dilleri II

Hafta 4: Bağlı Listeler

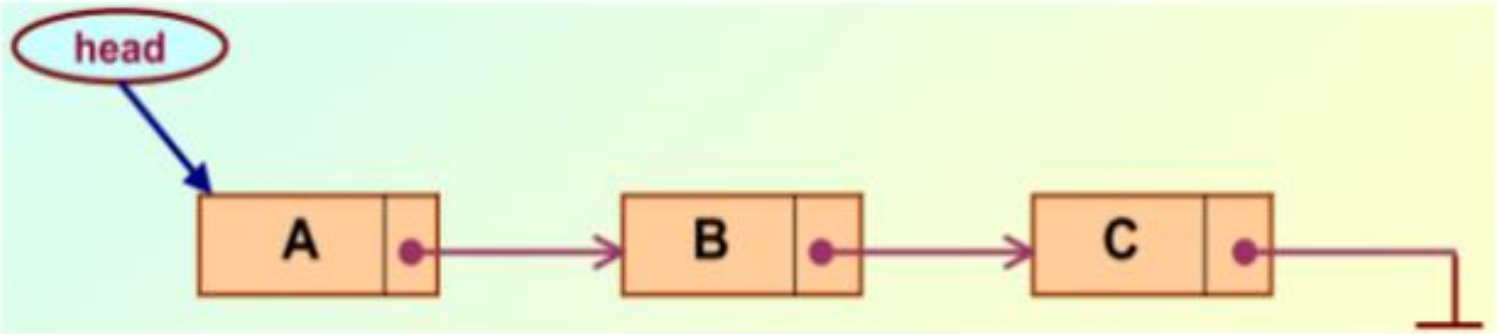
Yrd. Doç. Dr. Ümit ATİLA

BAĞLI LİSTELER

- Bağlı listeler konusuna çalışmanın iki yönden faydası var.
  - 1- Bağlı listeler gerçek programlarda kullanılabilecek bir veri yapısıdır. Bağlı listelerin güçlü ve zayıf yönlerini bilmek algoritmaların çalışma zamanı karmaşıklığı, kapladığı alan karmaşıklığı yönlerini düşünmenize yardımcı olur.
  - Bağlı listeler pointer'ların anlaşılması için iyi bir yöntemdir.

BAĞLI LİSTELER

- Bağlı liste çalışma zamanı sırasında değişebilen bir veri yapısıdır.
  - Ardışık elemanlar pointer ile bir birine bağlanır.
  - Son eleman NULL değeri gösterir.
  - Programın çalışması sırasında boyutu büyüyebilir veya küçülebilir.

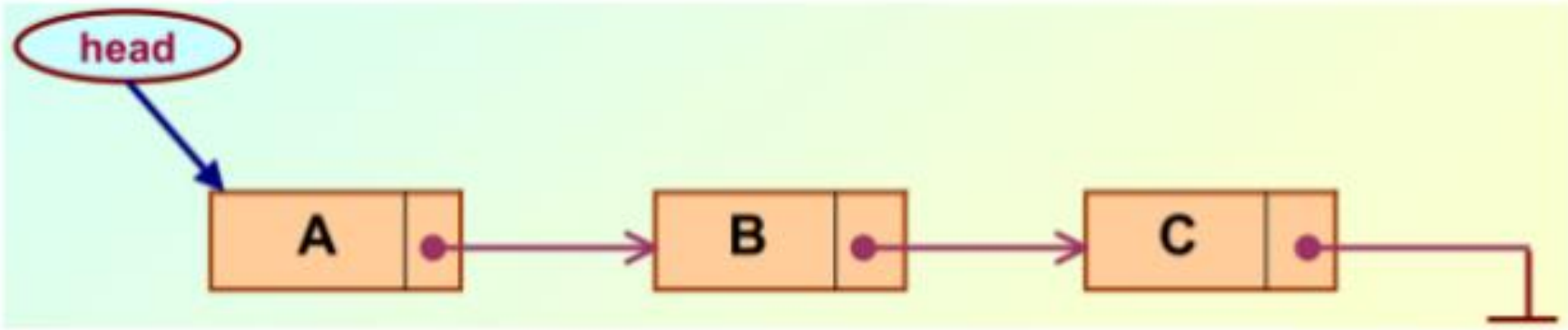


DİZİLER vs. BAĞLI LİSTELER

- Diziler aşağıdaki hususlarda uygundur:
  - En sona eleman ekleme ve en sondan eleman silme
  - Her hangi bir elemana erişme
- Bağlı listeler aşağıdaki hususlarda uygundur:
  - Eleman ekleme
  - Eleman silme
  - Sıralı erişim gerektiren uygulamalar
  - Eleman sayısının önceden tahmin edilemediği durumlar

