**Bağlı Liste Uygulaması***Yağız Türer, Hüseyin Yılmaz*

Bilgisayar Mühendisliği Bölümü

Kocaeli Üniversitesi

[yagizturer@gmail.com](mailto:yagizturer@gmail.com), [ylmzhsyn98@gmail.com](mailto:ylmzhsyn98@gmail.com)

***Özet***

Bu projede bağlı liste ve dinamik bellek yapısından faydalanarak şehirlerin plaka, isim, bölge ve komşuluk bilgilerini saklayan bir coğrafi bilgi sistemi altyapısı tasarlamamız istenmiştir. Sistem hazır bulunan bir text dosyasını kıyas alıp bu bilgiler üzerinden işlemler yapar.

**Giriş**

Bu çalışmada bizden şehirlerin plaka, isim, bölge ve komşuluk bilgilerini saklayan bir coğrafi bilgi sistemi tasarlamak amaçlanmıştır.

Şehirler kendi arasında bir bağlı listede, komşular ise kendi aralarında başka bir bağlı listede tutulmuştur. Şehirleri komşularıyla bağlamak için her şehrin düğümüne ilk komşusu bağlanmıştır.

Şehirler kendisinden önceki ve sonraki şehrin adresini saklarken, komşular ise yalnızca kendinden sonraki komşunun adresini saklamaktadır. Projede bizden isim ve plaka ile şehir ve komşu bilgisini aramamız, yeni şehirler ekleyebilmemiz, mevcut şehirlerin adresine ulaşarak birbirlerine komşu olarak bağlayabilmemiz, istenen şehirleri silebilmemiz, komşuluk ilişkilerini kaldırabilmemiz, maksimum ve minimum komşu sayısına göre şehirleri listeleyebilmemiz, önceden belirtilen bir bölgede yer alan tüm şehirleri ekrana yazdırabilmemiz, anlık olarak bağlı liste yapısını bir text dosyasında çıktı olarak saklamamız istenmiştir. Ayrıca bonus olarak da belirli sayı aralığında komşuya sahip olan ve belirli ortak komşulara sahip olan şehirleri listeleyebilmemiz ve tüm bu işlemleri tekrar tekrar bir menüden seçimini yaparak gerçekleştirmemiz istenmiştir. Başta mevcut bir text dosyasından şehir ve komşuluk bilgileri okunmakta ve tüm bu işlemler o bilgilere ekleme veya çıkarma yapılarak gerçekleştirilmektedir. Bu çalışmada bizden standart C dilini kullanmamız istenmiştir.

**Yöntem**

Şehir düğümlerini tutan structta isim, plaka, bölge, önceki düğümün adresi, sonraki düğümün adresi ve ilk komşunun adresi saklanmakta, komşu düğümlerini tutan structta ise komşunun plakası ve sonraki komşunun adresi saklanmaktadır. Kodun başında bir ilk düğüm oluşturularak bu düğüme malloc ile bellekten yer tahsis edilmekte, bağlandığı tüm adresler NULL’a eşitlenmekte ve şehir adı daha sonra hiç şehir eklenip eklenmediğini kontrol etmek için “bos” yapılmaktadır. Daha sonraki tüm fonksiyonlara bu ilk oluşturulan düğüm gönderilerek bağlı listede istenen düğümlere ulaşılmaktadır.

Öncelikle bağlı listemizin anlık halini bir çıktı dosyasına yazdırmak için listeyazdır fonksiyonu oluşturduk. Bu fonksiyon son düğüme gelene kadar tüm şehirlerin bilgilerini ve son komşuya gelene kadar komşularını yazdırmaktadır.

Başta dosyadan okurken yalnızca şehirlerin bilgilerini aldık ve komşuları atladık. Daha sonra tüm şehirlerin plakaları belleğe alındıktan sonra komşu okuma işlemini gerçekleştirdik. Bu okuma işlemi sırasında dosyadan komşu adları alınmakta, komşu düğümlerine bu adla uyuşan plakalar kaydedilmektedir. Tüm bu dosyadan okuma işlemleri bittikten sonra listeyazdır fonnksiyonu çalışarak bağlı listemiz çıktı dosyasına kaydedilmekte ve kullanıcıya menü gösterilerek yapmak istediği işlemler sorulmaktadır. Menü 0 tuşuna basılana kadar her işlemin ardından tekrar ekrana gelmektedir. Her bir işlem seçiminden sonra ekran temizlenerek görüntü kirliliğinden kaçınılmıştır. Bu komutun hem Linux hem de Windows’ta çalışması için system("clear||cls"); şeklinde yazılmıştır.

