BAĞLI LİSTE İLE KELİME SAYMA

Teyfik CANER – Hasan ÇOLAK
Mühendislik Fakültesi
Bilgisayar Mühendisliği
Kocaeli Üniversitesi
teyfikcaner@gmail.com - hasancolk99@gmail.com

Özet- Bu projede amaç text dosyası içerisinde bulunan kelimeleri sayarak, adet sayısına göre büyükten küçüğe doğru olacak şekilde bağlı listeye eklemektir.

Abstract-The goal of this project is, counting the words in the text file and to add to the list by sorting in order from the highest to the lowest.

I. PROBLEM TANIMI

Bu proje de text dosyasındaki kelimeleri sırası ile saydırarak adet sayısını hesaplatırız hesaplanan adet sayısı ve kelimeyi bağlantılı liste yapısına adet sayısını göz önünde bulundurarak ekleriz bu ekleme işleminde sayılan kelime direkt eklenmeli bütün kelimeleri saydırdıktan sonra ekleme işlemi yapılmamalıdır.

II. GİRİŞ

Tasarlanacak olan, bağlı liste ile kelime sayma uygulamasında , text dosyası içerisinden okunan kelimeler , adet sayısına göre sıralı bir şekilde bağlı listeye eklenecektir. Bağlı liste ; kelime ve adet bilgilerini içermeli , her kelime yalnız bir kez bulunmalıdır. Sıralama işlemi , tüm kelimelerin adetlerini hesaplayıp en son toptan bir şekilde yapılmamalı ; her bir yeni kelime , metin içerisinde aratılıp , metinde kaç defa kullanıldığı hesaplandıktan sonra bağlı listeye sıralı bir şekilde ekleyerek yapılmalıdır. Bağlı listeye ekleme yapılırken , kelimenin ekleneceği yere göre ; başa , sona ve araya ekleme metotları mutlaka kullanılmalıdır. Problemin çözümünde dizi kullanılmaması , bağlantılı liste kullanılarak çözülmesi istenmiştir.

III. YAPILAN ARAŞTIRMALAR

Bağlı liste kısmında pek bir sorunla karşılaştığımız söylenemez , konu bilgisi eksikliğinden kaynaklanan bazı sorunlarımız vardı , onu da Onur Gök hocanın E-DESTEK üzerindeki ders kayıtlarından ve Sadi Evren Şeker hocanın Youtube'daki ders videolarından faydalanarak hallettik. Dosyadan kelimelerin okunması ve pointer konularında da eksiklerimiz oldu. Bu eksikleri de Palme Yayınevi'ne ait "Programlamayı C ile Öğreniyorum" kitabından yararlanarak hallettik.

IV. GENEL YAPI

Bağlı listeyi "node" adını verdiğimiz struct yapılarıyla oluşturduk. Bu yapının içerisinde ; string türünde "kelime" değişkeni , int veri tipinde "adet" değişkeni , struct n* türünde "next" değişkeni bulunmaktadır. "kelime" değişkeni , dosyanan okunan kelimeleri ; "adet" değişkeni , kelimenin dosyada kaç defa geçtiğini ; "next" değişkeni , adet sayısına göre sıralanmış bağlantılı listede , kendisinden sonra gelecek kelimeye ait düğümün adresini tutar. Main içerisinde kullanılmak üzere "adet_bul" , "yazdir" , "basa_ekle" , "araya_ekle" , "sona_ekle" , "sirali_ekle" metotlarını yazdık.

"adet_bul" metodu , parametre olarak aldığı stringden dosyada kaç adet bulunduğunu int tipinde dönderir. Bu hesaplamayı yapabilmek için , "metin" adında bir file pointer oluşturulur. Bu pointer txt dosyasının sonuna kadar okuma yapacak şekilde döngüye sokulur. "kelime_2" adında bir string oluşturulur. Her döngü başlangıcında "kelime_2" stringine boş değer atanır , "metin" pointerı dosyadan okuduğu her kelimeyi strcpy fonksiyonunu kullanarak "kelime_2" stringinin içine atar. "kelime_2" , "metin" pointerından değer aldıysa , yani "kelime_2" nin içine boş değilse , önce kelimenin tüm harfleri küçültülür daha sonra , parametre olarak gönderilen kelimeyle aynı mı diye sorgulatılır. Eğer aynıysa adet sayısı 1 arttırılır. Bu döngü , dosyadaki tüm kelimelerle , parametre olarak gönderilen kelime kıyaslanana dek devam eder. En sonunda adet sayısı hesaplanıp dönderilmiş olur.

