

**YILDIZ TEKNİK ÜNİVERSİTESİ**

**ELEKTRİK-ELEKTRONİK FAKÜLTESİ**

**BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ**

2017-2018 Bahar Yarıyılı

VERİ YAPILARI VE ALGORİTMALAR

Prof. Yahya KARSLIGİL

1.ÖDEV

**LİNKLİ LİSTELER**

**AD-SOYAD=ÖZGÜR KAN**

**NUMARA=15011702**

**GRUP 1**

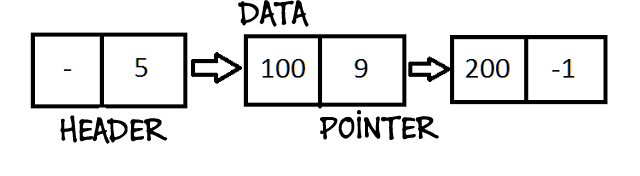
**LİNKLİ LİSTELER**

**Diziler ve linkli listeler bilgisayar bilimlerinde çeşitli etkenlere bağlı olarak sıkça kullanılan veri yapılarından iki tanesidir. Linkli listeleri daha iyi anlamak için dizileri ve çalışma mantığını kısaca bir özetleyelim.**

**Dizileri kısaca tanımlayacak olursak, aynı tip verilerin birbiri arkasına bir küme içinde tanımlanmasıdır diyebiliriz. Aynı tipte verilere tek bir isimle ulaşmak için kullanılır. Diziler tek boyutlu veya çok boyutlu olabilir. Dizinin elemanları bellekte peş peşe saklanır. Eğer siz verileri sıralı bir şekilde girmediyseniz veya daha sonradan sıralamadıysanız dizinin elemanları karışık olacaktır. Bu durum milyonlarca veri bulunan bir yapıda oldukça sıkıntı çıkaracaktır. Diziler statik ve dinamik olmak üzere iki farklı şekilde tanımlanabilir. Programın başında dizinin boyutu ve eleman sayısı belirtilirse derleyici o dizi için gereken bellek alanını program sonlanıncaya kadar saklı tutar ve bu alan başka bir amaç için kullanılamaz. Programın çalışması esnasında dizinin boyutu ve eleman sayısı değiştirilemez ve aşılamaz. Bu tür diziler statik dizi olarak adlandırılır. Program çalışırken bir dizinin boyutu ve eleman sayısı bazı yöntemlerle değiştirilebilir bu tür dizilere dinamik dizi denir. Aslında linkli listelerde çok boyutlu ve dinamik bir yapı olarak tanımlanabilir dersek yanlış olmaz. Dizilerin mantığını ve yapısını kısaca bir özetlemiş olduk şimdi asıl konumuz olan bağlı listelere geçelim.**

**Linkli listeler diğer bir deyişle bağlı listeler olarak adlandırılır. Aslında burada bizim için önemli olan bağlı kelimesidir. Kafamızda şöyle canlandırırsak her bir eleman diğer bir elemana ip yardımıyla bağlıymış gibi hayal edebiliriz. Diğer bir deyişle her bir eleman kendisinden sonra gelen elemanın adresini yani pointerınıda içerisinde saklar.**

**Burada akıllara şu soru gelir başlangıç elemanının adresini kim tutacak? Bunun basit bir çözümü vardır. İlk başta bağlı listenin ilk elemanını gösteren bir pointer tanımlanır. Bu pointer header olarak adlandırılır. Sonuncu eleman tail olarak adlandırılır. Listede şu anda bilgilerine ulaşılan aktif elemana current denir. Header ile ilk elemana ulaşıldıktan sonra diğer elemanlara da ulaşılmış olur. Bağlı listenin son elemanını belirtmek için ise pointerda NULL veya -1 değeri tutulur. Bağlı listenin daha iyi anlaşılabilmesi için aşağıda basit bir bağlı liste çizilmiştir.**

