BLM102

PROGRAMLAMA DİLLERİ II

Yrd. Doç. Dr. Baha ŞEN

baha.sen@karabuk.edu.tr

KBUZEM
Karabük Üniversitesi
Uzaktan Eğitim Araştırma ve Uygulama Merkezi

1. Listeler

Konunun daha rahat anlaşılması için günlük hayattan bir örnek verilebilir. Adlarının alfabetik sırasında göre duran ve bir elleri ile birbirini işaret eden askerler düşünelim.

Bu askerlerin adları, görevleri ve yaşlarını bir yapı(struct) dizi içerisinde saklayabiliriz. Ancak asker sayımız dinamik olarak değişkenlik göstereceğinden sabit bir dizi tanımlamamız uygun olmaz. Bunun yanısıra alfabetik sıraya göre dizildikleri için araya yeni asker eklenmesi ya da birisinin eksilmesi ayrı bir problem olacaktır.

Bunun gibi problemleri aşmak için liste kavramı ortaya çıkmıştır. Bu yeni yaklaşıma göre bir yapı tanımlanır ve yapının içerisinde en az bir tane kendi tipinde bir yapının adresini saklayabilen bir işaretçi olmalıdır. Örneğimizde askerlerin kolları işaretçi alanını, isimleri, görevleri ve yaşları ise veri alanını göstermektedir.

Dizili askerlere ait bilgiler her sorgulandığında önce listenin başındaki askere gidilir, ona ait bilgiler alındıktan sonra kolunun işaret ettiği askere geçilir. Bu işlem hiçbir asker işaret etmeyen asker bulununcaya kadar devam eder. Eğer yeni bir asker gelirse uygun araya sıkıştırılır, birisi giderse de gidenin gösterdiği asker gideni gösteren tarafından gösterilir.

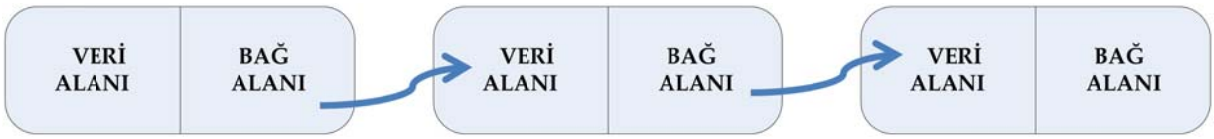
1.1. Liste Çeşitleri

Listeler işaret etme şekline ve sonlanma durumuna göre aşağıdaki gibi sınıflara ayrılabilir:

- Tek bağlı sıralı doğrusal listeler
- Çift bağlı sıralı doğrusal listeler
- Tek bağlı sıralı doğrusal listeler
- Çift bağlı sıralı doğrusal listeler

1.2. Tek Bağlı Sıralı Doğrusal Listeler

Bu veri yapısı iki ayrı alandan oluşur. Veri alanı ve bağ alanı. Bu yapıya listenin bir düğümü(node) adı verilir.



Veri alanı: Bu alanda bilgiler saklanır.

Bağ alanı: Bu alanda bir sonraki düğümün başlangıç adresi saklanır.

Listede bulunan elemanlar fiziksel olarak sıralı olmak durumunda değildir. Verilecek örnekteki liste, alfabetik karakterleri sıralı olarak saklayan bir listedir. Listede alfabetik olarak en küçük olan elemanın adresi bilinir. Liste başı işaretlendikten sonra büyüklük sırasında bir sonraki elemanların adresleri bağ alanlarına yazılarak liste sıralanır. En son elemanın bağ alanında sıfır bulunur, bu listenin bittiğini gösterir.

Örnek:

Adres	Veri Alanı	Bağ Alanı
1	H	0
2		Liste sonu
3	C	4
4	E	8
5		
6		
7	B	3
8	F	1

Liste başı →

1.2.1. Tek Bağlı Sıralı Doğrusal Listeleye Düğüm Ekleme

Sıralı bir listeye eleman eklemek için önce liste tanımlanır. Bunun için listede tutulacak verinin türü ve listenin eleman sayısı verilir. Önceki örnekte verilen listeyi ele alarak bu listeye 'G' harfini eklemek isteyelim. Önce listede bu veri için boş bir alan bulunur ve bağ alanlarında güncelleme yapılır. 'G' verisini 6.sıraya yazarsak 'F' nin bağı 'G' yi, 'G' nin bağı da 'H' yi gösterecek şekilde bağ alanları değiştirilir.

