T.C.
SAKARYA ÜNİVERSİTESİ

BİLGİSAYAR VE BİLİŞİM BİLİMLERİ FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

VERİ YAPILARI ÖDEV RAPORU

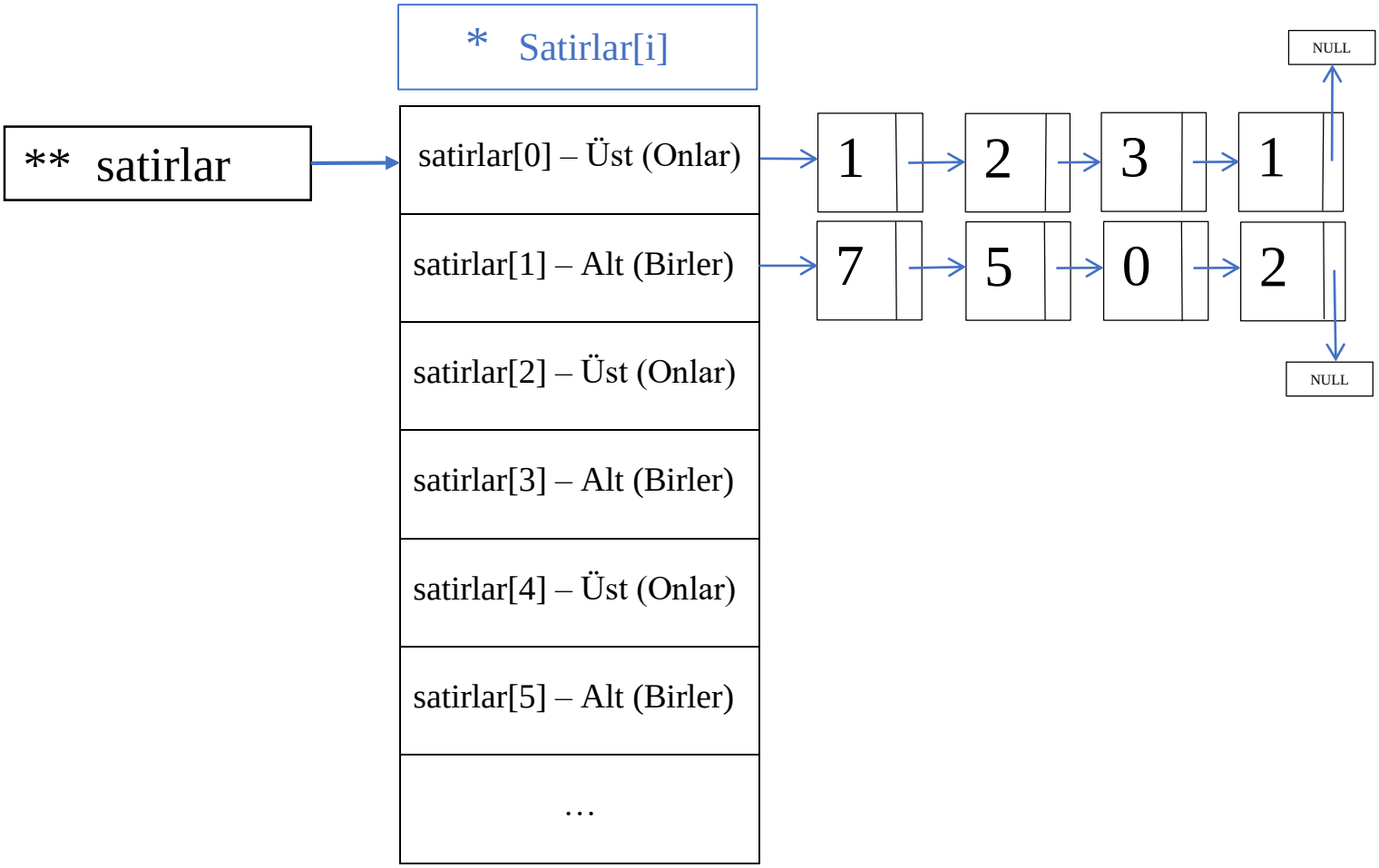
1.ÖDEV

Sinem Şaziye Karaca
G211210025

SAKARYA

Temmuz, 2023

Veri Yapıları Dersi

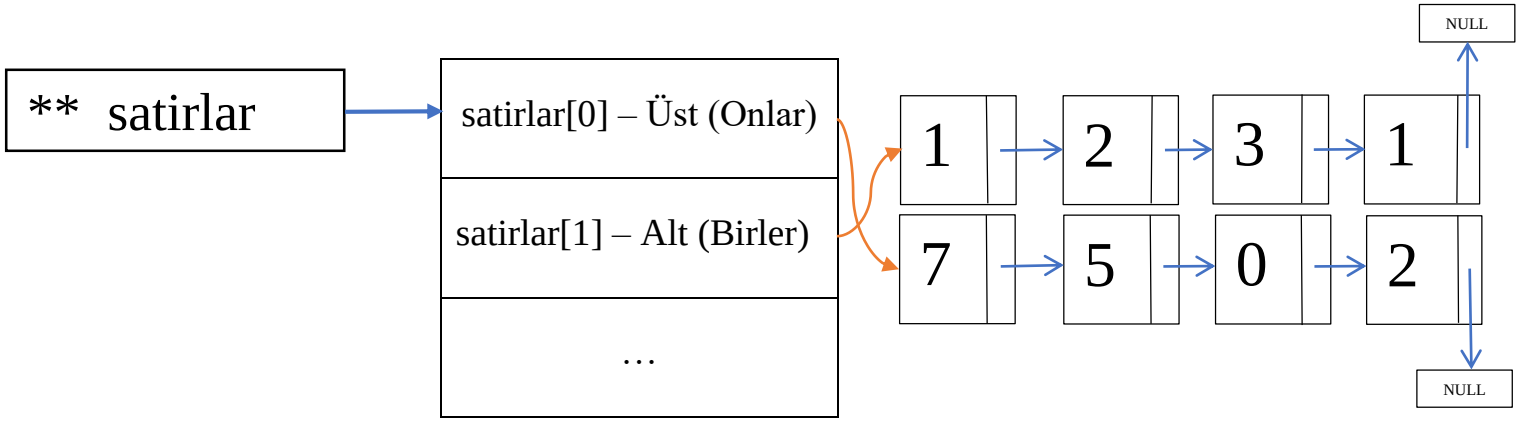


Şekil 1. Programın Genel Tasarımı

Verilen ödevde, içinde iki basamaklı sayıların bulunduğu Sayilar.txt dosyasını okuyan, okuduğu sayıların onlar ve birler basamaklarındaki rakamları onlar basamağı üst, birler basamağı alt olacak şekilde ayrı ayrı tek yönlü bağlı listelerde tutan bir program oluşturmamız isteniyor. Dosya okunup gerekli yerleştirme işlemleri yapıldıktan sonra kullanıcıdan alınan alt ve üst konum bilgilerine göre seçilen bir alt ve bir üst bağlı listenin kendi aralarında yerlerini değiştirmek programın ilk görevi. Yer değiştirme işleminin ardından programın gerçekleştirmesi beklenen ikinci görev ise alt ve üst listelerin, kendi aralarında olacak şekilde, i. düğümlerinin ortalamalarını alıp birbirleriyle toplayarak ortalamalarını hesaplamak. Hesaplanan ortalama değerler virgülden sonra bir basamak gösterecek şekilde konsol ekranında kullanıcıya sunulacak.