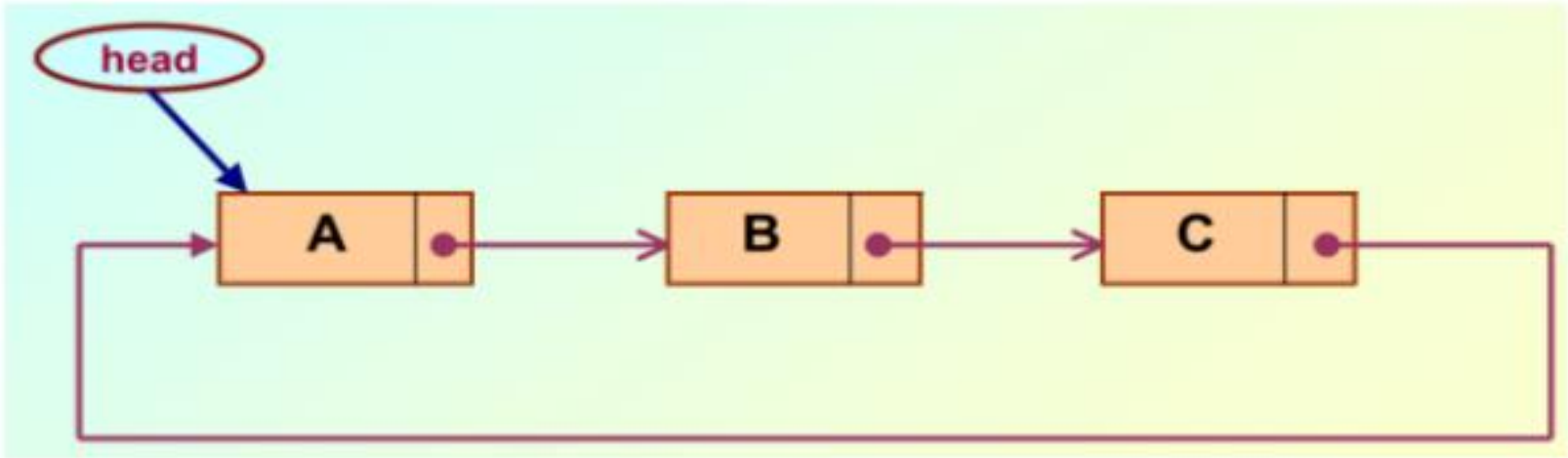
Liste Tipleri

- Bağlantıların yapılış biçimlerine göre listeler birkaç tiptir.
  - 1- Doğrusal bağlı listeler



