

# Algoritma ve Programlama II

## Hafta 10

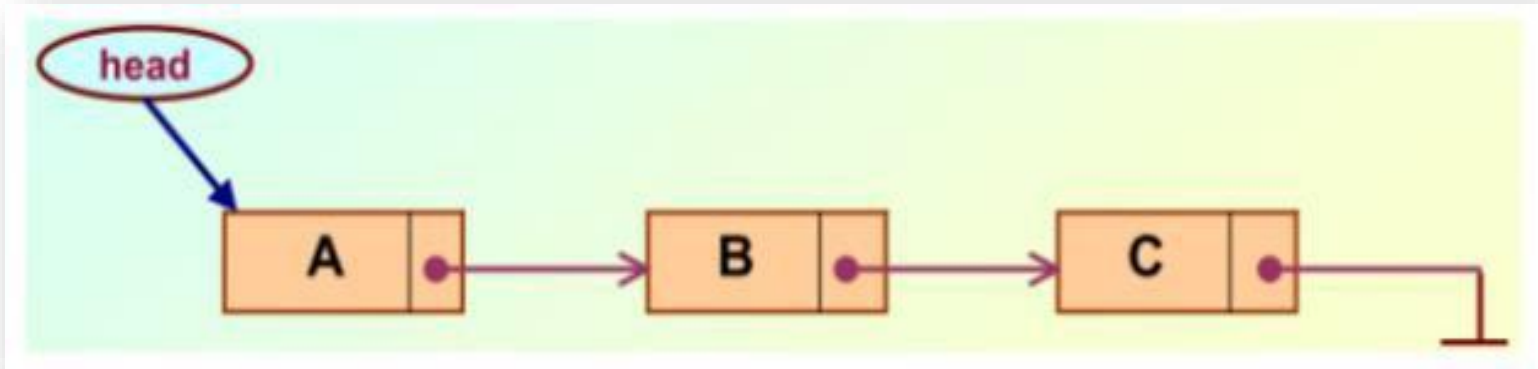
### Tek Bađlı Doğrusal Listeler

# Bağlı Listeler

- ▶ Bağlı listeler konusuna çalışmanın bazı faydaları var.
- ▶ 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 işaretçilerin anlaşılması ve algoritmaların işaretçiler ile birleşimi için iyi bir yöntemdir.

# Bağlı Listeler

- ▶ 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 istenildiği şekilde büyüyebilir veya küçülebilir.
- ▶ Bellek israfı yoktur.

