T.C.
SAKARYA ÜNİVERSİTESİ
BİLGİSAYAR VE BİLİŞİM BİLİMLERİ FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

Ders : Veri Yapıları
Dönem : 2020-2021 GÜZ Dönemi

Adı Soyadı : Muhammet Kemal Güvenç

Okul No : B181210076

Konu : 1.Ödev

Şube : 1.Öğretim – C Grubu

Yazılan kodda değişkenler, metot adları ve sınıf adları kodun evrensel olması için İngilizce olarak yazılmıştır. Fakat kodları size en iyi şekilde anlatabilmek için yorum satırları Türkçe olarak yazılmıştır. Değişkenler küçük harflerle yazılmış eğer birden fazla kelimeden oluşuyorsa araya _ işareti konularak adlandırılmıştır. Sınıf adları ise bu kurala ek olarak baş harfleri büyük olarak adlandırılmıştır.
Projede Node, Iterator ve List olmak üzere toplam 3 sınıf bulunmaktadır. Bu sınıflardan Node ve Iterator kullanıcının erişimine kapatılmış sadece List sınıfı yine belirli kısıtlar olmak üzere kullanıcıya açılmıştır. Bütün sınıflarda veriler, kodun güvenliği için get ve set metotları üzerinden alınıp, verilmiştir. Ayrıca veriler üzerinde değişikliğe gitmeyen fonksiyonlar const olarak belirtilmiştir. Sınıflar, başlık dosyası ve kaynak dosyası olmak üzere 2 dosyaya yazılmıştır. Başlık dosyalarında sınıfların üye değişkenleri ve metot bildirimleri, kaynak dosyalarında ise metotların gövdeleri bulunmaktadır. Kod yazılırken elden geldiğince gereksiz bellek tüketiminden kaçınılmıştır.
Düğüm sınıfı, kendi içinde tuttuğu eleman, ilerisinde ve gerisinde olan düğümlerin adresleri olmak üzere 3 üye değişkene sahiptir. Bir de her bir elemana ait get ve set metotları olmak üzere 6 tane metot bulundurmaktadır. Tabii, bunlara ek olarak bir de yapıcı metodu vardır. Düğüm sınıfından bir nesne oluşturulurken düğümün içine konacak eleman, ilerisinde ve gerisinde olan düğümlerin adreslerinin verilmesi şart koşulmuştur. Bu sınıfı Iterator ve List sınıfında kullanabilmek için bu sınıflar arkadaş sınıf olarak eklenmiştir.
Iterator sınıfı gösterdiği düğümün adresini tutacak bir gösterici dışında başka bir üye değişkene sahip değildir. Bir iterator oluşturulurken bağlanacağı bir düğüm adresi verilmesi şart koşulmuştur. Buna karşılık yapıcı metodu dahil olmak üzere 5 adet metodu bulunmaktadır. Bunlardan 2'si değişkenin get ve set metotları diğer 2'si ise Iterator'ün bağlı olduğu düğümden ileri ve geri gitmesini sağlamaktadır. Bu sınıfı List sınıfında kullanabilmek için List sınıfı arkadaş sınıf olarak eklenmiştir.
List sınıfında 2 tane üye değişkeni bulunmaktadır. Bunlar listenin ortasını işaret eden gösterici ile listenin boyutunu tutan değişkendir. Bu sınıftan yeni nesne oluşturulduğunda göstericiye NULL, boyuta 0 atanır. Yapıcı fonksiyon dışında 11 adet fonksiyon bulunmaktadır. Bunların 4'ü üye değişkenlerin get ve set metotlarıdır. Diğerleri de liste üzerinde işlem yapmak ya da işlem yapılırken yardımcı olan metotlardır. Bunlara şöyle bir göz gezdirirsek:

increase_size: Bu metot listenin boyutunu bir arttırır.
middle_item: Orta düğümdeki düğümün değerini döndürür.
show: Bu metot listenin ekrana çıkartılmasını sağlar. Mekanizması ise şu şekildedir: İlk önce listenin orta düğümü gösteren bir iterator oluşturulur. Sonra bu iterator listenin başına götürülür. Daha sonra ise bulunduğu düğümün değerini gösterecek şekilde tek tek listenin diğer ucuna kadar ilerler.
add_array: Bir dizi ve bu dizinin boyutunu parametre olarak alan sonra bu dizideki değerlerden oluşan bir dairesel çift bağımlı liste oluşturur. Önce dizinin ilk elemanını içeren bir düğüm oluşturulur. Bu düğümün adresi listenin orta düğümünü gösteren göstericiye atanır. Sonra bu orta düğümü gösteren bir iterator oluşturulur. Bu iterator aracılığıyla dizinin geri kalan elemanlarının yarısı, sol tarafta düğümler oluşturulurken buralara konur. Daha sonra iterator tekrar orta düğüme getirilir ve sol tarafa yapıldığı gibi bu seferde sağ tarafa dizi elemanlarının yarısı düğüm oluşturulurken buralara konur. En sonda ise bir dairesel çift bağımlı düğüm oluşturulmuş olur.