Yeni şehir eklenmek istediğinde kullanıcıdan bölge adı, şehir adı, plaka bilgisi ve komşularının plakaları alınmakta, bu bilgiler yenikomsuekle fonksiyonuna yollanmakta, eğer aynı ad veya plaka bilgisine sahip bir şehir yoksa listemize eklenmektedir. Yeni komşu eklenmek istediğinde ise bağlanmak istenen şehirlerin plakaları alınmakta, iki şehir sırasıyla birbirinin komşularına eklenmektedir.

Kullanıcı isterse plaka veya şehir adı girerek bir şehrin ve komşularının bilgilerini yazdırabilmektedir. Bunun için bağlı listede girilen bilgiyle uyuşan bir şehir aranmakta, eğer yoksa bu şehri listeye eklemek isteyip istemediği sorulmaktadır. Kullanıcı yeni bir şehir ve komşuluk bilgisi ekleyebildiği gibi, istediği takdirde mevcut şehir ve komşuluk bilgilerini silebilmektedir. Bir şehir silindiği zaman o şehri komşu olarak komşu listesinde tutan tüm şehirlerden de plakası silinmektedir. Daha sonra bu şehrin bilgilerini tutan düğüm free komutu kullanılarak bellekten bütünüyle temizlenmektedir.

Bölgeye göre şehir listeleme işleminde ise tüm şehirlerde gezilmekte ve bölge kodu uyuşan şehirler ekrana yazdırılmaktadır.

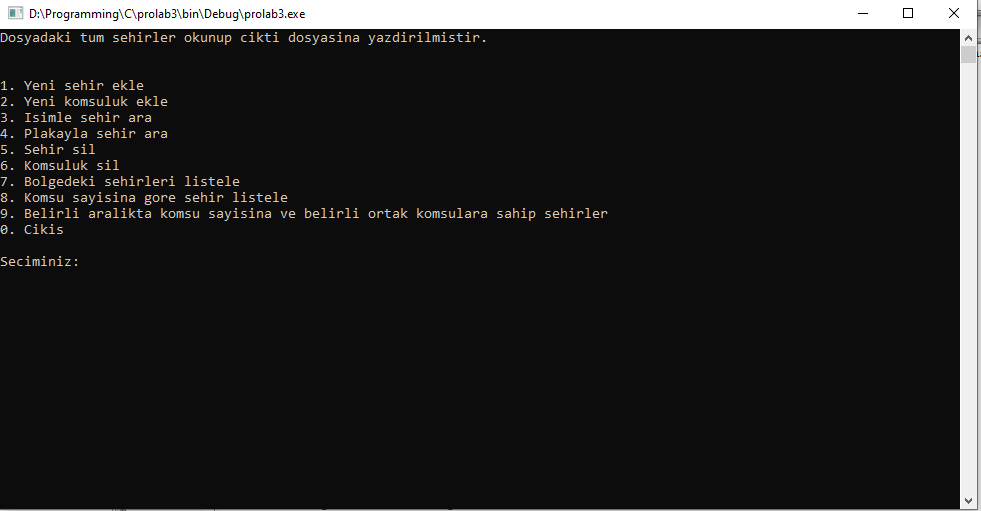
Liste üzerinde herhangi bir değişiklik yapılmasını gerektiren fonksiyonlara ilk düğüm pointer olarak yollanırken, değişiklik yapmayacak olan fonksiyonlara normal değişken olarak yollanmıştır. Böylece her fonksiyon sonunda ilk düğümü geri döndürme ihtiyacından kaçınıldı.