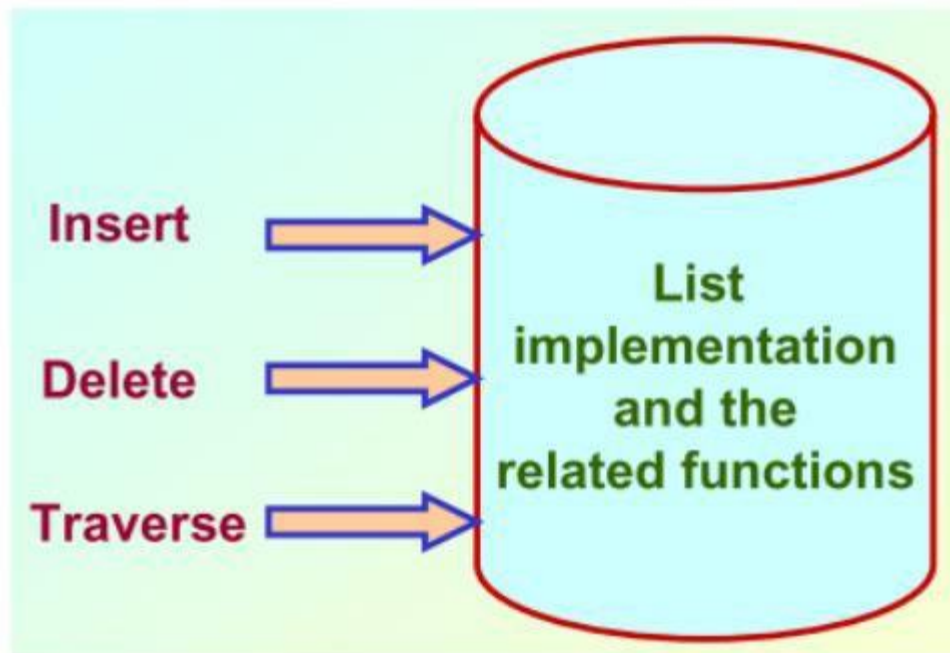


# Diziler ve Bağlı Listeler

- ▶ Diziler aşağıdaki hususlarda uygundur:
  - En sona eleman ekleme ve en sondan eleman silme
  - Rasgele her hangi bir elemana erişme
  - Belirli bir değer için arama yapma
- ▶ Bağlı listeler aşağıdaki hususlarda uygundur:
  - Eleman ekleme
  - Eleman silme
  - Sıralı erişim gerektiren uygulamalar
  - Eleman sayısının önceden tahmin edilemediği durumlar

# Bağlı Listeler

- ▶ Liste soyut bir veri tipidir.
- ▶ Bu veri tipi programcı tarafından tasarlanır.
- ▶ int, float gibi veri tiplerinden daha karmaşık yapıdadır.
- ▶ 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:

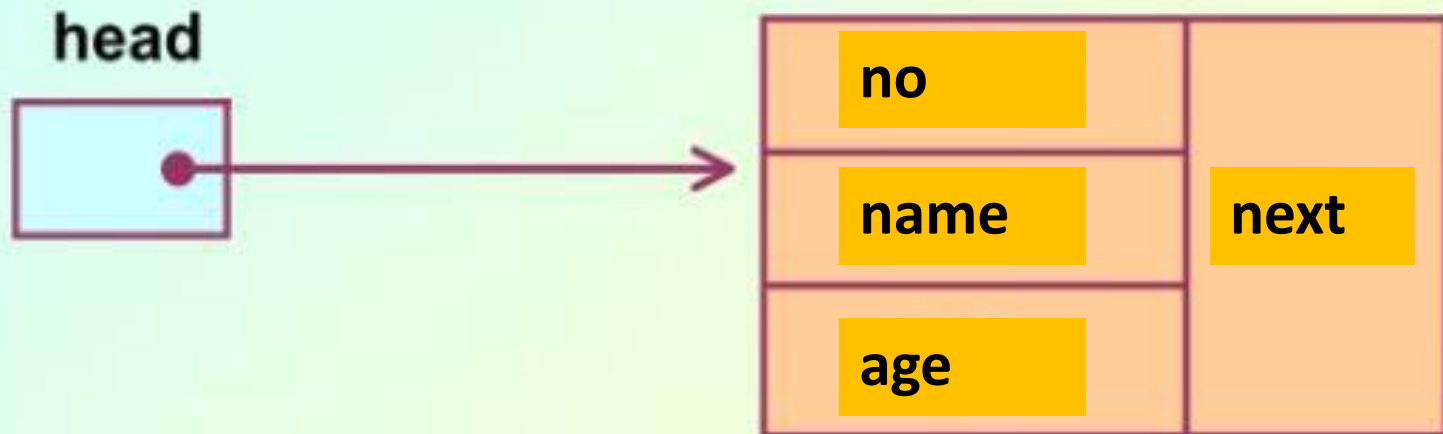
```
#include <stdio.h>
#include <stdlib.h>

struct student{
    int no;
    char name[40];
    int age;
    struct student *next;
};
typedef struct student node;
node *head,*newNode;
```

# Doğrusal Liste Oluşturma

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

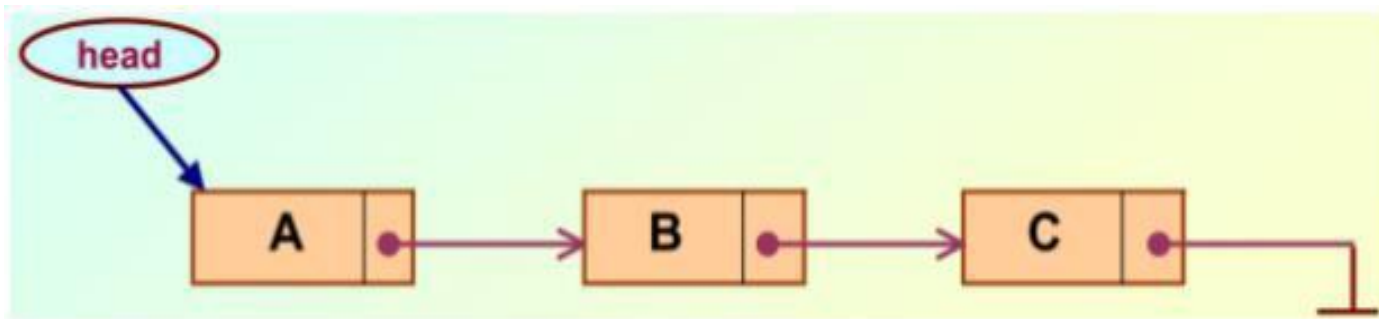
```
head = (node *) malloc(sizeof(node));
```