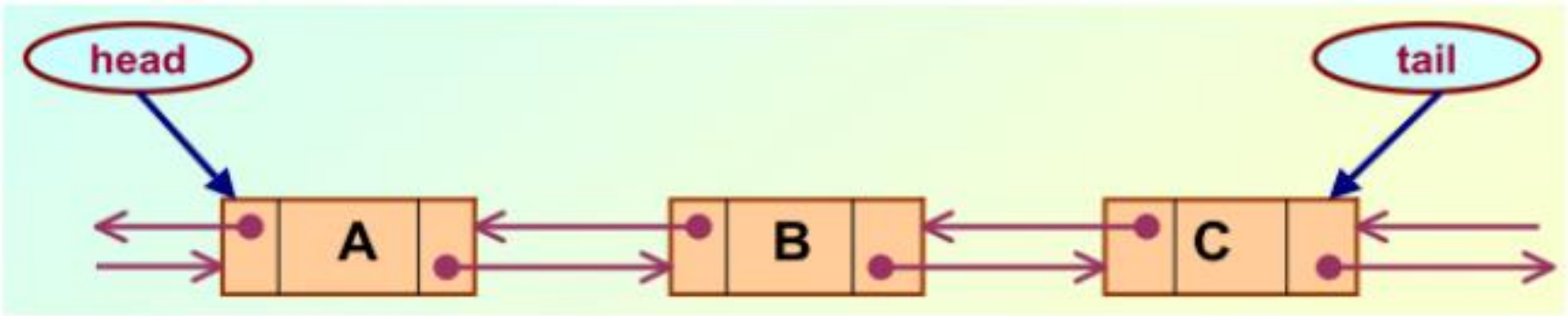
Liste Tipleri

- 2- Dairesel bağlı listeler
  - Son eleman listenin ilk elemanını gösterir.



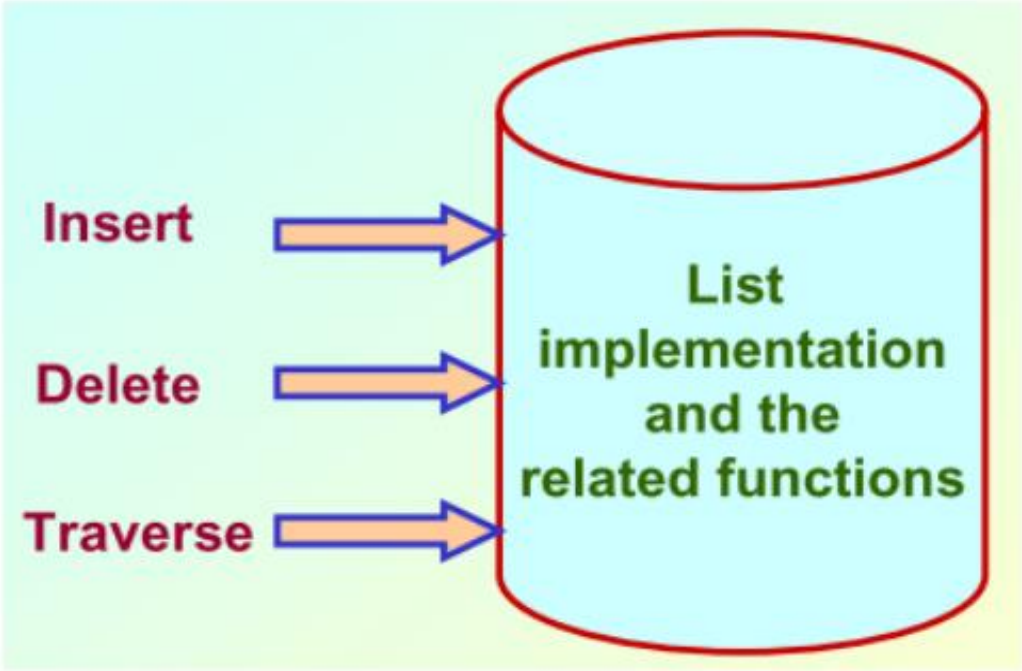
Liste Tipleri

- 3- İki bağlı listeler
  - Komşu düğümler arasında iki yönde de bağlantı kurulur.
  - Liste içerisinde hem ileri hem de geri yönde hareket edilebilir.



Bağlı Listeler

- Bağlı listelerin veri tipi programcı tarafından tasarlanır.
- int, float gibi veri tiplerinden daha karmaşık olur.
- Listelerde genel amaç ;



Bağlı Listelerde Temel İşlemler

- Liste oluşturma
- Listede dolaşma
- Listeye eleman ekleme
- Listeden eleman silme
- İki listeyi birleştirme