Örnek:

Adres	Veri Alanı	Bağ Alanı
1	H	0
2		Liste sonu
3	C	4
4	E	8
5		
6	G	1
7	B	3
8	F	~~1~~ 6

Liste başı →

1.2.2. Tek Bağlı Sıralı Doğrusal Listeden Düğüm Silme

Önce listeden çıkarılmak istenen eleman bulunur. Eğer silinecek eleman listede yoksa bu durum bir mesaj ile bildirilir. Eleman bulunduğunda bağ alanlarında güncelleme yapılarak eleman listeden silinir. Örneğimizdeki 'C' verisini silmek isteyelim. 'C' elemanı bulunur ve listede 'C' elemanını gösteren 'B' nin bağ alanına 4 yazılır('E' nin adresi). Son işlemde de 'C' nin bağ alanına sıfır yazmak gerekir. Liste başından itibaren sıralandığında 'C' listede yer almaz.

Örnek:

	Adres	Veri Alanı	Bağ Alanı
	1	H	0
	2		Liste sonu
	3	C	~~4~~ 0
	4	E	8
	5		
	6	G	1
Liste başı →	7	B	~~3~~ 4
	8	F	6

Örnek: Dışardan aldığı isimleri alfabetik sıra ile listeleyen, listeye yeni düğüm ekleyen, belirtilen düğümü silen ve düğümdeki en uzun isimi bulan programı yazınız.

```
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include <conio.h>
struct listyapi
{
  char   adi[21];
  struct listyapi *sonraki;
};
typedef struct listyapi listnode; //artık listyapi
yerine listnode kullanılacak
typedef listnode *listptr; // *listptr listnode
tipinde bir işaretçidir.
listptr headnode;   /* Herzaman listenin basini
gosterir */
void seeklist(char *searchthis, listptr *prevnode)
//listede arama yapar
{
  listptr c;
  c = headnode;
  *prevnode = c;
  while ((c->sonraki != NULL))
    {
      c = c->sonraki;
      if (strcmp(c->adi, searchthis) >= 0) break;
      *prevnode = c;
    }
}
void kayit(char *s)
/* prevnode kayidi newnode kayidini, newnode kayidi
prevnode'nin daha once gosterdigi kayidini gosterir.
*/
{
  listptr newnode, prevnode;
  newnode = (listptr) malloc(sizeof(listnode)); /*
yeni kayida yer a‡ */
```

```
  strcpy(newnode->adi, s);             /*bilgiyi yeni
kayida yaz */
  seeklist(newnode->adi, &prevnode);  /* listedeki
yerini bul */
  newnode->sonraki = prevnode->sonraki;  /* listeye
ekle */
  prevnode->sonraki = newnode;
}
void iptal(char *s)
/* newnode kayidi silinir. prevnode kayidi newnode
kayidinin gosterdigi kayidini gosterir. */
{
  listptr newnode, prevnode;
  seeklist(s, &prevnode);
  newnode = prevnode->sonraki;
  prevnode->sonraki = newnode->sonraki;
  free(newnode);
}
void listlist(void)
{
  listptr currentnode;
  currentnode = headnode;
  if (currentnode != NULL) currentnode = currentnode-
>sonraki;
  while (currentnode != NULL)
    {
      printf("%s  ",currentnode->adi);
      currentnode = currentnode->sonraki;
    }
  printf("\n");
}
void ebUz(void)
{
  int ln, cnt;
  listptr currentnode, findnode;
  currentnode = headnode;
  if (currentnode != NULL)
    {
      currentnode = currentnode->sonraki;
      ln=0;
      cnt=0;
    }
  findnode=currentnode;
  while (currentnode != NULL)
```

```
    {
      cnt++;
      if (strlen(currentnode->adi) >=
strlen(findnode->adi))
        {
          findnode = currentnode;
          ln=cnt;
        }
      currentnode = currentnode->sonraki;
    }
  printf("%s  %d  %d",findnode->adi, ln, cnt);
  getch();
}   /* En uzun isim; */
void main()
{
  char      sec;
  char      *s;
  headnode = (listptr) malloc(sizeof(*headnode));
  strcpy(headnode->adi," listenin basi");
  headnode->sonraki = NULL;
  do
    {
      system("cls");
      listlist();
      printf("\n\n1 - Giris\n2 - iptal\n3 - En Uzun
isim\n4 - Son\n\nSec :");
      sec = getche();
      switch (sec)
        {
        case '1':
          printf("\nAdi :");
          gets(s);
          kayit(s);
          break;
        case '2':
          printf("Adi \n");
          gets(s);
          iptal(s);
          break;
        case '3':
          ebUz();
          break;
        case '4':
          exit(0);
```

```
            break;
          }
      }
    while (1);
}
```

1.3. Çift Bağlı Sıralı Doğrusal Listeler

Bu veri yapısında tek bağlı sıralı doğrusal listeden farklı olarak önceki düğümün de adresini saklayan bir işaretçi bulunmasıdır. Bu sayede listede iki yönlü ilerleme yeteneği sağlanmış olur.