# 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

```
node* createList(){
    int n,k;
    node *head,*p;
    printf("How many students in the list?");
    scanf("%d",&n);
    for(k=0;k<n;k++){
        if(k==0){
            head = (node *)malloc(sizeof(node));
            p = head;
        }
        else{
            p->next = (node *) malloc(sizeof(node));
            p = p->next;
        }
        printf("Enter %d. student number: ",k+1); scanf("%d",&p->no);
        printf("Enter %d. student name: ",k+1); scanf("%s",p->name);
        printf("Enter %d. student age: ",k+1); scanf("%d",&p->age);
    }
    p->next = NULL;
    return head;
}
```

# Doğrusal Listede Dolaşma

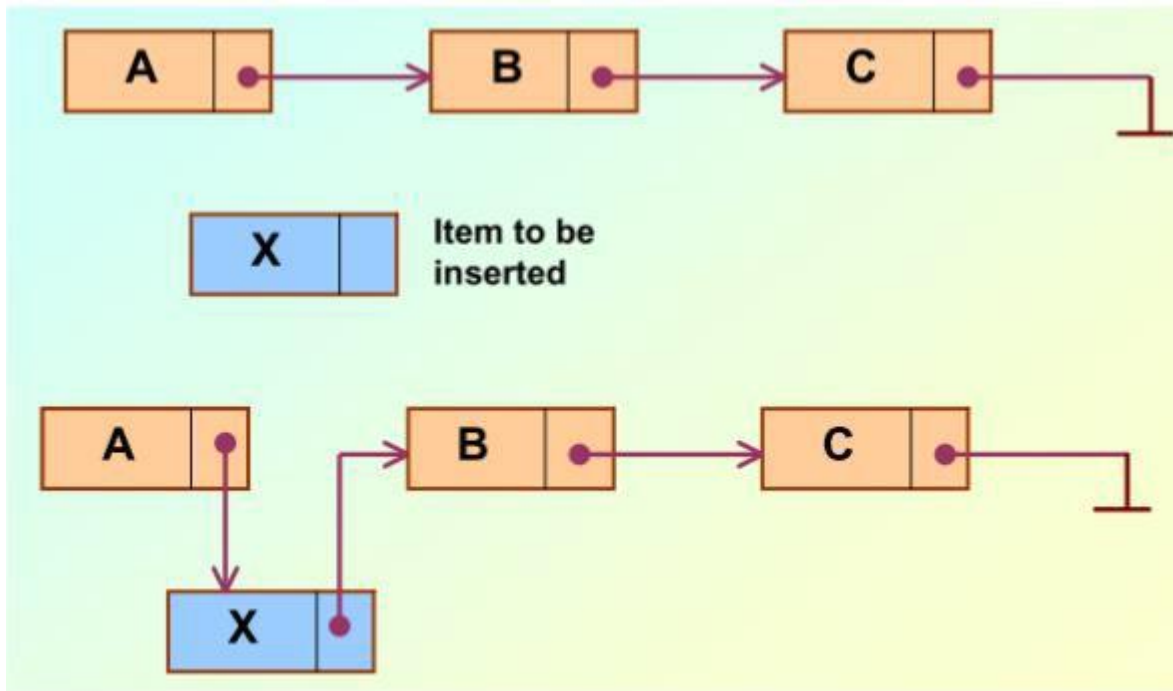
- ▶ Liste oluşturulduktan sonra ve head düğümü listenin ilk düğümünü gösterir hale gelince:
  - Pointer'ları takip et.
  - Düğümlere sıra geldikçe onların içeriklerini yazdır.
  - Sıradaki pointer NULL değeri gösteriyorsa dur.

```
void traverseList(node* head) {  
    int counter =1;  
    node *p;  
    p = head;  
    while (p!=NULL) {  
        printf("%d- %d %s %d \n", counter, p->no, p->name, p->age) ;  
        p = p->next;  
        counter++;  
    }  
}
```

