SAKARYA ÜNİVERSİTESİ
BİLGİSAYAR VE BİLİŞİM FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ
VERİ YAPILARI DERSİ ÖDEV 2

Ödevde kullandığım sınıf hiyerarşisi Dugum>BagilListe>Sayi>İslem şeklinde kurulmuştur. Bu sırada Gezici sınıfı BagilListe içerisinde yaptığım bazı işlemler için bana yardımcı olmaktadır.

Dugum.cpp

Dugum sınıfı içerisindeki tüm değişkenler public olarak tanımlanmış olup yapacağım projenin temelini bu sınıf oluşturmaktadır. BagilListe ler düğümler halinde gerçeklenip yorumlanır.

Gezici.cpp

Bu sınıf bagilliste içerisinde gerçekleştireceğimiz eylemler sırasında içerisinde bulundurduğu şimdiki dugumu yardımıyla listenin sonuna geldikmi? Listenin neresindeyiz? Aradığımız dugumu bulduysan geriye döndür gibi işlemler için kullanılmıştır. Bu kullanımlar sırasında kendisinden new anahtar kelimesiyle yeni bir nesne oluşturulmamıştır. Bir çeşit destek elemanı olarak kullanılmıştır.

BagilLliste.cpp

Projemizin ana konusu bagillisteler olup dökümanda benden istenilenleri gerçekleyebildiğimi söyleyebilirim. Bir çok standart liste fonksiyonunu barındıran bu liste bir tek yönlü bağlı liste olduğu için projeyi gerçekler bazı tasarımsal sıkıntılar yaşadığım oldu. Burada değinmek istediğim bir kaç önemli fonksiyon var. Bunlardan ilki Tersle fonksiyonu. Dediğim gibi listeyi tek yönlü tasarladığım için ve aritmetik işlemleri yapmaya en sağdan başladığımız için eşit girilmeyen liste durumlarında toplama işlemi yapmak probleme dönüşüyordu. Ben de buna çözüm olması için bir Tersle fonksiyonu yazdım. Bu fonksiyon listenin son elemanını ilk elemanına sondan bir önceki elemanını ikinci elemanına atayarak listedeki tüm dugum verilerinin yerlerini değiştiriyor böylece toplama işlemini yapabilmemiz için bize ortam hazırlıyor. Ardından + operatörünün aşırı yüklenmesi geliyor. Bunu yaparken gerçekten çok uğraşmam gerekti. Çünkü basamak sayısının eşit olduğu ya da fazladan bir basamak oluşacağı durumlarda yeni bir düğüm ekleyerek oluşan eldeyi oraya eklemem gerekiyordu.

Örneğin liste1+liste2 durumunda liste2 yi liste1 e ekleyip çıkan sonucu tekrar liste1 e eşitleyerek işe başladım yani pivot olarak seçtiğim liste soldaki listeydi her zaman için. Bu durumda girilen listelerin basamak sayıları önemliydi mesela ilk liste tek basamaklı ikinci liste ilk listenin basamak sayısından büyük girilirse bunları yer değiştirmem gerekiyordu. Tüm bunları yaptırmak epey zamanıma yol açtı fakat yaptığım denemeler sonucunda herhangi bir hata göremediğimi söylemeliyim. Olası tüm toplama durumlarını denediğimi düşünüyorum.

Bunun dışında yazdığım diğer fonksiyonlar standart olup herhangi bir tek yönlü bağlıliste gerçeklemesinde tekrar kullanılabilecek fonksiyonlardır.

Sayi.cpp

Sayi sınıfının görevi ödevi yollamam yakın bir dönemde değişiklik gösterdi. Buna göre +operatörünün aşırı yüklenmesi ve ekrana yazdırma operatörünün << aşırı yüklenmesi işlemleri artık sayi sınıfı için yapılacaktı. Hocamızın da dediği gibi zaten ödevi bitirmiş olanlar için bu küçük değişiklikleri yapmak pekte problem olmadı.

```
Sayi& Sayi::operator +(  Sayi& bg2 )
{

    *this->sayiListesi=*(this->sayiListesi)+*(bg2.sayiListesi);
    return *this;

}
```

Şekilde görüldüğü gibi gerçekleştiğim aşırı yükleme işlemi Sayi sınıfının içerisinde private olarak tanımlanmış bagilliste sınıfından değişkenleri birbirleriyle toplayarak yine yukarıda bahsettiğim gibi

tekrar this oparatörüne yani soldaki sayi nesnesinin içerisinde tanımlı olan bagillisteye atama yapıyor. Burada +operatörünü hem sayi hem de bagilliste sınıfı için aşırı yüklemiş oldum. Sayi sınıfı içerisindeki toplama işlemi gerçekleştirilmek istendiğinde sayi nesnelerinin içerdikleri bagillisteler toplanarak sonuç döndürülüyor. Böylece gerekli toplama işlemi gerçeklenmiş olmaktadır.

Yine ekrana yazdırmak için de burada << operatörünü aşırı yüklememiz gerekti. Fakat son adımda x+y işlemi için aralarda çizgi olamdan yazdırmamız gerektiğinden bir de operatörü aşırı yüklemeden void cinsinden ToplamiYaz() adlı bir fonksiyon tanımladım. Bu sınıf içinde benden istenilenleri gerçekleyebildiğimi düşünüyorum.

Islem.cpp

Gerçeklemesi en kolay ve zevkli sınıf bu sınıf oldu. Artık diğer sınıfların çalıştığından emin olduktan sonra bu sınıfın yazılması kalmıştı. İçinde private cinsinden 3 tane sayi nesnesi barındıran bu sınıf ilkListeYarat ve ikinciListeYarat fonksiyonları ile dışarıdan string cinsinde aldığı rakamları tek tek parse ederek Sayi nesnelerine oradan da dolaylı olarak bagillistelere ve düğümlere atamaktadır. Değerleri dışarıdan string olarak aldırmak istenilen büyüklükte sayıların toplanabilmesini sağlamaktadır. Ardından Topla() fonksiyonu yardımıyla sayılar aşırı yüklenmiş olan + operatörlerine bırakılmaktadır.

Sonuç

Ödevdeki kilit konular benim için dışarıdan alınacak değerlerin hangi cinste alınması gerektiği(string), +operatörünün nasıl düğümlerdeki her sayıyı tek tek toplayıp bunlardan oluşabilecek eldeleri bir sonraki toplama yansıtabileceklerini ve gerekli durumda nasıl ekstra bir düğüm eklenmesi gerektiğini tespitiydi. Ödevde benden istenilenleri yerine getirebildiğimi düşünüyorum.