Verilen problemin çözümünde ilk adım Şekil 1'de şematize ettiğim yapının tasarımını yapmak oldu. Tasarladığım bu görsel yapının programı gerçekleme sürecinde bana çok faydası oldu. Bu tasarımı gerçeklemeye Sayilar.txt dosyasını okuyup dosyadan alınan veriler üzerinde istenen işlemleri yapacak olan DosyaOku sınıfını oluşturarak başladım. Oluşturduğum bu sınıfın kurucu fonksiyonu, ilk olarak dosyanın satır sayısını bulmayla başlıyor. Bulduğu satır sayısı program boyunca özellikle döngülerin tekrar sayılarını belirlerken çok sık kullanılıyor. Satır sayısının bulunmasının ardından Run Time Stack bellek bölgesinde satirlar isimli bir çift pointer tanımlanıyor. Tanımlanan bu çift pointer, heap bellek bölgesinden dosyadaki satır sayısının iki katı kadar hücreden oluşan bir dizi alana alıyor. Alınan bu alanın her bir hücresi içerisinde pointer barındırıyor. Bu pointerlar, oluşturulacak olan üst ve alt listelerin, önce üst sonra alt olacak şekilde, ilk elemanlarını gösterecek şekilde ayarlanıyor. Bu işlemlerin ardından dosya kapanıyor ve kurucu fonksiyon tamamlanmış oluyor. Kurucu fonksiyon içerisinde gerçekleştirilecek işlemler için iki sınıf daha tanımladım. Bunlardan ilki, kullanıcı tanımlı bir tür olan Dugum sınıfı. Bu sınıfın bir tek görevi var: Alınan veriyi içeren bir bağlı liste düğümü oluşturmak ve bu düğümün sonraki düğümünün adresini barındıran işaretçiye NULL değerini atamak. Tanımladığım diğer bir sınıf BagliListe sınıfı. Tek yönlü bağlı liste oluşturan bu sınıfın tek görevi Dugum sınıfını kullanarak oluşturulan düğümlerin birleştirilmesiyle oluşan listeleri oluşturmak değil. Bu sınıf aynı zamanda DosyaOku sınıfında ortalama hesabı yapmak için kullandığımız fonksiyonun istenen düğümlerin elemanlarını matematiksel olarak toplayabilmesi için gereken verileri getiriyor. Yine DosyaOku sınıfında, ortalama hesabının yapılabilmesi için

gerekli olan, bir listenin maksimum eleman sayısının bulunmasına listeElemanSayisi fonksiyonuyla yardımcı oluyor. BagliListe sınıfının gerçekleştirmekle görevli olduğu son işlem, gerektiği zaman listelerin silinerek heap bellek bölgesinde kapladıkları alanların serbest bırakılması. Bu görevi gerçekleştiren fonksiyon ise sınıfın yıkıcı fonksiyonu. Kurduğum algoritma sayesinde yıkıcı fonksiyon tüm listeyi dolaşıyor. Bu dolaşma sırasında listenin ilk elemanından başlayarak son elemana kadar düğümleri tek tek silerek ilerliyor. BagliListe sınıfının (ve dolaylı olarak Dugum sınıfının) sıkça kullanıldığı DosyaOku sınıfında, kurucu ve yıkıcı fonksiyonlar haricinde, iki ana fonksiyon ve kod tekrarından kaçınmak amacıyla oluşturulmuş birkaç adet yardımcı fonksiyon bulunuyor. Ana fonksiyonlardan programın çalışması sırasında ilk kullanılan ustAltListeDegistir isimli konum değiştirme fonksiyonu. Bu fonksiyonun çalışabilmesi için kullanıcıdan iki adet konum bilgisi alınıyor. Alınan konum bilgilerinden ilki (Konum A) Şekil 1'de bulunan dizinin üst bağlı liste konumunu belirtirken ikincisi (Konum B) dizinin alt bağlı liste konumunu belirtiyor. Uygun olmayan konum bilgileri girildiğinde kullanıcıya uyarı veriliyor ve program sonlanıyor.