# Listeye Dügüm 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» olabilir.

# Listeye Düğüm Ekleme



# Listeye Düğüm Ekleme

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

# Listeye Düğüm Ekleme

```
node* addNode(node* head){
    int stdNo;
    node *p, *q;
    node *newNode = (node *) malloc(sizeof(node));
    printf("Enter new student number: "); scanf("%d",&newNode->no);
    printf("Enter new student name: "); scanf("%s",newNode->name);
    printf("Enter new student age: "); scanf("%d",&newNode->age);

    printf("Enter std number that new record will be added before: \n");
    printf("Press 0 to add to the end of list\n");
    scanf("%d",&stdNo);

    p = head;
    if(p->no == stdNo){ //add to beginning
        .....
        newNode->next = p;
        head = newNode;
    }
}
```

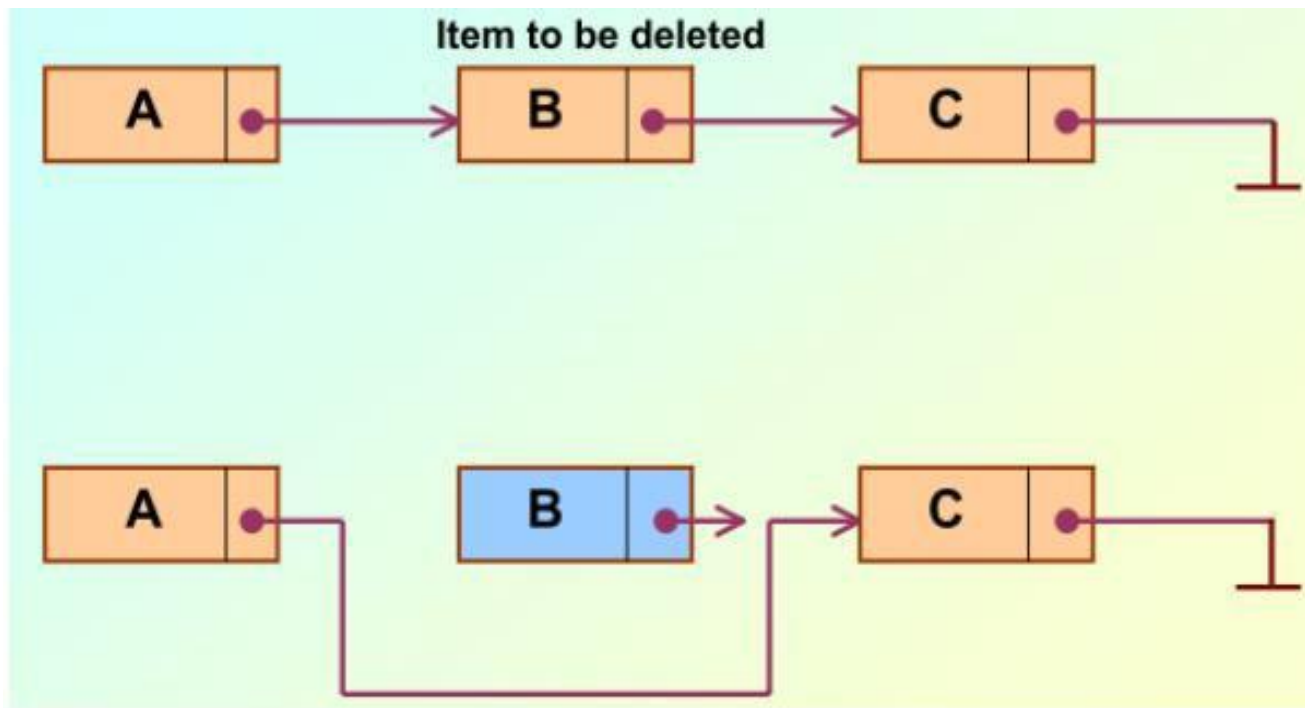
# Listeye Düğüm Ekleme

```
p = head;
if(p->no == stdNo){ //add to beginning
    newNode->next = p;
    head = newNode;
}
else{
    while(p->next != NULL && p->no != stdNo){
        q=p;
        p= p->next;
    }
    if(p->no == stdNo){ //Add nor beginning neither end
        q->next = newNode;
        newNode->next = p;
    }
    else if(p->next == NULL){ //Add to end
        p->next = newNode;
        newNode->next = NULL;
    }
}
return head;
}
```



