T.C.
SAKARYA ÜNİVERSİTESİ

BİLGİSAYAR VE BİLİŞİM BİLİMLERİ FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

PROGRAMLAMA DİLLERİNİN PRENSİPLERİ ÖDEV RAPORU

Gezegenler Arası Seyahat Simülasyonu

B231210081 - Umut Arda VURAL

SAKARYA

MAYİS, 2025

Programlama Dillerinin Prensipleri Dersi

Gezegenler Arası Seyahat Simülasyonu

Umut Arda Vural

[a] B231210081 1/A

Özet

Problemimiz, gezegenler arası seyahat etmesi planlanan uzay araçlarının belirlenen rotalar doğrultusunda güvenli şekilde varış noktalarına ulaşıp ulaşamadığının kontrol edilmesidir. Bu süreçte uzay araçlarının o anki durumu, konumu ve kalan seyahat süresi dikkate alınarak her bir gezegenin nüfus verilerinin doğru ve tutarlı biçimde güncellenmesi gerekmektedir. Uzay aracı imha olmuşsa ya da tüm yolcular hayatını kaybetmişse, bu durum kullanıcıya saat başı güncellenen kontrol ekranında anlık olarak gösterilmelidir. Her saat sonunda; araçların konumları, durumları, yaşam süresi olan yolcular ve gezegenler arasındaki mesafeler kontrol edilerek sistematik biçimde güncellenmelidir. Elde edilen tüm veriler kaydedilmeli ve sistemin son durumunu gösteren bir rapor oluşturulmalıdır.Bu ödevimizin , böyle bir simülasyon senaryosunu anlayıp modelleyebilmemiz için verilmiş olması muhtemeldir.İlk ödevin üstüne ikinci ödevde alt gezegen türleri de eklenip kodlar yeniden c de yazıldı .C de yazılırken ders de gösterildiği gibi soyut sınıf benzetimi yapıldı .Gezegen yapısı soyut sınıf olarak kullanıldı .

Anahtar Kelimeler: Tutarlılık , Güncellik , Doğruluk , Yönetilebilirlik

1.GELİŞTİRİLEN YAZILIM AŞAMALARI

1.1 Ödev nasıl gerçekleştirildi

Projede, Gezegen, Kişi ve Uzay Aracı yapıları için ayrı diziler tanımlanmış ve her veri türü bu dizilerde sıralı şekilde tutulmuştur.

Dosya okuma işlemlerinde önce gezegenler okunarak gezegen dizisine eklendi. Ardından, uzay araçları okunarak kalkış ve varış gezegenleri, gezegen dizisinden referans alınarak eşleştirildi ve uzay aracı dizisine eklendi.

Kişiler okunurken, bulundukları uzay aracına göre ilgili referanslar belirlendi ve kişi dizisine eklendi.

Her gezegen yapısında bir Zaman yapısı tanımlandı. Bu yapı simülasyonda saat ve tarih artırma işlemleri için kullanıldı. Ekran temizleme işlemleri system("cls") komutuyla yapıldı. Tüm işlemler main fonksiyonunda sıralı şekilde çalıştırıldı.

1.2 Hangi yöntem neden kullanıldı

1.2.1 Dizi kullanımı

Bu projede verileri sıralı ve düzenli bir şekilde tutmak için dizi yapısı kullanılmıştır. Gezegen, Uzay Aracı ve Kişi verileri için ayrı diziler tanımlanarak, dosyadan okunan veriler bellekte ardışık olarak saklanmıştır. Dizi kullanımı sayesinde elemanlara indeks üzerinden hızlı erişim sağlanmış, arama ve eşleştirme işlemleri kolaylaştırılmıştır. Ayrıca, veri sayısının önceden tahmin edilebilir olması durumunda dizi kullanımı bellek açısından da verimli olmuştur.

1.2.2 Nesne benzetimli programlama

C dili nesne tabanlı yapıyı doğrudan desteklemese de, projede struct tanımları ve bu yapılara ait işlevlerle (fonksiyonlarla) nesne benzetimli bir yaklaşım uygulanmıştır. Her veri türü için ayrı struct yapıları tanımlanmış, bu yapıların verilerine erişim

ve güncelleme işlemleri ilgili fonksiyonlar üzerinden gerçekleştirilmiştir. Bu sayede kapsülleme, modülerlik ve kontrollü erişim sağlanarak kodun okunabilirliği ve yönetilebilirliği artırılmıştır.

1.2.3 Zaman yapısı

Tarih ve saat işlemleri için kendi Zaman yapımı oluşturdum. Bu yapı içerisinde gün, ay, yıl ve saat alanları yer aldı. Zaman ilerledikçe, saat ve gün artışıyla birlikte ay ve yıl geçişleri kontrol edildi.