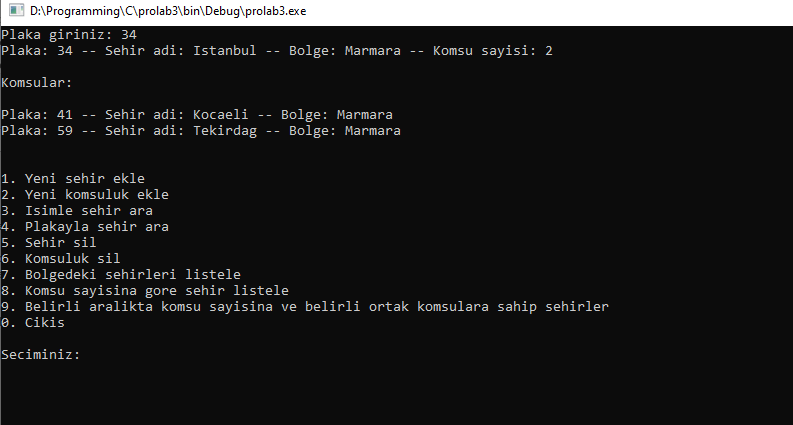
Kullanıcıdan bir string girmesi istendiğinde programda hatalar olmaması için buffer temizlendi. Tüm şehir ekleme, şehir silme, komşu ekleme ve komşu silme işlemlerinde eklenecek veya silinecek düğümün listenin başında veya sonunda konumlanabileceği hesaba katıldı. Böylece hem silme hem de ekleme işlemlerinde düğümler sıralı şekilde işleme alınarak daha sonra ayrıyeten sıralama yapma ihtiyacı ortadan kaldırıldı. Projede bonus olarak bizden belirli bir sayı aralığında komşuya sahip ve belirli komşulara sahip şehirleri listelememiz istenmişti, bunun için kontrol koyuldu, herhangi bir kriter sağlanmıyorsa kontrol 0’a eşitlendi ve tüm kontroller sonunda değişkenin değeri 1 kalan düğümler ekrana yazdırıldı. Şehirlerin komşu sayıları başta 0’a eşitlendi, ekleme veya çıkarma işlemlerine artırılıp azaltıldı.

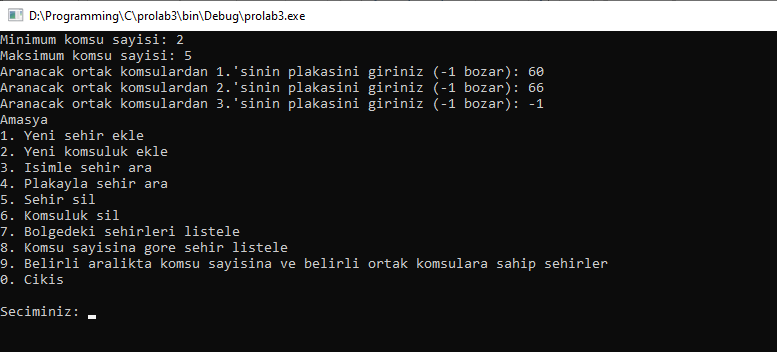
**Sonuçlar**

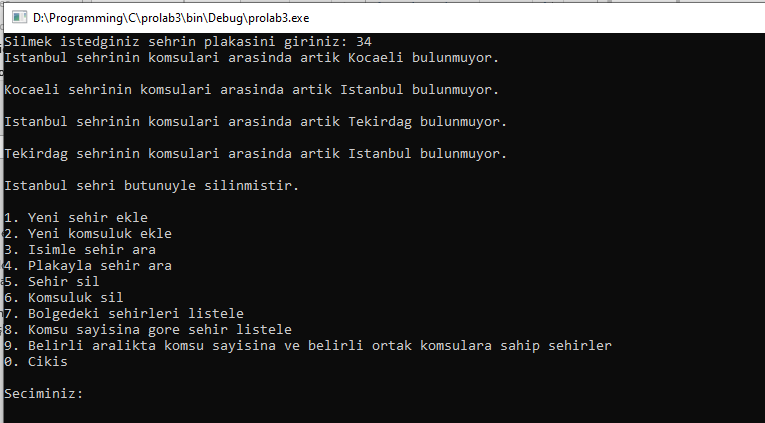
Bizden istenen tüm isterler gerçekleştirilmiştir. Dosyadan tüm şehirlerin ve komşularının bilgileri başarıyla belleğe aktarıldı, bu bilgiler üzerine ekleme veya çıkarma yapılabilmesi sağlandı, bu bilgilere istenildiği gibi ulaşılması sağlandı. Proje sonucunda bağlı liste yapısının kullanımı, dinamik bellek tahsisi, pointer ile fonksiyon kullanımı öğrenildi. Gerekli testler yapılarak programın runtime hataları vermesi önlendi.

