# PROGRAMLAMA LABORATUVARI 1

## 3. PROJE

Yunus Emre GÜL
Bilgisayar Mühendisliği Bölümü
Kocaeli Üniversitesi
yunemregul@gmail.com

### Özet

Bu doküman Programlama Laboratuvarı 1 dersi 3. Projesi için çözümümü açıklamaya yönelik oluşturulmuştur. Dökümanda projenin tanımı, çözüme yönelik yapılan araştırmalar, kullanılan yöntemler, proje hazırlanırken kullanılan geliştirme ortamı ve kod bilgisi gibi programın oluşumunu açıklayan başlıklara yer verilmiştir. Doküman sonunda projemi hazırlarken kullandığım kaynaklar bulunmaktadır.

### 1. Proje Tanımı

Coğrafi Bilgi Sistemi (CBS), dünya üzerindeki karmaşık sosyal, ekonomik, çevresel vb. sorunların çözümüne yönelik mekâna/konuma dayalı karar verme süreçlerinde kullanıcılara yardımcı olmak üzere, büyük hacimli coğrafi verilerin; toplanması, depolanması, işlenmesi, yönetimi, mekânsal analizi, sorgulaması ve sunulması fonksiyonlarını yerine getiren donanım, yazılım, personel, coğrafi veri ve yöntem bütünüdür. Bu proje kapsamında, şehirlerin komşuluk ilişkilerini dikkate alan çok basit bir harita verisi üzerinde bağlı listelerle ilgili temel işlevleri yerine getiren standart C dilinde yazılacak bir uygulama yapılması beklenmektedir.

Bu modelde her bir şehrin dört tane özelliğe sahip olduğu bilinmektedir: şehrin ismi, plaka kodu, hangi bölgede olduğu ve bu şehirden direk ulaşılabilen (komşu) şehirler. Şehirler arasındaki komşuluk ilişkisi bir pointer (işaretçi) yardımıyla tutulabilir. Her bir şehir bir bağlı listenin düğümü şeklinde tutulmalıdır.

Şehirler "sehirler.txt" isimli bir dosya içinde tutulacak olup dosya içeriğinin formatı aşağıdaki gibi olacaktır:

*<plaka kodu><,><sehir ismi><,><bulunduğu bolge><,><komşu sehir ismi><,><komşu sehir ismi>...*

Şehrin bulunduğu bölge için aşağıdaki kısaltmalar kullanılacaktır:

Akdeniz (AK), Doğu Anadolu (DA), Ege (EG), Güneydoğu Anadolu (GA), İç Anadolu (İA), Marmara (MA) ve Karadeniz (KA).

Örnek bir veri aşağıdaki gibidir:

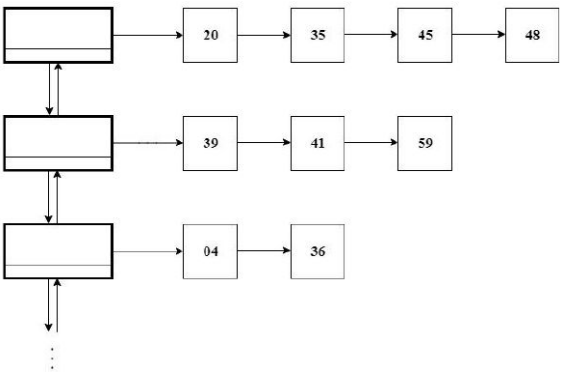
*30,Hakkari,DA,Van,Sirnak*

#### 1.1. Sistem Tasarımı

Tasarlanan sistem 2 çeşit düğüm (node) yapısından oluşturulmalıdır:

- 1. tip (şehir) düğüm yapısında her bir şehir için plaka kodu (int), şehir adı (string), ait olduğu coğrafi bölge (string) ve komşu sayısı (int) bilgileri tutulmalıdır.
- 2. tip (komşu) düğüm yapısında her şehir için komşu şehirlerin plaka kod (int) bilgilerini ardışık olarak tutulmalıdır.

Tasarlanan sistemde tüm düğümler plaka koduna göre ardışık sıra ile tutulmalıdır. Şekil 1'de örnek sistem modeli görülebilir.