# Listeden Dügüm 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

```
node* deleteNode(node *head) {
    int stdNo;
    node *p, *q;

    printf("Enter student number that will be deleted?");
    scanf("%d", &stdNo);

    p = head;
    if(p->no == stdNo) { //delete node at the beginning
        head = p->next;
        free(p);
    }
}
```

# Listeden Düğüm Silme

```
p = head;
if(p->no == stdNo){ //delete node at the beginning
    head = p->next;
    free(p);
}
else{
    while(p->next != NULL && p->no != stdNo){
        q=p;
        p= p->next;
    }
    if(p->no == stdNo){ //Delete from nor beginning neither end
        q->next = p->next;
        free(p);
    }
    else if(p->next == NULL){ //No node found to delete
        printf("No node found to delete");
    }
}
return head;
}
```

# Main Fonksiyonu

```
int main(void)
{
    node *head;
    int selection=0;
    printf("1.Create List 2.Traverse List 3.Add Node 4.Delete Node 5.Exit\r\n");
    while(1)
    {
        printf("Selection [1-5]?");
        scanf("%d",&selection);
        switch(selection)
        {
            case 1: head = createList();printf("Adres: %x\n",head);
                    traverseList(head);break;
            case 2: traverseList(head);break;

            case 3: head=addNode(head);
                    traverseList(head);break;
            case 4: head=deleteNode(head);
                    traverseList(head);break;
            case 5: exit(0);
        }
    }
}
```

# Tek Baęlı Liste Uygulaması-1

- Dıřarıdan aldıęı isimleri alfabetik sıra ile listeleyen, listeye yeni düęüm ekleyen, belirtilen düęümü silen ve düęümdeki en uzun ismi bulan tek baęlı liste uygulaması.

# Düğüm Yapısı

```
5 struct personel
6 {
7     char    adi[21];
8     struct personel *sonraki;
9 };
10
11 //artık struct personel yerine dugum kullanılacak
12 typedef struct personel dugum;
13 // *head dugum tipinde bir işaretçidir.
14 //Herzaman listenin başını gösterecek
15 dugum *head, *oncekiDugum,*yeni,*silinecek;
```

# Kayıt Arama

```
18 void ara(char *aranan) //listede arama yapar
19 {
20     dugum *p;
21     p = head;
22     oncekiDugum = head;
23     while(p->sonraki!=NULL)
24     {
25         p= p->sonraki;
26         if (strcmp(p->adi, aranan) >= 0) break;
27         oncekiDugum = p;
28     }
29 }
```



# Yeni Kayıt Ekleme

```
30 void ekle(char *s)
31 {
32     yeni = (dugum *) malloc(sizeof(dugum));
33     strcpy(yeni->adi, s);
34     ara(yeni->adi);
35     yeni->sonraki = oncekiDugum->sonraki; /* listeye ekle */
36     oncekiDugum->sonraki = yeni;
37 }
```

# Kayıt Silme

```
38 void sil(char *s)
39 {
40     ara(s);
41     silinecek = oncekiDugum->sonraki;
42     oncekiDugum->sonraki = silinecek->sonraki;
43     free(silinecek);
44 }
```

# Listeleme

```
45 void listele(void)
46 {
47     dugum *p;
48     p = head;
49     p = p->sonraki;
50     while (p!= NULL)
51     {
52         printf("%s \n",p->adi);
53         p = p->sonraki;
54     }
55 }
```

# En Uzun Kaydı Bulma

```
56 void enUzunBul(void)
57 {
58     dugum *p, *enuzun;
59     p = head;
60     p= p->sonraki;
61     enuzun=p;
62     while (p != NULL)
63     {
64         if (strlen(p->adi) >= strlen(enuzun->adi))
65         {
66             enuzun = p;
67         }
68         p = p->sonraki;
69     }
70     printf("\nEn uzun :%s  Uzunluk:%d",enuzun->adi,strlen(enuzun->adi));
71     getchar();
72 }
```

