T.C.
SAKARYA ÜNİVERSİTESİ

BİLGİSAYAR VE BİLİŞİM BİLİMLERİ FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

VERİ YAPILARI ÖDEV RAPORU

2.ÖDEV

Sinem Şaziye Karaca
G211210025

SAKARYA

Ağustos, 2023

Veri Yapıları Dersi

ÖDEV SÜRECİ

Veri Yapıları dersinin ikinci ödevinde içinde 0-256 arası sayıların bulunduğu metin dosyasını okuyan ve okuduğu her satırda elemanları birer yığına ekleyen program yazılması isteniyor. Fakat o an okunan satırdaki tüm elemanların aynı yığında olmaması, eğer eklenmesi beklenen yeni sayı çiftse ve yığının en üstünde bulunan sayıdan büyükse yeni bir yığın oluşturup eklenecek sayının oluşturulan yeni yığına eklenmesi isteniyor. Ardından bir satırın tüm yığınları oluşturulduktan sonra bu yığınların tek tek birer ikili arama ağacına eklenmesi isteniyor. Ağaçlara ekleme işleminin tamamlanmasının ardından elde edilen ağaçlar arasından yüksekliği en çok olanın seçilmesi gerekiyor. Eğer en yüksek değere sahip iki ağaç varsa önce bu ağaçların tüm değerlerinin toplamına bakılacak. Tüm düğümlerindeki verilerin toplamları da eşit olan ağaçlardan önce oluşturulan ağaç seçilecek. Şartlara uyan ağaç seçildikten sonra bu ağacın verilerinin ASCII karakter karşılıkları postorder formda olacak şekilde ekrana yazdırılacak.

Programa öncelikle Dosya Okuyan sınıfı oluşturarak başladım. Bu sınıfı oluşturduktan sonra yığın ve ağaç sınıflarını oluşturdum. Öncelikle Dosya Oku sınıfının içinde sıkça kullanacağım yığın ve ağaç veri yapılarının içeriklerini oluşturdum. Bu veri yapıları tek yönlü bağlı liste veri yapısı kullanılarak gerçeklendi ve elemanları kendi düğüm sınıflarında oluşturuldu. Yığın sınıfından bir nesne oluşturulduğu an çalışan kurucu fonksiyon yığının tepe değerinin adresini tutan pointera NULL değerini atıyor. Yığına ekleme işlemi yapılacağı zaman ekle fonksiyonu çalışıyor ve ekleme işlemi yığın listesinin ilk elemanın yerine geçecek şekilde yapılıyor. Getir fonksiyonu yığının tepesindeki elemanı getiriyor. Yığın boşsa -1 değerini döndürüyor çünkü dosyadaki elemanlarla kıyaslama yapılacak. Bir satır çift sayı ile başlıyorsa herhangi bir karşılaştırma sorunu yaşanmaması için boş yığında -1 değeri döndürülüyor. Çıkar fonksiyonu yığının tepesindeki verileri, yığında eleman kalmayana kadar, yığından çıkarıyor. Kodun tutarlı bir şekilde çalışıp çalışmadığını programı oluşturma esnasında kontrol etmek için bir adet yazdır fonksiyonu bulunmakta. Ayrıca bu sınıfta ağaca eleman eklerken döngünün kaç tur döneceğini belirlememize yardımcı olan eleman sayısı fonksiyonu bulunmakta. Bu fonksiyon bir yığında kaç eleman olduğunun bilgisini veriyor. Son olarak yıkıcı fonksiyon, yığında kalan elemanları siliyor.

Ağaç sınıfının, bu veri yapısının nesnesi oluşturulduğunda yapıcı fonksiyonu çalışıyor ve kök düğümünü tutan pointera NULL değerini atıyor. Sınıfın üye fonksiyonlarından Ekle fonksiyonu eklenecek verinin ağaçta var olup olmadığını kontrol etmekle görevine başlıyor. Bu kontrolü Var Mı isimli fonksiyon yardımıyla yapıyor. Var mı fonksiyonu kökten başlayarak bütün ağacı dolaşıyor. Bu dolaşım esnasında üzerinde bulunduğu düğümün verisiyle eklemek istediğimiz veriyi karşılaştırıyor. Veriler arasında bir eşleşme yakalarsa false değeri geri döndürüyor, aksi halde true değeri döndürerek kontrolü yapılan verinin ağaca eklenebileceğini bildiriyor. Eklenecek veri ağaca daha önceden eklenmişse yeni veri ağaca eklenmiyor. Eğer ağaçta yoksa ikili arama ağacının kurallarına uygun olarak ağacın ilgili bölgesine ekleniyor. Herhangi bir ağacın verilerinin ASCII karakter karşılıklarını postorder bir şekilde ekrana yazdıran Postorder, ağaçların yüksekliklerini hesaplayan Yükseklik ve bir ağacın bütün düğümlerinin verilerinin toplamlarını hesaplayan Düğümler Toplamı fonksiyonları bulunmakta. Yıkıcı fonksiyonun çağırmasıyla ağacın heapte kapladığı alanı iade etme işlemini gerçekleştiren Düğüm Sil fonksiyonu, kök düğümden başlayarak bütün düğümleri siliyor.