"basa_ekle" metodu ; gönderilen kelimeyi , bağlı listenin başına ekler. Parametre olarak ; node pointer türünde , string türünde ve int türünde olmak üzere 3 adet değişken alır. node pointer türündeki değişken , bağlantı listesinin başlangıç adresini ; string türündeki değişken , bağlı listeye eklenecek olan kelimeyi ; int türündeki değişken ise eklenecek olan kelimenin adet sayısını tutar. Ekleme işlemi için öncelikle yeni bir düğüm oluşturulur ve bu düğümün "kelime" değişkenine , parametre olarak gönderilen kelimeyi , "adet" değişkenine ise parametre olarak gönderilen adet sayısını atarız. Daha sonra düğümün "next"ine bağlı listenin başlangıç adresini atarız ve artık bağlı listenin başlangıç adresini yeni oluşturduğumuz düğüm olduğu için yeni düğümümüzün adresini geri döndeririz.

"sona_ekle" metodu ; gönderilen kelimeyi , bağlı listenin sonuna ekler. Parametre olarak ; node pointer türünde , string türünde ve int türünde olmak üzere 3 adet değişken alır. node pointer türündeki değişken , eklenecek olan kelimenin hangi düğümden sonra ekleneceğini ; string türündeki değişken , bağlı listeye eklenecek olan kelimeyi ; int türündeki değişken ise eklenecek olan kelimenin adet sayısını tutar. Ekleme işlemi için öncelikle yeni bir düğüm oluşturulur ve bu düğümün "kelime" değişkenine , parametre olarak gönderilen kelimeyi , "adet" değişkenine ise parametre olarak gönderilen adet sayısını atarız. Daha sonra parametre olarak aldığımız düğümün "next"ine yeni oluşturduğumuz düğümü ekleriz. Yeni düğümümüzün "next"ine de NULL ekleyerek bağlı listenin son düğümü yapmış oluruz.

"araya_ekle" metodu ; gönderilen kelimeyi , bağlı listede istenilen düğümün sonrasına ekler. Parametre olarak ; node pointer türünde , string türünde ve int türünde olmak üzere 3 adet değişken alır. node pointer türündeki değişken , eklenecek olan kelimenin hangi düğümden sonra ekleneceğini ; string türündeki değişken , bağlı listeye eklenecek olan kelimeyi ; int türündeki değişken ise eklenecek olan kelimenin adet sayısını tutar. Ekleme işlemi için öncelikle yeni bir düğüm oluşturulur ve bu düğümün "kelime" değişkenine , parametre olarak gönderilen kelimeyi , "adet" değişkenine ise parametre olarak gönderilen adet sayısını atarız. Daha sonra yeni oluşturduğumuz düğümün "next"ine , parametre olarak aldığımız düğümün "next"ini atarız. Parametre olarak aldığımız düğümün "next"ine de yeni düğümümüzün adresini atarız. Böylece yeni oluşturduğumuz düğümü , birinci parametrede verilen düğümün bir sonrasına ekleyerek araya yerleştirme işlemini tamamlamış oluruz.