Ayrıca, gerçek takvim kurallarına uygun olarak ayların gün sayıları dikkate alındı ve artık yıl kontrolü yapılarak Şubat ayı doğru şekilde işlendi. Tüm bu işlemler, manuel olarak yazdığım kontrol yapılarıyla sağlandı ve zamanın doğru bir şekilde ilerlemesi garanti altına alındı.

1.2.4 Dosyadan okuma yapısı

Dosya okuma işlemlerinde verileri mantıksal bir sıraya göre işledim. Öncelikle gezegen verilerini okuyarak gezegen dizisine ekledim. Ardından, uzay aracı verilerini işledim ve her bir uzay aracının kalkış ve varış gezegenlerini, daha önce oluşturulan gezegen dizisindeki verilerle karşılaştırarak referans şeklinde bağladım. Bu sayede uzay araçlarının doğru gezegenlerle ilişkilendirilmesi sağlandı ve veri bütünlüğü korundu.

Benzer şekilde, kişileri işlerken, kişinin bağlı olduğu uzay aracı bilgisi yine daha önce oluşturulan uzay aracı dizisi üzerinden bulunarak kişi nesnesine referans olarak aktarıldı. Böylece her kişinin hangi uzay aracında bulunduğu açık ve doğru biçimde ilişkilendirildi.

1.2.5 Hata kontrolleri

Uygulamanın hatasız çalışmasını sağlamak için, özellikle referans atamaları sırasında NULL kontrolleri yaptım. Örneğin, bir uzay aracının kalkış veya varış gezegeninin doğru şekilde bulunup bulunmadığı, ya da bir kişinin bağlı olduğu uzay aracının listede mevcut olup olmadığı kontrol edildi.

Eğer ilgili veri bulunamazsa, bu durumu önceden tespit edip programın çökmesini engellemek adına gerekli önlemler alındı. C dilinde try-catch yapısı olmadığı için bu kontroller if koşulları ile manuel olarak gerçekleştirildi.

Bu yaklaşım sayesinde, eksik veya hatalı veri girişlerinden kaynaklanabilecek çökme ve tutarsızlıkların önüne geçilmiş oldu. Yapılan her kontrol, projenin daha kararlı, sürdürülebilir ve güvenli olmasına katkı sağladı.

1.2.6 Zaman yapısı

Her gezegen için bağımsız bir zaman takibi sağlamak amacıyla, struct Gezegen yapısının içinde bir adet struct Zaman alanı tanımladım. Bu sayede her gezegenin kendi zaman bilgisini (gün, ay, yıl, saat) ayrı olarak tutması mümkün oldu.

Gezegen oluşturulurken, bu Zaman yapısı başlangıç değerleriyle birlikte tanımlandı ve her gezegenin kendi zaman kontrolü bu yapı üzerinden yapıldı. Zamanla ilgili tüm artış ve geçiş kontrolleri (saat → gün, gün → ay, ay → yıl, artık yıl vb.) Zaman yapısına ait fonksiyonlar ile gerçekleştirildi.Bu yapı sayesinde gezegenlerin zaman bilgileri birbirinden bağımsız, tutarlı ve kontrollü bir şekilde yönetildi.

1.2.7 Gezegen yapısı

Gezegen yapısı içerisinde gezegene ait isim, nufus, zaman bilgisi gibi verileri tuttum. Uzay araçları ile gezegenler arasında bağlantı kurmak amacıyla, uzay aracının kalkış ve varış gezegenlerini, daha önce oluşturulan gezegen dizisi üzerinden karşılaştırma ile bulup referans olarak atadım.Bu yöntem sayesinde, her uzay aracı nesnesi ile ona ait kalkış ve varış gezegenleri arasında doğru ve tutarlı ilişkiler kurulmuş oldu. Böylece veri yapıları arasında anlamlı bağlantılar oluşturularak işlemlerin güvenli ve düzenli bir şekilde ilerlemesi sağlandı.Soyut sınıf benzetimi ile alt gezegenleri Gezegen yapısı ile ilişkilendirdim azaltılcakomur fonksiyonu soyut olarak çalışmakta . Aldığı alt gezegen yapısına göre fonksiyon override edildi.Böylece projemde tasarımsal olarak gerekli olan soyut sınıf tasarımı gerçekleştirdim.

1.2.8 Uzay araci yapısı

UzayAraci yapısında, uzay aracına ait gerekli değişkenlerin (isim, hız, kalkış tarihi, varış süresi vb.) tanımlamalarının yanında, kalkış ve varış gezegenleri için pointer (referans) alanları tanımladım. Bu sayede, her uzay aracının bağlı olduğu gezegen bilgilerine doğrudan erişim sağlandı.Ayrıca, yapılandırma (başlatma) aşamasında, ayrı bir fonksiyon ile varış tarihi hesaplandı ve hesaplanan değer, uzay aracının zaman yapısına atanarak kaydedildi. Bu sayede uzay aracının varış zamanı, hem sistemde hem de simülasyonlarda doğru ve anlamlı şekilde kullanılabildi.