Bağlı Listeler

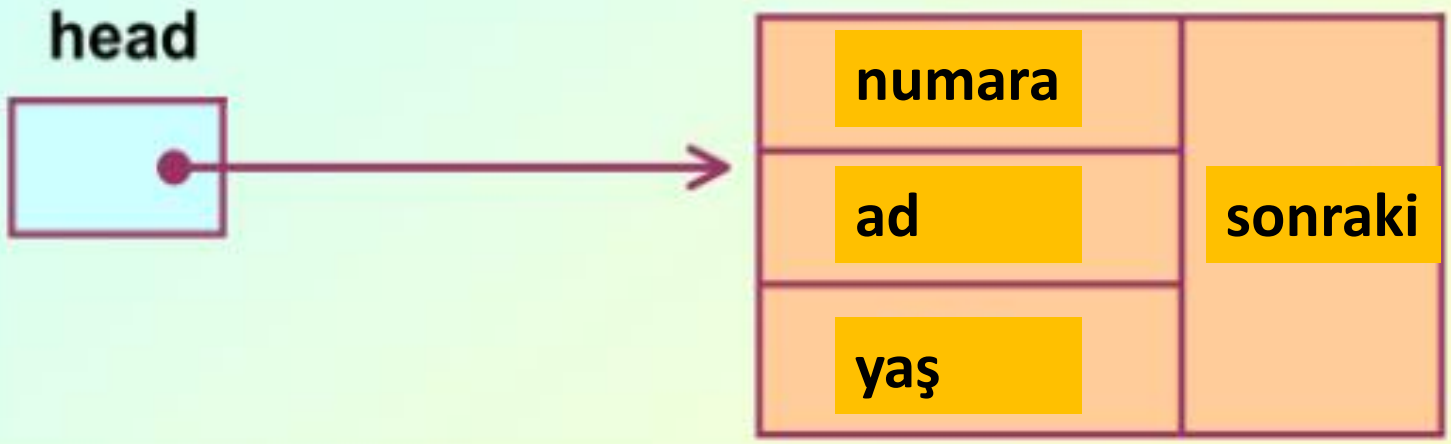
- Listedeki bir düğümün yapısının şu şekilde olduğunu düşünelim:

```
 3  struct personel {
 4      int numara;
 5      char ad[25];
 6      int yas;
 7      struct personel *sonraki;
 8  };
 9  //Kullanıcı tanımlı veri tipi "dugum"
10  typedef struct personel dugum;
```

Doğrusal Liste Oluşturma

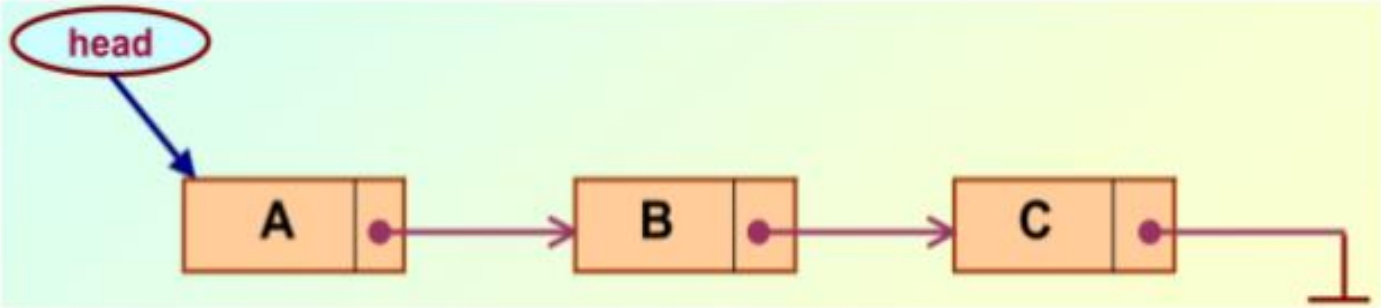
- İlk önce bir düğüm oluşturulmalı ve head'in bu düğümü göstermesi sağlanmalıdır.

```
head = (node *) malloc(sizeof(node));
```



Doğrusal Liste Oluşturma

- Eğer başlangıç bağlı listesinde N adet düğüm olacaksa:
  - N adet kayıt için teker teker hafızadan yer ayır.
  - Kayıtların alan bilgileri girilir.
  - Kayıtların bağlantıları düzenlenir böylece zincir kurulur.



Doğrusal Liste Oluşturma

```
16  dugum * listeOlustur()
17  {
18      int k,n;
19      dugum *p, *head;
20      printf("Kaç eleman gireceksiniz");
21      scanf("%d",&n);
22      for(k=0;k<n;k++)
23      {
24          if(k==0)
25          {
26              head = (dugum *)malloc(sizeof(dugum));
27              p = head;
28          }
29          else
30          {
31              p->sonraki = (dugum *)malloc(sizeof(dugum));
32              p=p->sonraki;
33          }
34          scanf("%d %s %d",&p->numara,p->ad,&p->yas);
35      }
36      p->sonraki = NULL;
37      return head;
38  }
```

Doğrusal Listede Dolaşma

- Liste oluşturulduktan sonra ve head düğümü listenin ilk düğümünü gösterir hale gelince:
  - Pointer'ları takip et.
  - Düğümlere sıra geldikçe onların içeriklerini yazdır.
  - Sıradaki pointer NULL değeri gösteriyorsa dur.