"sirali_ekle" metodu ; gönderilen kelimeyi , adetine uygun olacak biçimde ; "basa_ekle" , "sona_ekle" veya "araya_ekle" metotlarından birini kullanarak bağlı listeye sıralı olarak ekler. Parametre olarak ; node pointer türünde , string türünde ve int türünde olmak üzere 3 adet değişken alır. node pointer türündeki değişken , bağlantı listesinin başlangıç adresini ; string türündeki değişken , bağlı listeye eklenecek olan kelimeyi ; int türündeki değişken ise eklenecek olan kelimenin adet sayısını tutar. Fonksiyon ilk olarak gönderilen bağlı listenin boş olup olmadığını sorgulatır. Eğer boşsa , parametre olarak gönderilen kelime bağlı listenin ilk düğümünü oluşturur ve geri dönderilir. Bağlı liste boş değilse , yani daha önceden düğüm oluşturulmuşsa ikinci sorgu işlemine geçilir. Burada , gönderilen kelimenin bağlı listenin ilk düğümüne ait kelime olup olmadığı sorgulatılır. Eğer bağlı listenin ilk düğümüne eşitse hiçbir işlem yapılmadan geri dönderilir. Eğer eşit değilse "iter" adında bir node pointer değişkeni oluşturulur. Bu değişkene , bağlı listenin başlangıç adresi atanır. Daha sonra döngüye sokularak , eklenecek olan kelimenin bağlı listede bulunup bulunmadığı sorgulanır. Kelime , bağlı listede varsa hiçbir işlem yapılmadan geri döndürülür. Kelime , bağlı listeye daha önce eklenmemişse nereye ekleneceğini tespit edilir. İlk önce adet sayısı , bağlı listenin başlangıcındaki kelimenin adet sayısıyla karşılaştırılır. Eğer eklenecek kelimenin adet sayısı başlangıçtaki kelimenin adet sayısından fazla ise "basa_ekle" metodu kullanılarak yeni kelime bağlı listenin başlangıcına eklenir. Eğer başlangıçtaki kelimeden fazla değilse , döngüye sokularak hangi düğümün sonrasına ekleneceği belirlenir. "next"i NULL olan düğümün sonrasına eklenmesi isteniyorsa sona eklenecek demektir. Bu durumda "sona_ekle" metodu kullanılarak yeni kelime bağlı listenin sonuna eklenir. Diğer durumda ise eklenecek kelime başa ve sona eklenmediğine göre son ihtimal araya eklenir. "araya_ekle" metodu kullanılarak bu işlem gerçekleştirilir.

"yazdir" metodu ; kendisine gönderilen bağlı listenin başlangıç adresinden , bağlı listenin sonuna kadar giderek bağlı listedeki tüm kelimeleri adet sayısıyla birlikte ekrana yazdırır.

Yukarıdaki metodları açıkladığımıza göre programın işleyişine geçebiliriz. İlk başta bağlı listenin başlangıcı olarak kabul edeceğimiz , node pointer türünde "root" adında bir değişken oluştururuz. Bu değişkene başlangıç için NULL değeri atanır. Dosyadan kelime okumak maksadıyla , "metin" adında bir file pointer değişkeni oluştururuz. "metin" pointerı , text dosyasını okuma modunda açar. Dosyadan okunan kelimeleri atayabileceğimiz "kelime" adında bir string oluştururuz. Daha sonrasında ise "metin" pointerı , dosya sonuna kadar okuma yapacak şekilde döngüye sokulur. Her döngü başlangıcında "kelime" stringine boş değer atanır , "metin" pointerı dosyadan okuduğu her kelimeyi strcpy fonksiyonunu kullanarak "kelime" stringinin içine atar. "kelime" , "metin" pointerından değer aldıysa , yani "kelime" nin içine boş değilse ; önce kelimenin tüm harfleri küçültülür , "adet_bul" metoduyla adeti hesaplanır , en son da "sirali_ekle" metoduyla bağlı listeye eklenir. Dosyadaki tüm kelimeler okunduktan sonra "yazdir" metoduyla , sıralanmış bağlı liste ekrana yazdırılır.

V. EKSİKLİKLER

Projede tanımlanan probleme uygun çözümü üretip , bizden istenen tüm isterleri eksiksiz olarak yerine getirerek projeyi tamamladık.