1.2.9 Simülasyon ve main yapısı

Tüm işlemlerin gerçekleştiği yapı içinde, her iterasyonda sırasıyla gezegenlerin zaman bilgileri artırıldı, kişilerin yaşları azaltıldı, uzay araçlarının kalkış ve varış durumları kontrol edildi, kişilerin konum bilgileri güncellendi ve gezegenlerin nüfus bilgileri yeniden hesaplandı. Bu işlemlerden önce her döngüde konsol ekranı temizlenerek görsel düzen sağlandı. Ardından, simülasyonun sona erip ermediği kontrol edilerek gerekli durumlarda döngü sonlandırıldı. Tüm bu yapı, ana fonksiyon içinde (event simulation fonksiyonu) başlatılarak simülasyonun genel akışı yönetildi.

1.2.10 Varış tarihinin hesaplanması

Varış tarihini hesaplarken önce uzay aracı kalkış yapana kadar kalkış gezegeninde geçen gün farkını buldum. Bu gün farkını, kalkış gezegeninin bir gününün kaç saat olduğuyla çarptım ve kalkışa kadarki saat farkını elde ettim. Ardından, bu saate uzay aracının yol süresini ekleyerek toplam geçen saat miktarını hesapladım. Bu toplam saati de varış gezegeninin gün uzunluğuna göre bölerek kaç gün geçtiğini buldum. Son olarak da bu gün sayısını varış gezegeninin kendi tarihine ekledim. Bu işlemleri yaparken tarih hesaplamalarının daha sağlıklı ve hatasız olması için Julian takvimine göre dönüşüm yapan fonksiyonlar kullandım. Julian sistemi sayesinde artık yıl gibi durumları da düzgün şekilde kontrol edebildim. Ayrıca bu yöntemi tercih etme sebebim, klasik zaman kütüphanelerinde karşılaşılabilecek 2038 yılı problemi gibi sınırları aşmak ve sistemi daha güvenli hale getirmekti.

1.3 Ödevin geliştirme amacı

Ödevimin amacı, gelecekte gelişen teknolojilerle insanların yeni gezegenlere seyahat etmesi ya da kolonileşme (yerleşme amaçlı konaklama) süreçlerini yönetecek bir kontrol uygulaması geliştirmektir. Bu uygulama, uzay teknolojileri ve biyoteknoloji alanlarındaki yenilikleri göz önünde bulundurarak, insanların gezegenler arası yolculuklarını organize etmek, klanolişme süreçlerini izlemek ve her iki sürecin güvenli bir şekilde yönetilmesini sağlamak amacıyla kullanılabilir. Bu sayede, uzaya seyahat eden kişilerin yeni gezegenlerde yerleşmeye çalışan bireylerin süreçlerinin doğru bir şekilde kontrol edilmesi ve yönetilmesi sağlanarak, gelecekteki uzay kolonizasyonu için sağlam bir altyapı oluşturulmuş olur.

2.ÇIKTILAR

Büyük veri setleri ve küçük veri setlerinde gerekli kontroller yapılıp eldeki veriler ile tüm çıktılar aynı oldu .Kontroller yapılırken eksik ve hatalı hesaplamalar düzeltildi .

3.SONUÇ

Bu proje, veri ilişkilerinin tutarlı, tarih bilgilerinin güncel ve sistemin genel yapısının doğru şekilde kurulması sayesinde başarıyla tamamlanmıştır. Kullanılan nesne yönelimli yapı, projenin yönetilebilirliğini artırmış ve sürdürülebilir bir mimari sunmuştur. Gerçek hayatta bu yapıların lojistik, ulaşım ve uzay teknolojilerinde karşılığı bulunmakta olup; bu projeyle bu sistemleri küçük ölçekte simüle etme fırsatı buldum. Kişisel olarak da algoritmik düşünme, hata ayıklama ve sınıflar arası ilişki kurma konularında kendimi geliştirdim.

* Ödev Sorumlusu. Umut Arda Vural B231210081,

Mail Adresi: umut.vural@ogr.sakarya.edu.tr

4.KAYNAKÇA

- https://support.microsoft.com/en-us/office/insert-julian-dates-functions-0c7fa6aa-daff-402e-9990-93a5b76ba018
- https://orbital-mechanics.space/reference/julian-date.html örnek kodlar bu kaynakçaları raporumdada belirttim örnek kodları c ye uyarladım