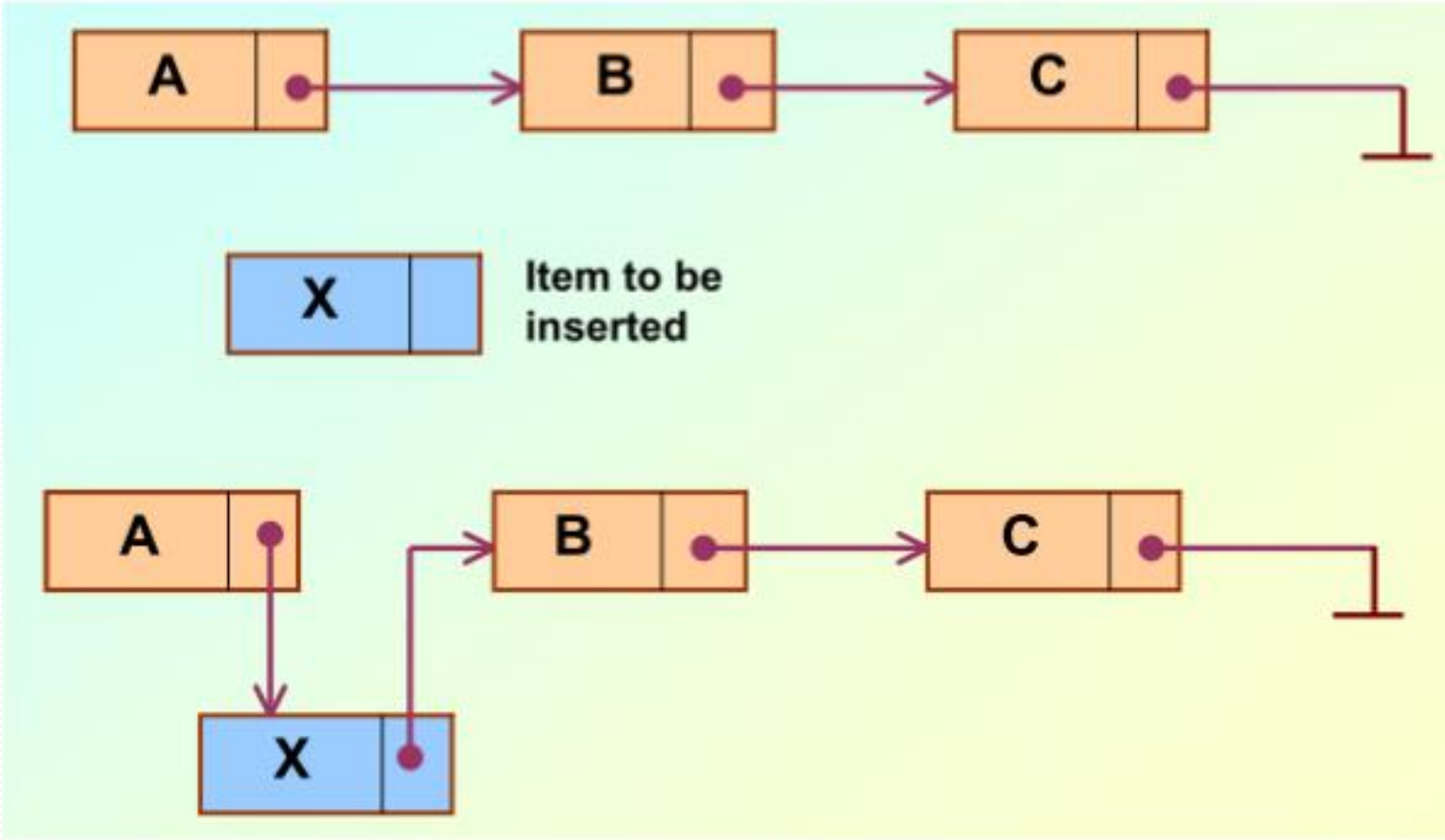
Doğrusal Listede Dolaşma

```
39  void listeDolas(dugum *head)
40  {
41      int sayac =1;
42      dugum *p;
43      p = head;
44      while (p!= NULL)
45      {
46          printf("Dugum %d:%d %s %d",sayac,p->numara,p->ad,p->yas);
47          sayac++;
48          p = p->sonraki;
49          printf("\n");
50      }
51  }
```