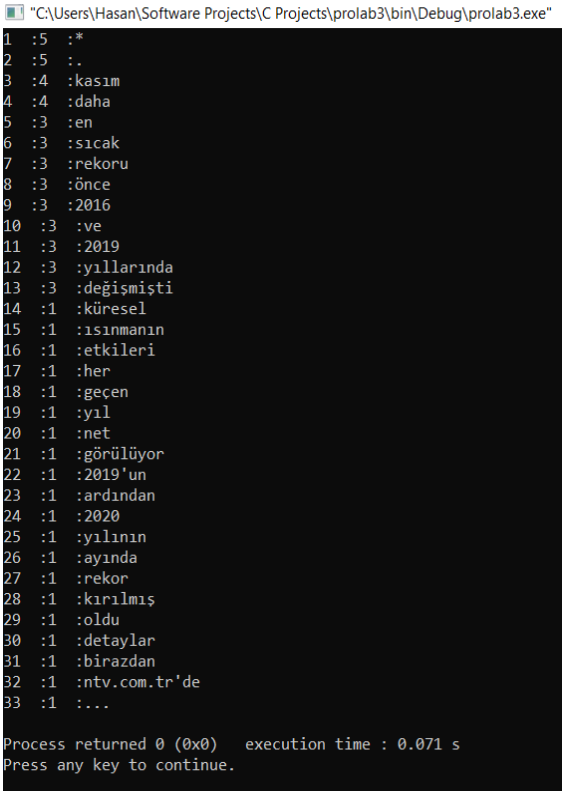
VI. KAZANIMLAR

Projenin oluşturulmasında bağlı liste yapısını kullandığımızdan dolayı ; bağlı listeye eleman ekleme , elemanların yerlerini değiştirme , sıralama gibi birçok konuda işlemler yaparak konuya olan hakimiyetimizi arttırdık. Daha önceden öğrendiğimiz fakat pek fazla uygulama şansı bulamadığımız pointer yapısını da bağlı listede kullanarak işlevini daha iyi kavramış olduk. Stringlerin birbiri içerisine aktarılması ve birbirleriyle karşılaştırılması gibi durumlarda string.h kütüphanesine ait fonksiyonları kullanarak bu kütüphanede daha fazla tecrübe kazanmış olduk. Ayrıca bu projeyle birlikte ilk kez grup çalışmasını deneyimlemiş olduk.

VII. YAKLAŞIMLAR

Bağlı liste içeriğini oluşturduktan sonra , kodun nasıl ilerleyeceğini yaptığımız fikir alışverişleri sonucunda genel hatlarıyla kararlaştırdık. Daha sonra bu ilerleyişe uygun olacak biçimde kullanacağımız metotları belirledik. Metotların oluşturulması sırasında sıkıntı yaşamamak adına konu eksiklerimizi giderdik ve metotların içeriğini tamamladık. Projeyi , farklı durumları göz önünde bulunduracak şekilde defalarca test ettik ve bu sırada oluşan aksaklıkları tespit edip probleme uygun çözüm üreterek projeyi tamamladık.