# Main Fonksiyonu

```
74 void main()
75 {
76     char    sec;
77     char    s[21];
78     head = (dugum *) malloc(sizeof(dugum));
79     strcpy(head->adi, " listenin basi");
80     head->sonraki = NULL;
81     do
82     {
83         system("cls");
84         listele();
85         printf("\n\n1 - Ekle\n2 - Sil\n3 - En Uzun isim\n4 - Cikis\n\nSec :");
86         sec = getche();
87         switch (sec)
88         {
89             case '1':printf("\nAdi :"); gets(s);
90                 ekle(s);break;
91             case '2':printf("\nAdi "); gets(s);
92                 sil(s);break;
93             case '3':enUzunBul();break;
94             case '4':exit(0);break;
95         }
96     }
97     while (1);
98 }
```

# Tek Baęlı Liste Uygulaması-2

► Öğrencilere ait numara, ad, vize ve final bilgilerini tutan bir tek baęlı doğrusal liste uygulaması için:

- Kayıt ekleme silme
- Öğrencileri başarı notlarını hesaplayarak (Vize %40, Final %60) listeleme
- Sınıfın en yüksek başarı notuna sahip öğrenci bilgilerini listeleme
- Sınıfın başarı notu ortalamasını hesaplama

# Düğüm Yapısı

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 struct ogrenci{
5     int no;
6     char adi[40];
7     int vize,final;
8     struct ogrenci *sonraki;
9 };
10 typedef struct ogrenci dugum;
11 dugum *head,*yeniDugum,*oncekiDugum,*enBasarili;
12
```

# Liste Oluşturma

```
13 void ogrenciListeOlustur()
14 {
15     int n,k;
16     printf("Kac elamanli liste olusturacaksin");
17     scanf("%d",&n);
18     for(k=0;k<n;k++)
19     {
20         if(k==0) //ilk düğüm ekleniyor
21         {
22             yeniDugum = (dugum *)malloc(sizeof(dugum));
23             head = yeniDugum;
24         }
25         else
26         {
27             yeniDugum->sonraki = (dugum *) malloc(sizeof(dugum));
28             yeniDugum = yeniDugum->sonraki;
29         }
30         printf("Ogrenci No:");scanf("%d",&yeniDugum->no);
31         printf("Ogrenci Adi:");scanf("%s",&yeniDugum->adi);
32         printf("Ogrenci Vize:");scanf("%d",&yeniDugum->vize);
33         printf("Ogrenci Final:");scanf("%d",&yeniDugum->final);
34     }
35     yeniDugum->sonraki = NULL;
36 }
```



# Öğrenci Kaydı Listeleme

```
38 void ogrenciListele()  
39 {  
40     double donemNotu =0;  
41     dugum *p;  
42     p = head;  
43     while(p!=NULL)  
44     {  
45         printf("\n ***** \n");  
46         printf("Ogrenci No:%d\t",p->no);  
47         printf("Ogrenci Adi:%s\t",p->adi);  
48         printf("Ogrenci Vize:%d\t",p->vize);  
49         printf("Ogrenci Final:%d\t",p->final);  
50         donemNotu = p->vize*0.4+p->final*0.6;  
51         printf("Donem Notu : %.2f\t", donemNotu);  
52         printf("\n ***** \n");  
53         p = p->sonraki;  
54     }  
55 }
```

# Öğrenci Kaydı Ekleme

```
57 void ogrenciEkle()
58 {
59     int kayitNo;
60     dugum *p, *q;
61     yeniDugum = (dugum *) malloc(sizeof(dugum));
62     printf("Ogrenci No:");scanf("%d",&yeniDugum->no);
63     printf("Ogrenci Adi:");scanf("%s",yeniDugum->adi);
64     printf("Ogrenci Vize:");scanf("%d",&yeniDugum->vize);
65     printf("Ogrenci Final:");scanf("%d",&yeniDugum->final);
66
67     printf("Hangi kayittan oncesine eklemek istiyorsunuz");
68     printf("\nListe sonuna eklemek icin 0 gir");
69     scanf("%d",&kayitNo);
70
71     p = head;
72     if(p->no == kayitNo) //başa ekle
73     {
74         yeniDugum->sonraki = p;
75         head = yeniDugum;
76     }
```

# Öğrenci Kaydı Ekleme

```
80     else
81     {
82         while(p->sonraki !=NULL)
83         {
84             q=p;
85             p=p->sonraki;
86             if(p->no == kayitNo) break;
87         }
88         if(p->no == kayitNo) //araya ekleme
89         {
90             q->sonraki = yeni;
91             yeni->sonraki = p;
92         }
93         else if(p->sonraki == NULL)
94         {
95             p->sonraki = yeni;
96             yeni->sonraki = NULL;
97         }
98     }
99 }
```

# Öğrenci Kaydı Silme