Listeye Düğüm Ekleme

- Ekleme için:
  - Yeni kayıt oluşturulur.
  - Yeni kaydın sonraki göstericisi kendinden bir sonra gelecek kaydı gösterecek şekilde ayarlanır.
  - Yeni kayıttan önceki kaydın sonraki göstericisi yeni kaydı gösterecek şekilde ayarlanır.
- Belli bir düğümden önce düğüm ekleme biraz daha karışıktır.
  - Burada anahtar olarak isimlendirilen bir değer kullanılır.
  - Bizim örneğimizde anahtar alan «numara».

Listeye Düğüm Ekleme



Listeye Düğüm Ekleme

- Düğüm başa ekleniyorsa
  - Sadece bir tane «sonraki» göstericisi değiştirilir.
    - Head, yeni düğümü gösterecek şekilde ayarlanır.
    - Yeni düğüm daha önce ilk düğüm olan düğümü gösterir.
- Düğüm sona ekleniyorsa
  - İki tane <sonraki» göstericisi değiştirilir.
    - Son düğüm artık yeni düğümü gösterir.
    - Yeni düğüm NULL gösterir.
- Düğüm araya ekleniyorsa
  - İki tane <sonraki» göstericisi değiştirilir.
    - Yeni düğüm önceki düğümün sonrakini gösterir.
    - Önceki düğüm artık yeni düğümü gösterir.

Listeye Düğüm Ekleme

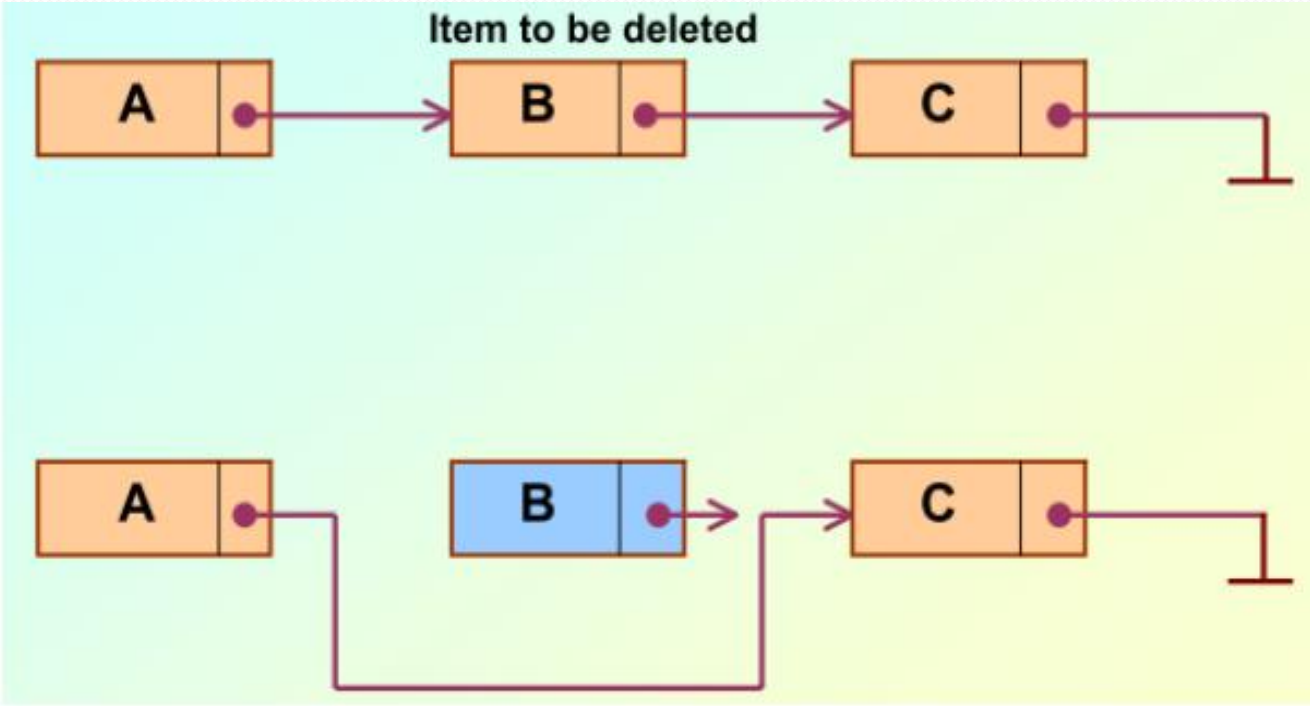
```
49 void dugumEkle(dugum **head)
50 {
51     int kayitNo;
52     dugum *p, *q, *yeni;
53     yeni = (dugum *) malloc(sizeof(dugum));
54     printf("\nEklenecek veriyi gir.Numara?Ad?Yas?\n");
55     scanf("%d %s %d",&yeni->numara,yeni->ad,&yeni->yas);
56
57     printf("Hangi kayit no'dan once eklenecek");
58     scanf("%d",&kayitNo);
59
60     p= *head;
61     if(p->numara==kayitNo) //başa eklenecekse
62     {
63         yeni->sonraki = p;
64         *head = yeni;
65     }
```