**Şekil 1.** Örnek sistem modeli

### 1.2. İsterler

Tasarlanan sistem Şekil 1'i temsil edecek olup sınırsız sayıda düğüm ve komşuluk tutabilmelidir. Dinamik bir yapıda olmalıdır. Ayrıca aşağıdaki tüm isterler kullanıcıya bir menü aracılığı ile sunulmalıdır:

- Kullanıcı yeni bir şehir, yeni bir komşuluk ekleyebilmelidir. Aynı kayıtların eklenmesine sistem izin vermemelidir.
- Herhangi bir şehir, ismi veya plaka kodu ile aratıldığında şehir bilgileri (plaka no, şehir adı, bölgesi, komşu sayısı) ve komşu şehirlerinin bilgileri (plaka no, şehir adı ve bölgesi) gösterilmelidir. Listede olmayan bir şehir için arama yapıldığında "şehir listede yok, eklemek ister misiniz?" gibi bir seçenek sunulmalıdır.
- Kullanıcı bir şehri veya komşuluk kaydını silebilmelidir.
- Kullanıcı herhangi bir bölgede bulunan şehirlerin bilgilerini listeleyebilmelidir.
- Belli bir komşu sayısı kriterine (minimum-maksimum) uyan şehirler listelenebilmelidir.
- Modelin mevcut yapısı 'sehirler.txt' dosyasına yazdırılabilmelidir.

## 2. Araştırmalar ve Yöntem

Proje gerçekleştirilirken yapılması gereken ilk şey programın çalıştığı klasördeki 'sehirler.txt' dosyasının okunmasıydı. Dosyanın ismi ve yeri değişmeyeceğinden kolayca yaptım.

Şehirler dosyası okunduktan sonra içindeki bilgileri ayrıştırıp gerekli yerlerde kullanılması gerekiyordu. Her şehir bilgisi farklı satırda verildiği için bunu yapmanın en iyi yolunun dosyayı satır satır okumak olduğuna karar verdim.

Şehirler dosyasındaki her satıra ulaşabilince satırlardaki bilgileri işlemek kalıyordu. Her şehre ait bilgiler virgüllerle ayrıldığından satırları virgüllere göre bölerek parça bilgileri işledim. Örneğin satırın ilk parçasındaysak (0. indis) listeye eklenecek yeni şehrin plakasını belirle, ikinci parçasındaysak eklenecek şehrin ismini belirle gibi...

Şehirler dosyasındaki bilgileri işleyebiliyor olsamda karşılaştığım bir sorun komşuların eklenirken tüm şehirlerin eklenmiş olması gerektiğiydi. Şehirler dosyasında komşuların sadece ismi verildiğinden ve şehre eklerken plakasıyla ekleneceğinden tüm şehirlerin gezilip komşunun plakasının bulunması gerekiyordu. Bu sorunu çözmek için basit olarak önce tüm satırları okuyup şehirleri listeye ekledim, ardından tüm satırları tekrar okuyup şehirlere ait komşuları ekledim.

Şehirleri tutacak liste global bir değişken olarak tanımlanıp tüm program aynı liste üzerinde işlem yapabilirdi ancak bu yöntem yerine hem kendimi denemek hem de projeye ekstra bir zorluk katmak amacı ile şehir ekleyen, şehir silen fonksiyonlarımı birden çok şehir listesi üzerinde işlem yapabilir şekilde tanımladım. Böylece program sadece bir şehir listesine bağımlı ve kısıtlı olmamış oldu.

Projede şehirlerin ve komşuların sıralı olarak tutulması isteniyor. Her ikisi için de aynı algoritmayı kullanarak şehirleri ve komşuları sıralı biçimde ekledim. Sıralı bir listeden şehir silmek de sırasını bozmayacağından bu kısımda da sorun yaşamadım.

Program bazen, düzgün derlense bile, çalışırken 'Segmentation fault' gibi genelde bellek kaynaklı hatalar verebiliyordu. Bu hatalar çalışma zamanı hataları olduğundan ve bunların kaynağını derleyici derlerken gösteremediğinden program çalışırken takip etmek gerekiyordu. Bu takip işlemini de GDB (GNU Debugger) kullanarak yaptım. Böylece program çalışırken hata verdiği anda nerede olduğunu görüp ilgili çözümü uygulayabildim.

Projede modelin güncel yapısının 'cikti.txt' isimli dosyaya yazdırılması isteniyordu. Arayüze eklediğim 'Bilgi Listele' seçeneği ile hem terminale hem de 'cikti.txt' dosyasına aynı bilgileri yazdırdım. Aynı bilgileri hem dosyaya hem terminale iki kez yazdırmak için iki farklı fonksiyon çağırmak yerine önce *printf* kullanarak terminale ardından terminalin çıktısını bir dosyaya yönlendirip (*dup, dup2 fonksiyonları)* yine aynı *printf* bölümü ile dosyaya yazdırdım. Böylece aynı çıktı kodunu tekrar tekrar yazmaktan kurtuldum hem de 'cikti.txt' dosyası ile terminalde aynı çıktı uyumunu yakaladım.

