B191210088 - Furkan Cebar 1. Ödev Raporu

Benden "Sayilar.txt"'den tek sayı adedinde sayılar okuyup bunu Cift Yönlü Dairesel Bağıl Listeye önce orta eleman sonra yarısı sola yarısı sağa olacak şekilde atamam istenmiş ve ben de bunun için dosyadan satır satır string okuyup ardından stringstream ile okuduğum stringi integer değerlere böldüm 'integer sayısı + 1' elemanlı bir dizinin 1. elemanından son elemana kadar sayıları yerleştirdim 0. elemana da dizideki eleman sayısını koydum çünkü 'dynamic array' 'lerin eleman sayısı elde edilemiyor. Daha sonrasında benden okuduğum listeler arasında orta değeri en büyük ve en küçük olanı bulmam istendiği için ve bu elemanlara liste oluşturmadan (hafıza da gereksiz yer ayırmadan) erişebileceğim için liste oluşturmadan önce max ve min adında 2 liste oluşturup ilk 1. satırdaki diziyi bunlara atayıp sonrasında listenin orta elemanı ile dizilerin [1] indexli elemanlarını karşılaştırdım ve en büyük, en küçük orta değere sahip listelerim max ve min oldu. Daha sonrasında bunları çaprazlamam istendi. Bunun için geçici bir temp listesi oluşturup minde ki değerleri ters çevirip temp e yazdım ardından maxda ki değerleri min'e yazdım ve max listesini hafızadan silip temp ile değiştirdim. Ardından max ve min listelerini ekrana yazdırdım.

Daha kolay bir kullanım sunmak adına << ve >> operatörlerinin overloading'ini öğrenip bu operatörler aracılığıyla diziden listeye aktarım ve listenin ekrana basılmasını sağladım. ((örnek 'array >> *min;' ve 'std::cout << min'))

Ödevde herhangi bir zorlukla karşılaşmadım. Eğer talimatları yanlış algılamadıysam eksik bir yer bıraktığımı düşünmüyorum.

Listeyi oluştururken 5. hafta gösterdiğiniz listelerin çalışma mantığını anlatan sitede listelerin nasıl çalıştığını inceleyerek ve sistemde yüklü olan LinkedList dökümanının üzerinde oynamalar yaparak ilerledim. Bu sırada daha farklı ve kendime has nasıl yazabilirim diye düşündüm. Hata.hpp ve Hata.cpp'yi projemden kaldırdım çünkü tamamiyle birbirine bağlı olan bir listede herhangi bir hata durumu yaşanması söz konusu olmadığını düşündüm. (Örneğin -3 ile 3 arasında oluşturulmuş bir listede 4. eleman = -3. eleman olacaktır ve arama yaparken bu şekilde kullanılabilir.)

Oluşturduğum liste nasıl çalışıyor:

- Liste Node adı verilen parçalardan oluşuyor, bu parçalar integer tipindeki veriyi, sonraki ve önceki node'un pointerlarını tutuyor.
- Node oluşurken eğer sonraki ve önceki tanımlanmazsa Node kendisini pointliyor bu sayede eklenen veriyle birlikte yuvarlak bir veri havuzu oluşuyor ve eğer sonraki ve önceki tanımlanırsa bağlanmayı otomatik Node constructoru sağlıyor aynı şekilde aradan Node çıkarmak içinde sadece Destructor yeterli oluyor.
- Bütün ekleme ve silme işlemleri iteratör aracılığıyla sağlanıyor.
- Liste içinde arama işlemlerinde önce iteratörün şuanki konumundan mı yoksa orta düğümün konumundan mı daha hızlı erişilir şeklinde bir sorgu yapılıyor ve hangisi daha kolay olacaksa o şekilde arama gerçekleştiriliyor. ((örnek: iteratör konumu:5 ve gidilecek konum:8 ise 8-5 < 8 olduğu için iteratörün şuanki konumundan ilerleniyor ama iteratör konumu:9 ve gidilecek konum:2 ise 9-2 > 2 olduğu için iteratörün konumu sıfırlanıyor ve 0 dan 2 ye ilerliyor)) Bu sorgu işlemi listenin boyutu ve listenin orta düğümünü gerektirdiği için List classı üzerinde gerçekleştirilip ardından Iteratör üzerinde ileri ve geri gidiliyor.