Listeye Düğüm Ekleme

```
69      else
70      {
71          while(p->sonraki!=NULL && p->numara!=kayitNo)
72          {
73              q = p;
74              p= p->sonraki;
75          }
76          if(p==NULL) //Sona eklenecekse
77          {
78              q->sonraki = yeni;
79              yeni->sonraki = NULL;
80          }
81          else if(p->numara == kayitNo) //Araya eklenecekse
82          {
83              yeni->sonraki = p;
84              q->sonraki = yeni;
85          }
86      }
87  }
```

Listeden Düğüm Silme

- Silinecek düğümden bir önceki düğümün sonraki göstericisi değiştirilerek silinecek düğümden sonraki düğümü göstermesi sağlanır.



Listeden Düğüm Silme

- Belli bir düğümü silmek için (numara bilgisi verilen düğüm):
  - 3 durum söz konusudur.
    - Silinecek düğüm ilk düğüm olabilir
    - Silinecek düğüm son düğüm olabilir.
    - Silinecek düğüm arada bir düğüm olabilir.

Listeden Düğüm Silme

```
85  void dugumSil(dugum **head)
86  {
87      int kayitNo;
88      dugum *p, *q;
89      printf("Hangi kayit no silinecek");
90      scanf("%d",&kayitNo);
91
92      p= *head;
93      if(p->numara==kayitNo)//ilk düğüm siliniyorsa
94      {
95          head = p->sonraki;
96          free(p);
97      }
```

Listeden Düğüm Silme

```
101         else
102         {
103             while(p->sonraki!=NULL && p->numara!=kayitNo)
104             {
105                 q = p;
106                 p= p->sonraki;
107             }
108             if(p==NULL)
109             {
110                 printf("Uygun kayit bulunamadi.Silme basarisiz!");
111             }
112             else if(p->numara == kayitNo) //Aradaki silinecekse
113             {
114                 q->sonraki = p->sonraki;
115                 free(p);
116             }
117     }
```

Tek Bağlı Liste Uygulaması

```
116 int main(void)
117 {
118     int secim=0;
119     dugum * aktif;
120
121     printf("1-Liste Olustur\n2-Liste Dolas\n3-Dugum Sil\n4-Dugum Ekle\n5-Cikis");
122     while(1)
123     {
124         printf("\nSecim [1-5]?\n");
125         scanf("%d",&secim);
126         switch(secim)
127         {
128             case 1 : aktif = listeOlustur();
129                      listeDolas(aktif);
130                      break;
131             case 2 : listeDolas(aktif);break;
132             case 3 : dugumSil(&aktif);
133                      listeDolas(aktif);
134                      break;
135             case 4 : dugumEkle(&aktif);
136                      listeDolas(aktif);
137                      break;
138             case 5 : exit(0);break;
139             default: printf("Yanlis secim");break;
140         }
141     }
142     while( getchar() != '\n' ) { /*do nothing*/} ;
143         getchar() ; /* wait */
144     return 0;
145 }
```

Uygulama-2

- Dışardan aldığı isimleri alfabetik sıra ile listeleyen, listeye yeni düğüm ekleyen, belirtilen düğümü silen ve düğümdeki en uzun isimi bulan tek bağlı liste uygulaması.