is_big: Parametre olarak aldığı liste ile metodu çağıran nesneyi ortalarındaki düğümün değerinin büyüklüğü ile kıyaslar. Eğer metodu çağıran nesne büyük veya diğeriyle eşitse true, küçükse false döndürür.
invert: Çaprazlama işleminde kullanılan bir metottur. Listenin elemanlarını ters çevirir. Örnek vermek gerekirse şu şekildeki bir listeyi: 1 2 3 4 5 şöyle yapar: 5 4 3 2 1 Bunu ise şu şekilde yapar: 2 adet orta düğümü gösteren iterator oluşturulur. Bu iteratorlerden biri sola biri sağa olmak üzere listenin uçlarına doğru ilerler. İlerlerken de değerleri karşılıklı olarak değiştirirler.
cross_lists: Bu metot, aldığı liste ile onu çağıran nesneyi çaprazlar. Bu metodun temelinde şu vardır: İki liste çaprazlanınca oluşan yeni listeler ile eski listeler arasında bir ilişki bulunmaktadır. Diyelim ki A ve B adında 2 listemiz var ve bu A ve B listelerinin çaprazlanması sonucu A' ve B' listeleri oluştu. A' listesi B listesinin elemanlarının ters çevrilmiş ama ortasında A listesinin orta elemanı bulunacak şekilde olacaktır. Ben de bu ilişkiyi kullanarak bu metodu oluşturdum. Çalışması yukarıdaki mantık üzere ilk önce iki liste invert metoduyla ters çevrilir ve bu listelerin orta elemanları karşılıklı olarak değiştirilir. Böyle yapınca A listesi B', B listesi A' olur. Bunu düzeltmek için listelerin orta düğümlerini gösteren gösterici ile boyutları karşılıklı olarak değiştirilir. Böylece A listesi A', B listesi B' olur.

Ana programın çalışması ise şu şekildedir: İlk önce Sayilar.txt dosyasını açmaya çalışır. Eğer açamaz ise "Sayilar.txt dosyasi acilamadigi icin program kapatiliyor..." uyarısı verir ve kapanır. Dosya sıkıntısız bir şekilde açıldıktan sonra Sayilar.txt dosyasından kaç tane satır olduğu bilgisini okur ve satır miktarı kadarlık bir liste dizisi oluşturulur. Bu yapılırken bir anlamda kontrol de yapılır. Şöyle ki dosya okunmadan önce satır sayısı -1 olarak ayarlanır. Eğer dosya boş ise herhangi bir şey okuyamayacağı için bu değer değişmeyecektir. Program da bu değeri kontrol ederken hâlâ bu değerin korunduğunu fark edince "Dosya bos oldugu icin program kapatiliyor..." uyarısı verip, kendini kapatacaktır. Eğer her şey düzgün giderse program tekrar dosyayı okumaya devam eder. Satırların başından satırda kaç tane eleman olduğu bilgisini okur ve eleman miktarı kadarlık bir dizi oluşturulur. Daha sonra program, satır sonuna kadar, okuduğu elemanları bir diziye atarak, okumaya devam eder. Bu oluşan diziyi listelerden birine yerleştirir. Program bu mantıkla tüm dosyayı okur ve tüm listeleri doldurur. Bu arada Sayilar.txt dosyasında kaç adet satır olduğu ve satırda kaç adet sayı olduğu bilgisi girildiği için program gereksiz işlemlerden kurtulmuş ve daha hızlı işlem yapar hale getirilmiştir. Listeler oluşturulduktan sonra tek tek ekranda gösterilir. Sonra en büyük ve en küçük listeler, orta düğümler kıyaslanacak şekilde belirlenir ve onlar da ekranda gösterilir. Son olarak bu iki liste çaprazlanıp, ekrana orta düğümlerinin adresleri ile birlikte ekranda gösterilirler.

Not: Sayilar.txt'nin ilk satırında okunacak satır sayısı yazar ve her satırın başında, o satırda okunacak kaç eleman olduğu yazar.

Not: Dersin koordinatörü Dr.Öğr.Üyesi Muhammed Fatih Adak, Sayilar.txt'ye verilerin kaçar adet olduğunu yazmamın bir sıkıntı olmayacağını söyledi.