## 3. Geliştirme Ortamı

Projemi Linux sistemde, Visual Studio Code OSS üzerinde geliştirdim. Projemi derlerken GCC (GNU Compiler Collection) kullandım. Karşılaştığım bazı hataları çözerken GDB (GNU Debugger) kullandım. Projemin gelişimini ve versiyonlarını takip edebilmek için de Git versiyon kontrol sisteminden faydalandım.

# 4. Kod Bilgisi

## 4.1. Akış Diyagramı

Kısım ektedir. [1](akis_diyagrami.jpg)

## 4.2. Algoritma

Bu kısımda projenin genel algoritmasına açıklık getireceğim.

Program çalıştığında olduğu klasördeki 'sehirler.txt' dosyasını açıyor. Eğer dosya mevcut değilse ilgili hatayı yazdırıp programdan çıkıyor. Dosya mevcut ise programın genelinde kullanılmak üzere bir şehir listesi (*struct sehirDugum *list)* tanımlıyor. Ardından açılan 'sehirler.txt' dosyasındaki tüm satırları okuyup, satırlardan ana şehir bilgilerini çıkarıp daha önce tanımlanan *list* adresine ekliyor. Eğer şehirlerin herhangi bir bilgisi eksik ya da hatalı ise ilgili hata mesajını yazdırıp programdan çıkılıyor.

Şehirleri listeye eklerken dikkat edilmesi gereken uygun sıra ile eklenmesi. Projede istendiğine göre listedeki şehirler plaka sırasına göre ardışık eklenmiş olmalıdır. Bunu sağlamak için de bir *sehirEkle* fonksiyonu tanımladım. Bu fonksiyonun içinde yeni bir şehir eklenirken tüm şehirleri gezip eğer yeni eklenen şehrin plakası gezilen şehrin plakasından küçükse önüne ekliyorum, eğer tüm şehirler gezilip listenin sonuna gelindiyse listenin sonuna ekliyorum. Bu işlemi yaparken de listede hiç şehir olup olmadığı veya gezilen şehrin önünün listenin başlangıcı olup olmadığı gibi ekstra koşulları da kontrol ediyorum. Değerlendirdiğim koşullar detaylı şekilde ekteki akış diyagramında bulunabilir.

Şehirler listeye eklendikten sonra tüm satırları tekrar gezip bu sefer şehirlere ait komşuları ekliyorum. Komşu eklenirken yine plaka sırasına göre ardışık şekilde eklenmesi gerektiğinden şehir eklerken ki algoritmanın aynısını uyguluyorum.

Şehirler ve komşular listeye eklendiğinde geriye isterler kalıyor. İsterleri kullanıcıya sunacağım arayüz için *switch* yapısı kullandım, kullanıcının girdiği seçime göre işlemleri gerçekleştirdim. Neredeyse her ister için farklı bir fonksiyon tanımlayarak programın ileride geliştirilebilir olmasını ve düzgün bir kod yapısına sahip olmasını sağladım.

## 4.3. İstatistik

Program kodu boşluksuz ve yorumsuz yaklaşık 773 satırdan oluşmaktadır. Kod düzenini sağlamak için yaklaşık 115 boş satır kullanılmıştır. Okuyucuya izlenim oluşturması için yaklaşık 169 yorum satırına yer verilmiştir.

Kullandığım standart kütüphaneler ve ne için kullandığım kabaca aşağıdaki gibidir:

*<stdio.h>, <stdlib.h>*

Temel işlemler için. (printf, malloc..)

*<string.h>*

String fonksiyonları için. (strlen, strchr..)

*<stdbool.h>*

Doğru-yanlış (true-false) veri tipini kullanabilmek için.

*<unistd.h>,<fcntl.h>*

Dosyaya yazma işlemi yapılırken *stdout stream*ini dosyaya yöneltmek için. Detaylı anlatım 2. bölüm sonunda bulunmaktadır.

# Kaynakça


1. C'de dosya açmak, yazmak: https://www.tutorialspoint.com/cprogramming/c_file_io.htm
2. *strtok, strchr* fonksiyonları kullanımı: https://www.geeksforgeeks.org/strtok-strtok_r-functions-c-examples/ https://www.tutorialspoint.com/c_standard_library/c_function_strchr.htm
3. *dup, dup2* fonksiyonları kullanımı: https://www.geeksforgeeks.org/dup-dup2-linux-system-call/
4. Karşılaştığım çeşitli diğer problemler: https://stackoverflow.com/