Uygulama-2

```
 5  struct listyapi
 6  {
 7    char   adi[21];
 8    struct listyapi *sonraki;
 9  };
10
11  //artık listyapi yerine listnode kullanılacak
12  typedef struct listyapi listnode;
13  // *headnode listnode tipinde bir işaretçidir.
14  //Herzaman listenin başını gösterecek
15  listnode *headnode;
```

Uygulama-2

```
17  void ara(char *searchthis, listnode **prevnode) /
18  {
19    listnode *c;
20    c = headnode;
21    *prevnode = c;
22    while ((c->sonraki != NULL))
23      {
24        c = c->sonraki;
25        if (strcmp(c->adi, searchthis) >= 0) break;
26        *prevnode = c;
27      }
28  }
```

Uygulama-2

```
29 void kayit(char *s)
30 /* prevnode kayidi newnode kayidini,
31 newnode kayidi prevnode'nin daha once gosterdigi kayidini gosterir. */
32 {
33   listnode *newnode, *prevnode;
34   newnode = (listnode *) malloc(sizeof(listnode)); /* yeni kayida yer at */
35   strcpy(newnode->adi, s);              /*bilgiyi yeni kayida yaz */
36   ara(newnode->adi, &prevnode);  /* listedeki yerini bul */
37   newnode->sonraki = prevnode->sonraki;  /* listeye ekle */
38   prevnode->sonraki = newnode;
39 }
```

Uygulama-2

```
40  void iptal(char *s)
41  /* newnode kayidi silinir.
42  prevnode kayidi newnode kayidinin
43  gosterdigi kayidini gosterir. */
44  {
45    listnode *newnode, *prevnode;
46    ara(s, &prevnode);
47    newnode = prevnode->sonraki;
48    prevnode->sonraki = newnode->sonraki;
49    free(newnode);
50  }
```

Uygulama-2

```
51 void listlist(void)
52 {
53   listnode *currentnode;
54   currentnode = headnode;
55   if (currentnode != NULL) currentnode = currentnode->sonraki;
56   while (currentnode != NULL)
57     {
58       printf("%s  ",currentnode->adi);
59       currentnode = currentnode->sonraki;
60     }
61   printf("\n");
62 }
```

Uygulama-2

```
63 void enUzunBul(void)
64 {
65   listnode *currentnode, *enuzun;
66   currentnode = headnode;
67   if (currentnode != NULL)
68     {
69         currentnode = currentnode->sonraki;
70     }
71   enuzun=currentnode;
72   while (currentnode != NULL)
73     {
74       if (strlen(currentnode->adi) >= strlen(enuzun->adi))
75         {
76           enuzun = currentnode;
77         }
78       currentnode = currentnode->sonraki;
79     }
80   printf("%s  %d",enuzun->adi,strlen(enuzun->adi));
81   getchar();
82 }   /* En uzun isim; */
```

Uygulama-2

```
83  void main()
84  {
85    char      sec;
86    char      s[21];
87    headnode = (listnode *) malloc(sizeof(listnode));
88    strcpy(headnode->adi," listenin basi");
89    headnode->sonraki = NULL;
90    do
91      {
92        system("cls");
93        listlist();
94        printf("\n\n1 - Ekle\n2 - iptal\n3 - En Uzun isim\n4 - Çıkış\n\nSec :");
95        sec = getche();
96        switch (sec)
97          {
98          case '1':printf("\nAdi :"); gets(s);
99            kayit(s);break;
100         case '2':printf("Adi \n");  gets(s);
101           iptal(s);break;
102         case '3':enUzunBul();break;
103         case '4':exit(0);break;
104         }
105     }
106   while (1);
107 }
```