```
97 void ogrenciSil()  
98 {  
99     int kayitNo;  
100     dugum *p, *q;  
101  
102     printf("Silmek istediginiz ogrenci no gir");  
103     scanf("%d",&kayitNo);  
104  
105     p = head;  
106     if(p->no == kayitNo) //baştakini sil  
107     {  
108         head = p->sonraki;  
109         free(p);  
110     }
```

# Öğrenci Kaydı Silme

```
115     else
116     {
117         while(p->sonraki !=NULL)
118         {
119             if(p->no == kayitNo) break;
120             else {
121                 q=p;
122                 p=p->sonraki;
123             }
124         }
125         if(p->no == kayitNo) //araya ekleme
126         {
127             q->sonraki = p->sonraki;
128             free(p);
129         }
130         else if(p->sonraki == NULL)
131         {
132             printf("Silinecek ogrenci no yok\n");
133         }
134     }
135 }
```

# En Başarılı Öğrenci Bulma

```
130 double basariNotuHesapla(int vize,int final)
131 {
132     return (0.4*vize)+(0.6*final);
133 }
134
135 void enBasariliOgrenci()
136 {
137     double basariNotu = 0;
138
139     dugum *p;
140     p = head;
141     enBasarili = head;
142
143     while(p->sonraki!=NULL)
144     {
145         p=p->sonraki;
146         if(basariNotuHesapla(p->vize,p->final)>basariNotuHesapla(enBasarili->vize,enBasarili->final))
147             enBasarili = p;
148     }
149     printf("En basarili ogrenci:\n");
150     printf("No:%d - Ad: %s Basari Notu:%.2f\n",enBasarili->no,
151         enBasarili->adi, basariNotuHesapla(enBasarili->vize,enBasarili->final));
152 }
```

# Sınıf Ortalaması Hesaplama

```
154 void sinifBasariOrtalamasi()  
155 {  
156     double basariNotuOrtalamasi = 0;  
157     double toplam =0;  
158     int sayac = 0;  
159     dugum *p;  
160     p = head;  
161     if(p==NULL)  
162     {  
163         printf("Listede kayit yok!");  
164     }  
165     else  
166     {  
167         do  
168         {  
169             toplam += basariNotuHesapla(p->vize,p->final);  
170             p=p->sonraki;  
171             sayac++;  
172         }  
173         while(p!=NULL);  
174  
175         basariNotuOrtalamasi = toplam/sayac;  
176         printf("Basari Notu Ortalamasi: %.2f \n",basariNotuOrtalamasi);  
177     }  
178 }
```

# Main Fonksiyonu

```
180 int main(void)
181 {
182     int secim=0;
183     printf("1-Liste Olustur \n2-Yeni Kayit Ekle \n3-Kayit Sil
184     while(1)
185         {
186             printf("Secim yap [1-5]?");
187             scanf("%d",&secim);
188             switch(secim)
189             {
190                 case 1: ogrenciListeOlustur();
191                     ogrenciListele();break;
192                 case 2: ogrenciEkle();ogrenciListele();break;
193                 case 3: ogrenciSil();ogrenciListele();break;
194                 case 4: enBasariliOgrenci();break;
195                 case 5: sinifBasariOrtalamasi();break;
196             }
197     }
198 }
```



# Kaynaklar

Doç. Dr. Caner ÖZCAN, KBÜ Yazılım Mühendisliği [www.canerozcan.net](http://www.canerozcan.net)\*

- ▶ Doç. Dr. Fahri Vatansever, “Algoritma Geliştirme ve Programlamaya Giriş”, Seçkin Yayıncılık, 12. Baskı, 2015.
- ▶ Kaan Aslan, “A’dan Z’ye C Klavuzu 8. Basım”, Pusula Yayıncılık, 2002.
- ▶ Paul J. Deitel, “C How to Program”, Harvey Deitel.
- ▶ “A book on C”, All Kelley, İra Pohl

\* Bu dersin slaytları genelde bu kaynaktan türetilmiştir.

# Kaynaklar

Ders içerikleri ve duyurular için dersin web sitesine aşağıdaki adresten ulaşabilirsiniz.

[hru-algpro.github.io](https://hru-algpro.github.io)

S o r u l a r

?



Dinlediğiniz için teşekkürler.