Veri Yapıları Ödev 2 Raporu

Yazarlar: İsmet GÜZELGÜN(b121210025@sakarya.edu.tr)
İbrahim AKDAĞ(b121210027@sakarya.edu.tr)
İlhan KAYA(b121210005@sakarya.edu.tr)

Özet

Ödevde C++ programlama diliyle, bağlı listeler yardımıyla sınırsız sayı aldırmamız ve 4 adet sınıf oluşturmamız isteniliyor. Gerekli kısıtlamalar ise listeleri yazdırırken ve toplatırken aşırı yüklememiz gereken bir takım operatörlerden ibaret. Ödevde istenilenleri kısmen yerine getirdiğimizi düşünüyoruz. Açıklamalar aşağıdadır.

Dugum.cpp

Dugum oluşturulması için gereken elemanları içeren sınıf olmakla beraber sonradan "<<" operatörünün aşırı yüklenmesi sırasında kullanılacak bir takım fonksiyonlara da ev sahipliği yapmaktadır. Yazdığımız dugum sınıfı ekstra bir şey içermemektedir.

Sayi.cpp

Yazdığımız sayı sınıfı hem bir liste iteratörü gibi davranıyor hem de string olarak alınan sayıların parçalanıp sayı nesneleri halinde tutulmasında rol oynuyor. İçerisinde bir takım kontrol amaçlı fonksiyon barındırarak diğer sınıfların gerçeklenmesinde kodun geri kalan kısmında epey yardımcı oldu.

BagilListe.cpp

BagilListe sınıfı düğümlerden nesne alarak oluşturulan sayı türünden dugum verilerine ev sahipliği yapıyor aynı zamanda aralarında bir ilişki kurarak ekleme, çıkarma sıralama Heap bellek bölgesinde veri tutma gibi işlemleri gerçekleştirebilmemize yardımcı oluyor. Burada +operatörünü ve <<operatörünü aşırı yüklemek durumunda kaldık.<<operatörü için işler fena gitmedi fakat + operatörünü yüklemek ve işler hale getirmek beraberinde bir çok tasarım problemine sebep oldu. Öncelikle ödev dosyasında verilen sayılardan elde oluşmuyor olması bizim için bir soru işaretiydi. Biz istedik ki eğer elde oluşursa bunu bir solundaki tarafa atabilsin. Bu arada tüm bunlara gelmeden önce asıl kararsızlık yaşadığımız bölüm +operatörünü bir friend fonksiyon olarak mı tanımlamalıydık yoksa bagilliste cinsinden tek bir parametre aldırmalı ve bunu aynı bahsi geçen ödev dökümanındaki metoda uygun hale getirerek + operatörünün sağındaki listeyi solundaki listenin elemanlarına ekleyerek mi götürmeliydik. Herkesin konuya dair kendi fikri vardı ancak en son bahsi geçen tasarım şeklinde karar kıldık. İlk yazdığımızda yine bir çok tasarım sıkıntısı yaşadık hatta +operatörünü aşırı yüklemeden ödevi yollamayı düşündük. Sayıları string olarak aldırmayı akıl edip basit bir switch döngüsüyle tüm o to_string gibi hazır fonksiyonların mingw kütüphanesiyle olan uyuşmazlığından sıyrılmayı başarabilmiş belki de zor olan aşamayı ödev dökümanını okur okumaz aşmıştık fakat +operatörün epey başımızı ağrıtıyordu. İlk birkaç hali sadece toplama işlemini yapabiliyordu ve işin kötüsü ancak soldan sağa yapabiliyordu toplamayı ve eğer ilk girilen sayı ikinci girilen sayıdan daha küçük olursa program çalışmayı durduruyordu. İşin içinden çıkamayınca += operatörünü de aşırı yükleyerek iş yükünü azaltmaya çalıştık fakat bunda da başarılı olamadık. Kullanmaya karar verdiğimiz format liste=&(*listex+*listey) formatını destekler hale geldiğinde birkaç if döngüsü ile ilk girilen sayı büyük olsa da küçük olsa da işlem yaptırabilir hale getirmeyi başardık. Ama bu kez önümüzde elde problemi vardı. Elde işini halletmemiz epey zamanımızı aldı. Program şu anki haliyle bu işi başarabilmekte sayıları yine yazmak mecburiyetinde kaldığımız Tersle fonksiyonu yardımıyla önce tersten toplatıp sonra toplamın atanacağı listeyi de tersleyerek düzgün bir şekilde elde oluşursa

oluşan eldeyi bir sağa kaydırarak toplama işlemini gerçekleyebilmektedir. Aşamadığımız tek sorunsa eğer toplamak için gönderilen 2 listeden uzunluğu fazla olandan daha çok basamaklı bir sonucumuz olursa oluşan bu sonucun eldesini listenin en sağına ekleyememezden kaynaklanıyor. Bunun içinde çok uğraştık hatta gereken if koşulunu yorum satırı olarak bırakmış bulunuyoruz +operatörünü aşırı yüklediğimiz fonksiyonun içerisinde. Sorun olan kısmı eğer bu fonksiyonu çalıştırmak istersek tutarsızlık oluyor bazı durumlarda doğru cevabı vermesine rağmen bazen alakasız bir sayı atayabiliyor. Bu sorunu aşamadığımız için ödevi kısmen başarabildiğimizi söyleyebiliriz.

Islem.cpp

Bu sınıf en son test.cpp dosyası içerisinde tanımlanıp listelerin oluşturulması ve toplama işleminin gerçekleştirilip sonucun ekrana çıkartılması görevini üstleniyor. Nesneye dayalı programlama mantığına uygun olarak tasarlayamadığımızı düşündüğümüz bir fonksiyonlar dizisinden oluşuyor. Bunu düzeltecek zamanı bulmak isterdik zira liste oluşturmak için 2 farklı fonksiyon kurmak zorunda kaldık oysa isterdik ki tek bir listeoluştur fonksiyonu hem liste1 hem liste2 için çalışabilsin. Farklı uzunluklarda gelen string cinsinden sayıları tek bir fonksiyonla listelere ekleyemedik. Burada yazmış olduğumuz KarakterdenSayiyaDondur(char) fonksiyonu oldukça işimize yaradı doğrusu.Bu sayede string dizilerini parçalayıp integer türüne dönüştürmek oldukça kolay hale geldi. Aslında tasarımın bu aşamasında hiç integer türüne dönüştürmekle uğraşmayıp parçaladığımız string parçalarıyla yani char türünden verilerle devam edilebilirdi diye de düşünmedik değil. Ama integer dan devam etme kararı aldık.

Sonuç

Bu kez üç kişi yazmayı denediğimiz bu ödev yapısı gereği başta basit gözükse de gerek üç farklı kişinin kendi yazdığı kodların diğerlerinin yazdığı kodlarla uyuşmaması gerekse tasarıma dair farklı bakış açıları işleri ve tasarım sürecini bizim için hayli uzattı. Oluşabilecek fazladan eldeyi listenin başına ekleyebilmeyi ve ödevi size kusursuz haliyle gönderebilmeyi isterdik fakat verilen süre içerisinde bunu başaramadık. Bunun dışında sayıları kullanıcıdan string olarak aldırıp parçala fikri ödevde bizden istenilen sınırsız sayı girişini mümkün kıldı. Verilecek sonraki ödevler için özellikle tasarım kısmında daha çok kafa yormamız gerektiğini anladığımız bu ödev gerek neler yapabileceğimizi tartışırken gerekse araştırırken bize bağlı listeler ve nasıl çalıştıklarına dair oldukça fazla bilgi kattı diyebiliriz.