****

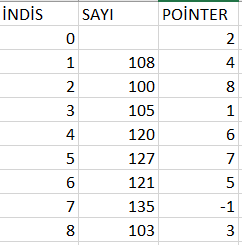
**LİNKLİ LİSTELER**

**AVANTAJLARI VE DEZAVANTAJLARI**

**Bağlı listelerde her bir eleman kendisinden sonra gelen elemanın adresini tuttu fakat bu bize ne kazandıracak sorusu hemen akıllara gelir. Örnekte de görüldüğü gibi bellekte karışık olan verilerimizi basit olarak sıralamış olduk. Bu sıralama işlemini bellekteki yerlerini değiştirmeden yaptık. Bu da bize önemli ölçüde hız kazandırmış oldu. Bağlı listenin bir diğer önemli özelliği ise eleman ekleme ve silme işlemlerinin çok kolay olmasıdır. Örneğin 100 elemanlık bir dizide 50. elemanı silmek istersek ve bağlı liste kullanmazsak geriye kalan 49 elemanın indisini birer kaydırmamız gerekecek ve daha büyük dizilerde bu işlem içinden çıkılmaz bir hal alacaktır. Fakat bağlı listede 50. elemandan bir önceki elemanın pointer adresini 50 değil de 51 yaparsak ve 50. elemanın pointerını -9 yaparsak otomatik olarak 50. Elemana ulaşamayacağız yani silinmiş olarak kabul edeceğiz. Bu durum ekleme işlemi içinde aynıdır. Bağlı listelerin bir diğer özelliği de dinamik olmasıdır. Bu da eleman eklerken bize sorun yaratmaması anlamına gelir. Bağlı listelerde dizilerdeki gibi gereksiz alan kullanımı yoktur.**

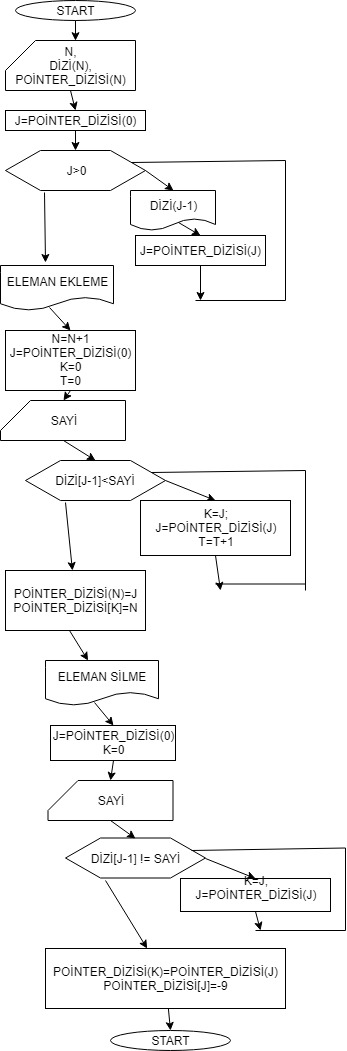
**Bağlı listelerin avantajları gibi dezavantajları da vardır. Bağlı listeler daha fazla alan kullanır. Bunun nedeni her eleman ek olarak birde kendisinden sonra gelen elemanın pointer adresini tutar. Diğer bir dezavantaj ise rastgele erişimin maliyetli olmasıdır. Bir dizide 10. İndise direk ulaşabilirken bağlı listelerde aradaki bütün düğümleri dolaşmamız gerekir. Bu da zaman ve maliyeti arttırır.**

**Örnek bir linkli liste tablosu**

****

**LİNKLİ LİSTE ÖRNEK**

**Örneğin elimizde 10 adet sayı bulunduğunu düşünelim ve bu 10 adet sayıyı karışık bir şekilde bir diziye yerleştirelim. Daha sonra linkli listeler yardımıyla bu sayıları sıralayalım, yeni sayı ekleyelim ve bir sayıyı silelim.**