Şekil 2. Bağlı Liste Konum Değiştirme

Örneğin kullanıcı tarafından Konum A için 0 ve Konum B için 0 değerleri alınırsa Şekil 2'de gösterildiği gibi dizideki pointerların, gösterdikleri listeler değişmeden, dizi içerisindeki konumları değişiyor. Tabii konum değiştirme işleminin arka planında alınan konum değerleri direkt kullanılmıyorlar. Oluşturduğum tasarım neticesinde ustAltListeDegistir fonksiyonuna gönderilen konum bilgilerinden Konum A'nın iki katı ve Konum B'nin iki katının bir fazlası satirlar dizisinin indisi olarak kullanılıyor. Ardından bu indisler kullanılarak dizideki seçilen pointerların yerlerinin değişmesi sağlanıyor. Konum değiştirme işlemi tamamlandıktan sonra sıra üst ve alt listelerin ortalamalarının hesaplanmasına geliyor. Bu hesaplama işlemini gerçekleştirmek için kullanılan fonksiyon, hangi tür listenin ortalamasını alması gerektiğine dair bir parametre alıyor. Satirlar dizisini iki, listeleri de birer kere dolanan program ilk döngüde kaç tur atacağını belirleyebilmek için maxElemanSayisi isimli yardımcı fonksiyonu kullanıyor. Bu fonksiyon, yaptığı hesaplamalar sonucunda üst ve alt liste çiftlerindeki maksimum eleman sayısını bulup bulduğu değeri ana fonksiyona döndürüyor. Bir içerdeki döngü ise ortalamasını hesapladığı liste türüne göre ya dizi sonundan bir önceki indise kadar ya da dizi sonuna kadar ilerliyor. En içteki döngü ise listedeki gerekli veriyi alıyor. Eğer listeden bir veri geldiyse bu veriyi toplama ekleyip toplamın bölüneceği bölen değerini bir arttırıyor. Program gereken ortalamaları tek tek alıp ortadaki döngüden çıktığında, elde ettiği ortalamaları birbirleriyle topluyor ve aynı türdeki listelerin tamamının ortalaması hesaplanmış oluyor. Elde edilen bu üst ve alt ortalama değerlerinin, yuvarlanmadan, virgülden sonra sadece bir basamağının gösterilmesi için virguldenSonraBirBasamak isimli yardımcı fonksiyon kullanılıyor. En sonunda elde edilen değerler konsol ekranında kullanıcıya gösteriliyor. DosyaOku sınıfı da BagliListe sınıfında olduğu gibi bir adet yıkıcı fonksiyon içeriyor. Bu yıkıcı fonksiyon bellekte çöp oluşmaması için öncelikle tek tek bağlı listelerin ilk elemanlarını gösteren pointerların gösterdiği listelerin kapladığı alanları serbest bırakıyor. Bu aşamada eş zamanlı olarak BagliListe sınıfının yıkıcı fonksiyonu da, sahip olunan liste adedince, çalışıyor. Tüm listelerin yok edilmesinin ardından satirlar çift pointerının gösterdiği, heap bellek bölgesinde oluşturulmuş olan pointer dizisinin kapladığı alan da serbest bırakılıyor. Bellekten alınan alanların iade edilmesi işleminin tetiklenmesi de dahil olmak üzere listelerin konum değiştirmesi ve üst/alt listelerin ortalamalarının alınması işlemlerinin tetiklenmesi, Test sınıfı içerisindeki main fonksiyonunda oluşturulan DosyaOku sınıfının nesnesi kullanılarak gerçekleştiriliyor. Main fonksiyonunun sonunda heapte oluşturulan bu nesnenin sahip olduğu alan da serbest bırakılıyor ve programın çalışması tamamlanarak sonlanmış oluyor.

Ödev, program tasarlama ve programdan yapması beklenen işlemlere göre algoritma geliştirme kısmında pratik yapmamı sağladı.