Veri Yapıları

ÇİFT YÖNLÜ BAĞLI LİSTE İLE SAYISAL İŞLEMLER

1. GELİŞTİRİLEN YAZILIM

İlk olarak en zorlandığım kısım yer değiştirme oldu. Kağıda çizerek daha kolay kavradım. Özellikle ilk elemanların değişmesi kısmı sıkıntılı oldu çünkü bağlı listelerin head'lerinin de güncellenmesi gerekiyordu. Araya veya sona eklemede elemanların belli adımlarla prev ve next lerini değiştirmek yeterli oldu.

Kodun içeriğinden bahsedilecek olursa hpp dosyalarını şablon olarak kullanıldı ve asıl kod cpp dosyalarına yazıldı. Çift yönlü bağlı liste istendiği için adresleri tutmak amaçlı bir düğüm dosyası oluşturuldu. Bu dosyada önceki ve sonraki adresleri tutması amaçlı değişkenler eklendi. Çift yönlü bağlı liste de gerçekleşecek işlemler için ise 'DoubleLinkedList' dosyası oluşturuldu. Buraya 'Node' dosyası eklendi.

- **'DoubleLinkedList' dosyasında bulunan metotlar ve amaçları;**

Node* FindPrevByPosition(int) : Araya ekleme işlemlerinde kullanılabilmesi için öncekine yerleş metodu. Bir for döngüsüyle head den baslayarak itr->next ile liste dolaşılıyor parametre olarak gelen index'e gelene kadar for dönüyor ve böylece bir önceki düğüme yerleşmiş oluyoruz.

Node* FindByHead() : Bulunduğu düğümün ilk elemanını döndürüyor.

DoubleLinkedList() : Kurucu fonksiyon. Listenin boyutu 0 veriliyor ve liste başı NULL u gösteriyor.

bool isEmpty()const : Eğer boyut sıfırsa true döndürüyor.

int Count()const : Listenin boyutunu döndürüyor.

void add(const int) : Sona ekleme yapıyor. Insert metodu çağrılıyor ve parametre olarak listenin boyutu ve eleman veriliyor.

void insert(int,const int) : Parametre olarak index değeri ve eleman alınıyor. Eğer başa eklencekse yeni bir düğüm oluşturuluyor ve head artık burayı gösteriyor. Eğer araya veya sona eklenecekse bir önceki elemana

yerleşiyoruz. Yerleştiğimiz elemanın next i yeni oluşturulan düğümü gösteriyor. Yeni elemanımızın previ önceki elemanı next i şuanki bulunduğumuz konumu gösteriyor.

void removeAt(int) : Insert işlemine benzer şekilde silme yapılıyor ayrıca del pointerının tuttuğu düğüm iade ediliyor.

const int karsilastir(int)const : Bu metot for döngüsüyle listeyi dolaşarak bulunduğu index in datasını döndürüyor.

void inverse() : Listenin başından başlanıyor. Temp de bulunduğumuz düğümün next i tutuluyor. Bulundugumuz dugumun next i yerine previ yazılıyor. Bir dahaki while döngüsünde kullanmak için Prev değişkenine şuan ki iterasyon yazılıyor. Bulunduğumuz düğüm artık bir sonraki düğüm.

void guncelleme(Node*) : Item olarak gelen değer head e atanıyor yani head güncelleniyor. Yer değiştirme metodu için yapılmış bir methottur.

void yerDegistir(Node*, Node*, int) : Parametere olarak iki düğümün item ını alıyor.

- test.cpp de Sayı 1 in head'ini alıp bir değişkene atıyorum ve bu metotta parametre olarak head'in gösterdiği düğümü alıyorum.
- Eğer baştaki elemanlar değişecekse temp bulunduğumuz düğümün next ini tutuyor (ilk düğümü)
- İlk liste Sayı 1 ikinci liste Sayı 2 olsun, Sayi1'in next'i Sayı 2'nin next'ini gösteriyor
- Sayı 2'nin next'inin previ temp'in gösterdiği yeri Sayi1'in head'ini gösteriyor
- Sayı 2'nin next'i temp'in gösterdiği yeri gösteriyor.(Sayı 1'in head'inin next'i)
- Son olarak Sayı 1'in temp'in previ Sayı 2'yi gösteriyor.
- İlk elemanları değiştirmiş oluyoruz fakat Sayi1'in ilk elemanı Sayi 2 listesine gittiği halde head'i hala Sayi1'i gösteriyor yani sanki ilk düğümleri değil kalan diğer tüm düğümleri değiştirmiş gibi gözüküyor. Bu durumu düzeltebilmek için Sayi1 ve Sayi2'nin head'ini güncellemek gerekiyor.
- O yüzden bu fonksiyonu Sayı 1 ile çağırdığım için head olarak Sayı 1 listesinin başı tutuluyor, parametre olarak Sayı 2'nin head'i de alındığı için ben bunu Sayı 1 in head'i de (head = item2;) bu işlemden sonra Sayı 2'nin head'ini gösteriyor.
- Geriye sadece Sayı 2'nin head'ini eski Sayı1'in head'ini göstermesi kaldı. Zaten test.cpp de Sayı1'in head'i değişkende saklanmıştı. Test.cpp de güncelleme fonksiyonu çağrılıyor ve head güncellenmiş oluyor.
- Son elemanların değişmesi ise daha kolay iki elemanın da next i NULL gösteriyor elemanların previnin değişmesi ve elemanın önceki düğümünün nextini değişmesi yeterli.

friend ostream& operator<<(ostream&, DoubleLinkedList&) : Ekrana yazdırmak için kullanılan method.

void clear() : Listeyi temizleyen metod.

~DoubleLinkedList() : Yıkıcı fonksiyon. Çöpler temizleniyor.

- **'Sayi.cpp' dosyası ;**

  Test.cpp de kullanmak için gerekli bazı fonksiyonlar bu dosyaya yazıldı. Bu cpp dosyasında dosyadaki verileri stringe yazmak için bir fonksiyon ve dosyadaki sayıları '#' karakterine göre ayırıp vector dizisine atmak için bir fonksiyon yazıldı. Ayrıca dosyadan okurken 3 lü parçalara bölündüğünde ilk karakterin '0' gelmesi durumunda '0' yerine '1' yazılmasının kontrolü yapıldı.

- **'Test.cpp' dosyası ;**

  Bu dosyada oluşturduğum ArrayList dizisi substr ile üçlü parçalara bölündü ve sırayla listelere atandı. Sonra oluşan iki liste karşılaştırıldı ve çıkan sonuca göre gerekli fonksiyonlar çağrıldı.

- **'makefile' dosyası ;**

  İlk olarak DoubleLinkedList.cpp nin bir kütüphane dosyası oluşturuldu sonra bu kütüphane dosyası ve test.cpp derlenerek çalıştırılabilir bir exe oluşturuldu.