**Program Kodları**

**#include <stdio.h>**

**#include <stdlib.h>**

**int main()**

**{**

**int Eleman\_Sayisi;**

**int sayi,i,j,secim=0;**

**printf("Eleman sayisii giriniz:");**

**scanf("%d",&Eleman\_Sayisi);**

**int \*dizi=(int\*) malloc(Eleman\_Sayisi\*sizeof(int));**

**int \*Pointer\_dizisi=(int\*) malloc((Eleman\_Sayisi+1)\*sizeof(int));**

**printf("Header degerini giriniz:");**

**scanf("%d",&sayi);**

**while(sayi>Eleman\_Sayisi || sayi<=-2){**

**printf("Header degerini eleman sayisindan buyuk ve -2 dem kucuk bir deger olamaz yeniden giriniz:");**

**scanf("%d",&sayi);**

**}**

**\*Pointer\_dizisi=sayi;**

**Pointer\_dizisi++;**

**printf("Ana diziyi ve Pointer dizisini giriniz:\n");**

**for(i=0;i<Eleman\_Sayisi;i++){**

**printf("dizinin %d elemanini giriniz:",i+1);**

**scanf("%d",&sayi);**

**\*dizi=sayi;**

**dizi++;**

**printf("%d elemanın point ettigi eleman:",i+1);**

**scanf("%d",&sayi);**

**while(sayi>Eleman\_Sayisi || sayi<=-2){**

**printf("Pointer degeri eleman sayisindan buyuk ve -2 den kucuk bir deger olamaz yeniden giriniz:");**

**scanf("%d",&sayi);**

**}**

**\*Pointer\_dizisi=sayi;**

**Pointer\_dizisi++;**

**}**

**dizi=dizi-Eleman\_Sayisi;**

**printf("Girilen dizi :| ");**

**for(i=0;i<Eleman\_Sayisi;i++){**

**printf("|%d",\*dizi);**

**dizi++;**

**}**

**printf("|\n");**

**Pointer\_dizisi=Pointer\_dizisi-Eleman\_Sayisi-1;**

**printf("Pointer dizisi:");**

**for(i=0;i<=Eleman\_Sayisi;i++){**

**printf("|%d ",\*Pointer\_dizisi);**

**Pointer\_dizisi++;**

**}**

**printf("|");**

**Pointer\_dizisi=Pointer\_dizisi-Eleman\_Sayisi-1;**

**dizi=dizi-Eleman\_Sayisi;**

**printf("\nislem seciniz \n1-)Diziyi Siralama \n2-)Diziye Eleman Ekleme \n3-)Diziden Eleman Silme \nCikis icin 9'a basiniz...");**

**scanf("%d",&secim);**

**while(secim>0){**

**if(secim==1){**

**printf("\n\n Dizi siralandi==>");**

**j=\*Pointer\_dizisi;**

**while(j>0){**

**printf("|%d",dizi[j-1]);**

**j=Pointer\_dizisi[j];**

**}**

**}**

**else if(secim==2){**

**Eleman\_Sayisi++;**

**dizi = realloc(dizi,sizeof(int)\*Eleman\_Sayisi);**

**Pointer\_dizisi=realloc(Pointer\_dizisi,sizeof(int)\*(Eleman\_Sayisi+1));**

**printf("dizinin %d elemanini giriniz:",Eleman\_Sayisi);**

**scanf("%d",&sayi);**

**dizi[Eleman\_Sayisi-1]=sayi;**

**j=\*Pointer\_dizisi;**

**int t=0,k=0;**

**while(dizi[j-1]<sayi){**

**k=j;**

**j=Pointer\_dizisi[j];**

**t++;**

**}**

**printf("%d elemanın point ettigi eleman:%d oldu",i,j);**

**Pointer\_dizisi[Eleman\_Sayisi]=j;**

**Pointer\_dizisi[k]=Eleman\_Sayisi;**

**printf("eleman eklendi...");**

**printf("\n Girilen dizi :| ");**

**for(i=0;i<Eleman\_Sayisi;i++){**

**printf("|%d",dizi[i]);**

**}**

**printf("\n Pointer dizisi:");**

**for(i=0;i<Eleman\_Sayisi+1;i++){**

**printf("|%d ",Pointer\_dizisi[i]);**

**}**

**}**

**else if(secim==3){**

**printf("\n\nSilmek istediğiniz elemani giriniz :");**

**scanf("%d",&sayi);**

**j=\*Pointer\_dizisi;**

**int k=0;**

**while(dizi[j-1]!=sayi){**

**k=j;**

**j=Pointer\_dizisi[j];**

**}**

**Pointer\_dizisi[k]=Pointer\_dizisi[j];**

**Pointer\_dizisi[j]=-9;**

**printf("eleman silindi...");**

**printf("\n Girilen dizi :| ");**

**for(i=0;i<Eleman\_Sayisi;i++){**

**printf("|%d",dizi[i]);**

**}**

**printf("\n Pointer dizisi:");**

**for(i=0;i<Eleman\_Sayisi+1;i++){**

**printf("|%d ",Pointer\_dizisi[i]);**

**}**

**}**

**else if(secim==9){**

**exit(0);**

**}**

**printf("\nislem seciniz \n1-)Diziyi Siralama \n2-)Diziye Eleman Ekleme \n3-)Diziden Eleman Silme \nCikis icin 9'a basiniz...");**