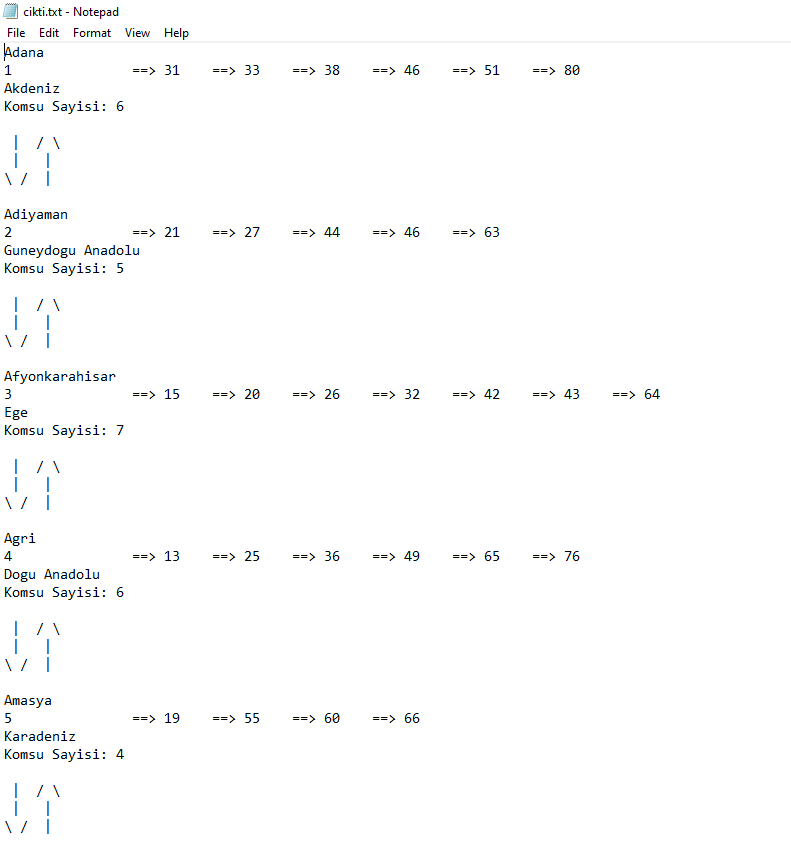
**Çıktılar**

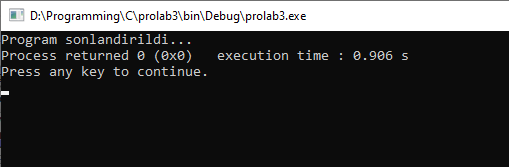
****

****

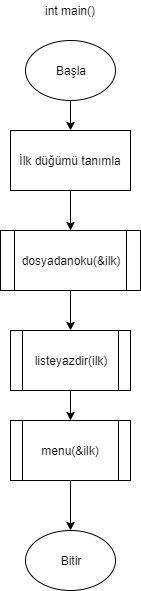
****

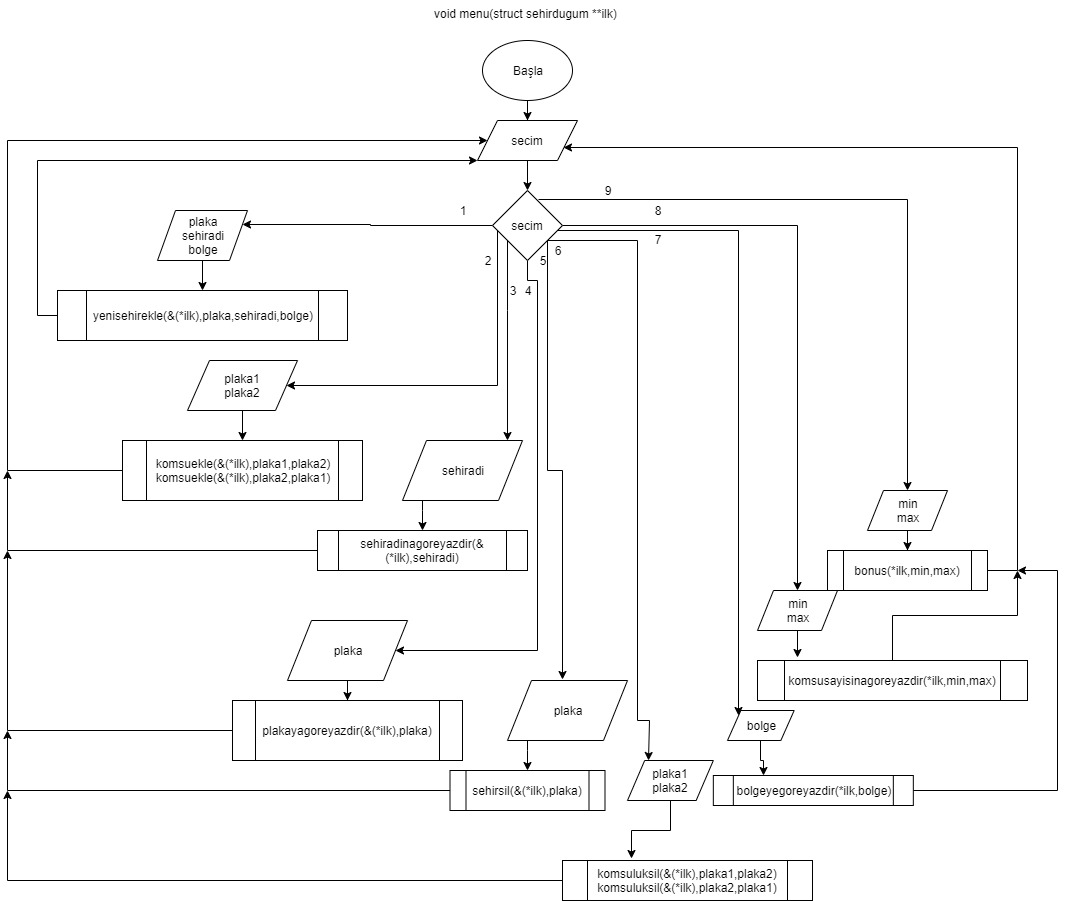
****

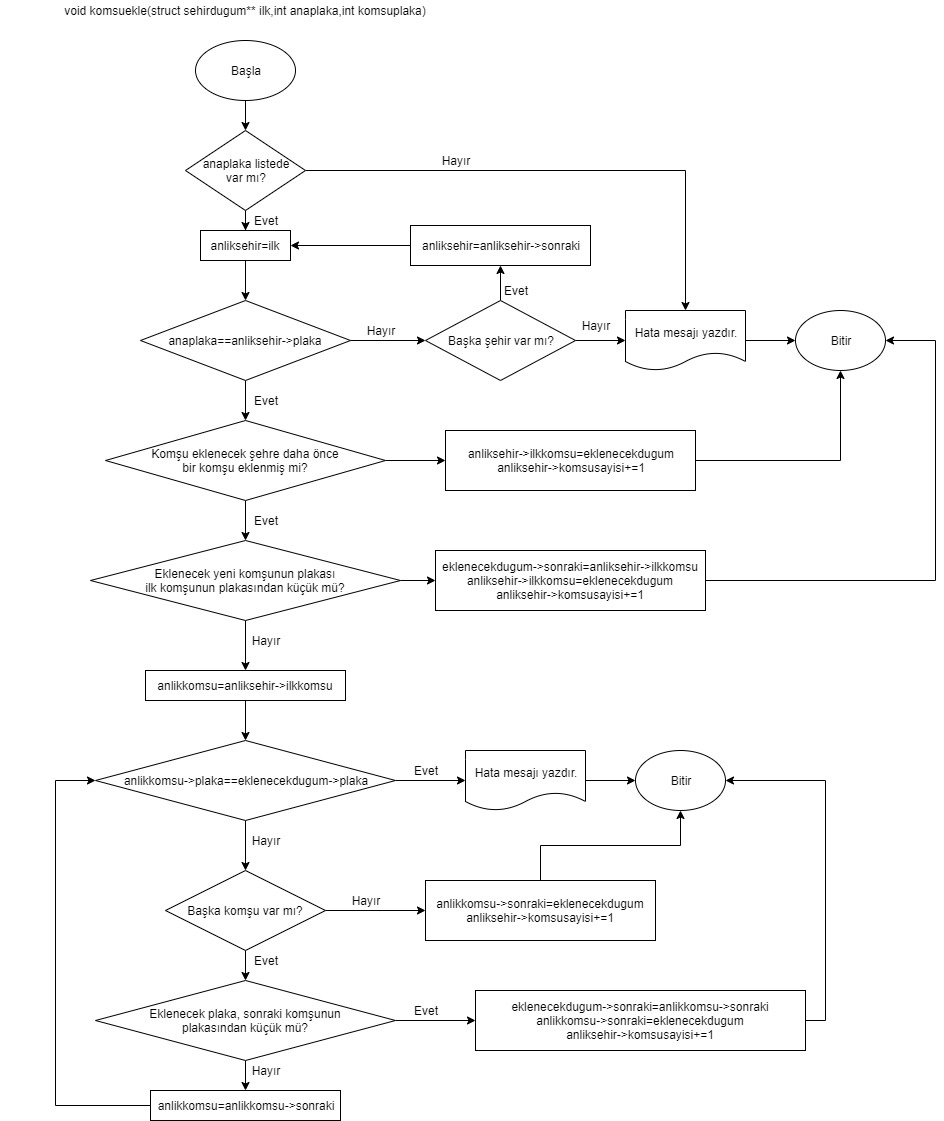
****

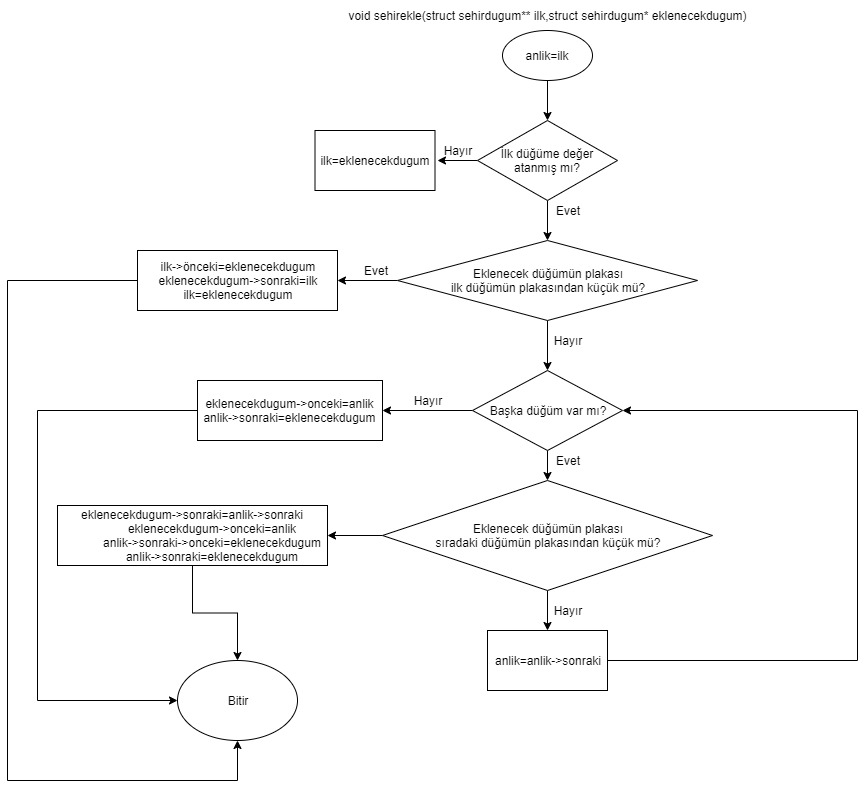
****

**Akış Şemaları**

****

****

****

****

**Kaynaklar**

<https://www.geeksforgeeks.org/structures-c/>

<https://www.tutorialspoint.com/cprogramming/c_structures.htm>

<https://www.geeksforgeeks.org/dynamic-memory-allocation-in-c-using-malloc-calloc-free-and-realloc/>

<https://www.tutorialspoint.com/c_standard_library/c_function_malloc.htm>

<https://www.geeksforgeeks.org/linked-list-set-1-introduction/>

<https://www.learn-c.org/en/Linked_lists>

<https://www.tutorialspoint.com/cprogramming/c_pointers.htm>

<https://www.programiz.com/c-programming/c-pointers>

<https://www.tutorialspoint.com/cprogramming/c_file_io.htm>

<https://www.tutorialspoint.com/c_standard_library/string_h.htm>

<https://www.geeksforgeeks.org/strtok-strtok_r-functions-c-examples/>

<https://www.geeksforgeeks.org/fprintf-in-c/>

<https://www.geeksforgeeks.org/clearing-the-input-buffer-in-cc/>

<https://www.learn-c.org/en/Linked_lists>

<https://www.programmingsimplified.com/c/data-structures/c-program-implement-linked-list>

<https://www.thegeekstuff.com/2012/08/c-linked-list-example/>

Onur Gök Veri Yapıları ve Algoritmalar ve Programlama Slaytları