Dosya Oku sınıfı, sayılar isimli metin dosyasının içinde bulunan verileri okuyor ve yaptığı işlemleri her satırda baştan başlamak koşuluyla tekrarlıyor. Bu sınıftan nesne oluşturulduğu gibi çalışan kurucu fonksiyon öncelikle tüm dosyadaki en çok elemanı bulunan satırın eleman sayısı kadar heap bellek bölgesinden alan alıyor. Aldığı bu alan, yığınların tepe verilerinin adreslerini tutan pointer dizisini oluşturuyor. Bu işlemi gerçekleştirebilmek için Satır Max

Eleman isimli yardımcı fonksiyonu kullanıyor. Ardından dosyadan okuduğu verileri, istenen biçimde, yığınlara ekleyen Oku ve Ekle fonksiyonu çağrılıyor. Bu fonksiyon yapacağı tüm işlemleri satır satır yapıyor. Bir satırı okuduğunda öncelikle yeni bir yığın oluşturuyor. Ardından o satırın verilerini sırayla oluşturduğu yığına eklemeye başlıyor. Eğer yığına eklenecek bir sonraki sayı çift ve yığına en son eklenen sayıdan büyükse yeni bir yığın oluşturuluyor. Bu işlem satır sonuna kadar devam ediyor. Yığınlar oluşturulduktan sonra sıra bu yığınlardaki verilerle ağaç yapılarını oluşturmaya geliyor. Ağaca Ekle fonksiyonunun yığınlar çift pointerı ve ilgili satırda oluşturulan yığın sayısının parametre olarak gönderilerek çağrılmasıyla yığın sayısı kadar dizi heap bellek bölgesinden alınıyor. Alınan bu bölge her bir ağacın kök düğümünün adresini barındırıyor. Ardından senaryoya uygun şekilde ağaçlar oluşturuluyor. Oluşturulan her ağaçta verilerin alındığı yığının bellekte kapladığı alan serbest bırakılıyor. Ağaçlar oluşturulup yığın serbest bırakıldıktan sonra sıra uygun ağacı seçmeye geliyor. Ağaç Seç fonksiyonu ağaçlar çift pointerını ve ağaç sayısını parametre olarak alıyor. Çift pointer kullanımıyla bir satırın tüm ağaçlarına erişilebiliyor ve bu ağaçlar arasında kıyaslama yapılabiliyor. Öncelikle ağaçların yükseklikleri kıyaslanıyor, yüksekliği en fazla olan ağaç seçiliyor. Yükseklikleri aynı olan ağaçların tüm düğümlerindeki verilerin toplamı kıyaslanıyor, toplamı büyük olan ağaç seçiliyor. Eğer tüm düğümlerin verilerinin toplanması sonucunda sonuçlar aynı çıkarsa önce oluşturulan ağaç seçiliyor. Seçilen ağacın verilerinin ASCII karakter karşılıkları postorder formda ekrana yazdırılıyor. Bu işlemin ardından ağaçların da görevi tamamlanmış oluyor. İlk olarak ağaçlar pointer dizisinin her bir elemanının barındırdığı ağaç yapıları, ardından dizinin kendisi belleğe iade ediliyor. Sleep fonksiyonu sayesinde her satırın ekrana yazılışının ardından 10 milisaniye bekleniyor. Tüm bu işlemler tamamlandıktan sonra aynı işlemler bir sonraki satır için tekrarlanıyor. Tüm bu süre boyunca her satırda oluşturulan ağaçlar ve bu ağaçların tutulduğu dizi belleğe iade edilirken yığınların sadece kendileri iade ediliyor. Yığınların tutulduğu dizi ise programın sonlanma aşamasında çağrılan Dosya Oku sınıfının yıkıcı fonksiyonu içerisinde belleğe iade ediliyor.

Test sınıfı içerisindeki main fonksiyonunda Dosya Oku sınıfının heapte bir nesnesi oluşuyor. Oluşturulan bu nesne sayesinde veriler istenen şekilde ekrana yazdırılıyor. Program tamamlanmadan hemen önce heapte oluşturulan Dosya Oku nesnesi belleğe iade ediliyor ve program sonlanıyor.

Bu ödev yığın ve ağaç veri yapılarının nasıl oluşturulduğunu, farklı sınıflar içerisinde nasıl kullanıldığını ve pointer kullanımını daha iyi kavramamı sağladı. Algoritma oluşturma konusunda da pratik yapmamı sağladı.