**scanf("%d",&secim);**

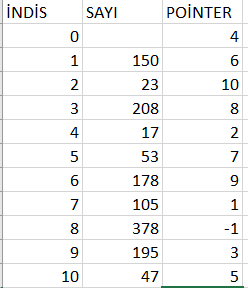
**}**

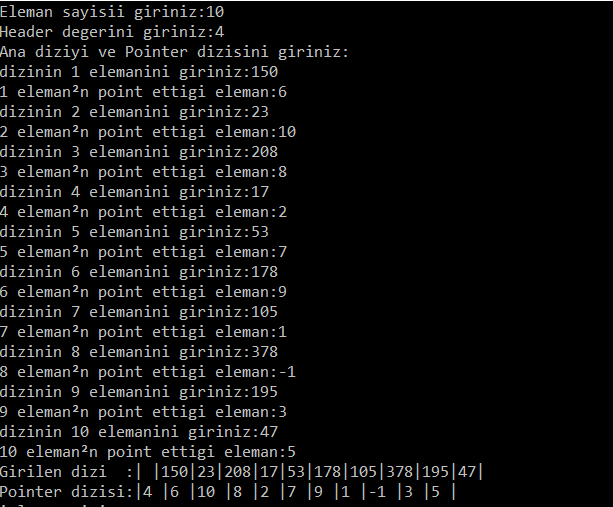
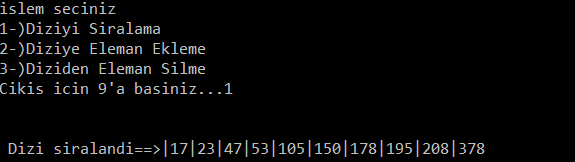
**return 0;**

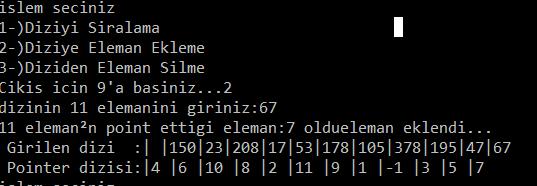
**}**

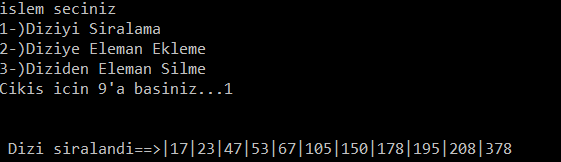
**Program çıktısı**

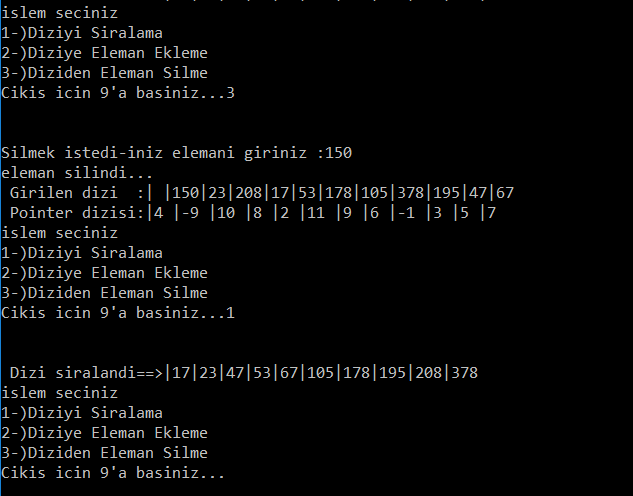
**Örnekte kullanılan linkli liste**

****

****

****

****

****

**Kaynaklar ve Referanslar:**

**http://www.yazilimcilardunyasi.com/2017/03/statik-dizi-ve-dinamik-dizi.html**

**http://www.algoritmauzmani.com/veri-yapilari/bagli-liste-nedir-linked-list-nedir/**

**https://www.youtube.com/watch?v=-5y2ArGzX-I&index=3&list=PLDUOF2Be-kzlga9FbVXHgRCJ41E9yPq9Z**

**https://www.youtube.com/watch?v=r3uOBb3BM-0&list=PLh9ECzBB8tJN9bckI6FbWB03HkmogKrFT**

**https://www.hackerrank.com/domains/data-structures?filters%5Bsubdomains%5D%5B%5D=linked-lists**

**https://www.youtube.com/watch?v=njTh\_OwMljA**