VIII. GÖRSELLER

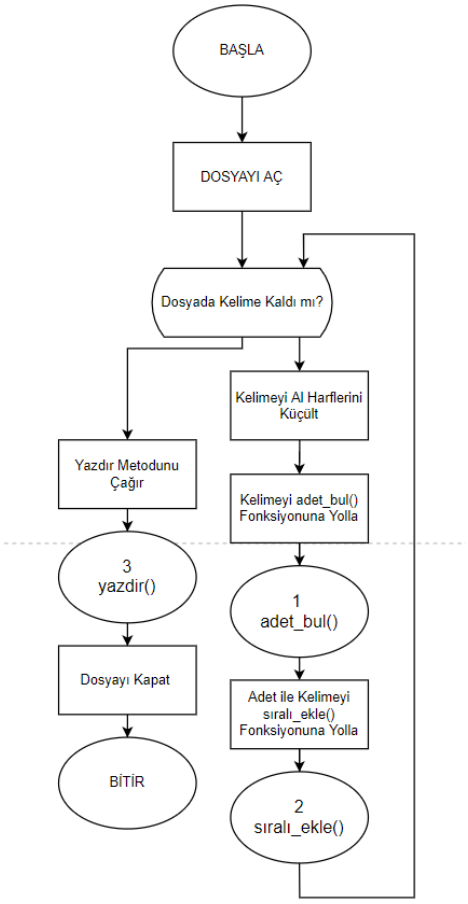


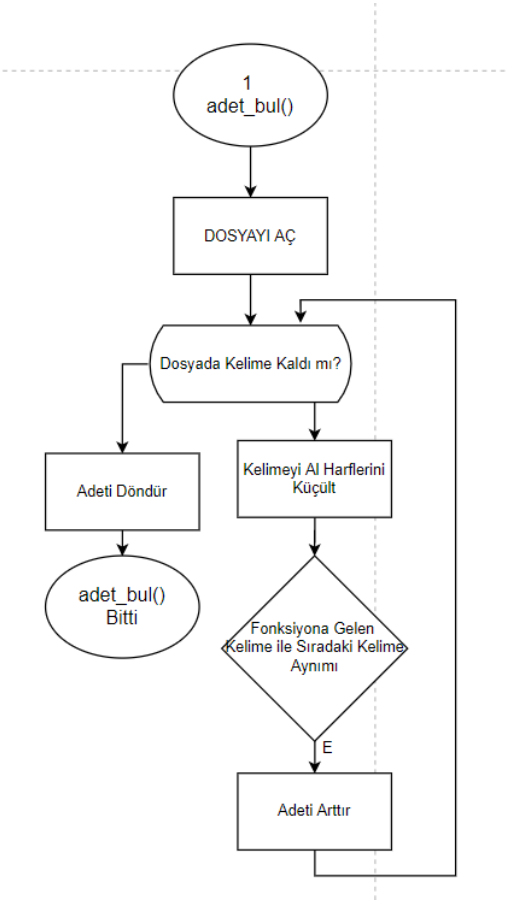
KAYNAKÇA

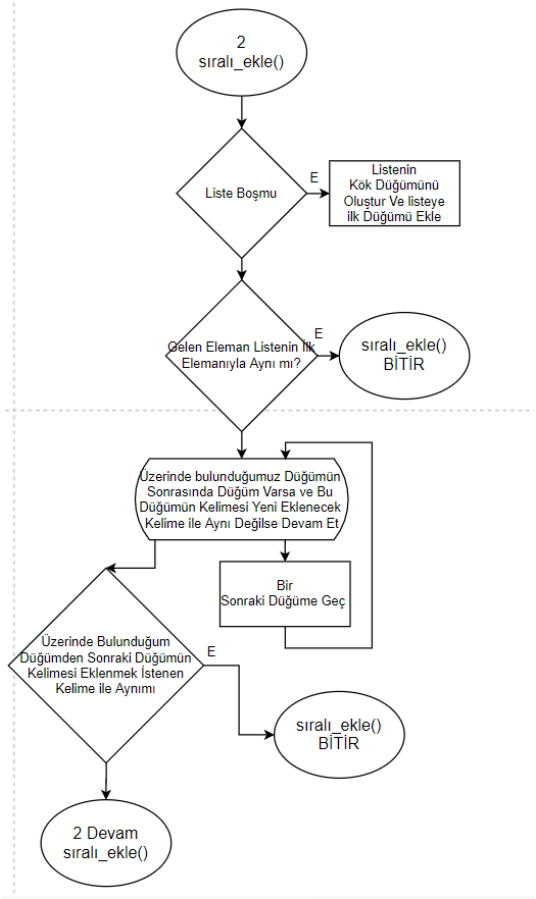
[1]
Muhammet Yorulmaz,Seher Yorulmaz (2018). Programlamayı C ile Öğreniyorum. Ankara: Vadi Grup Basım A.Ş

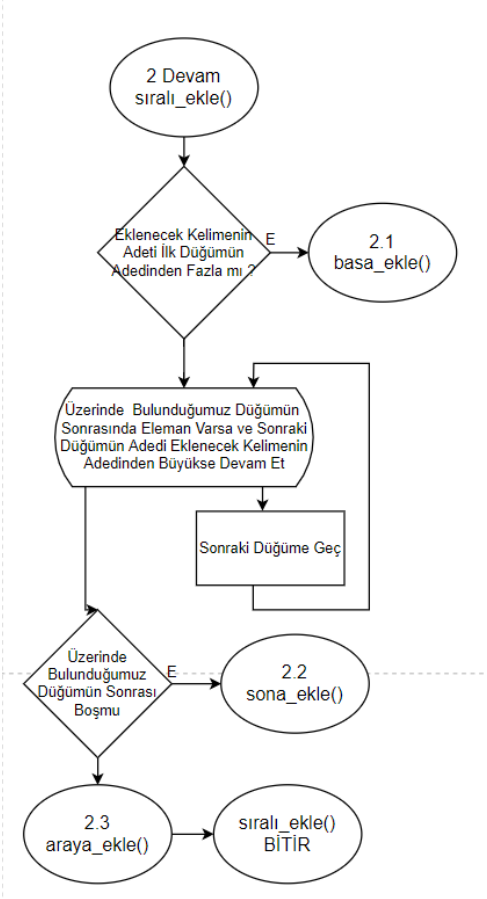
[2]
Şeker,Sadi Evren. "Veri Yapıları".(Acces Date : 28.01.2016)
https://www.youtube.com/playlist?list=PLh9ECzBB8tJN9bckI6FbWB03HkmogKrFT

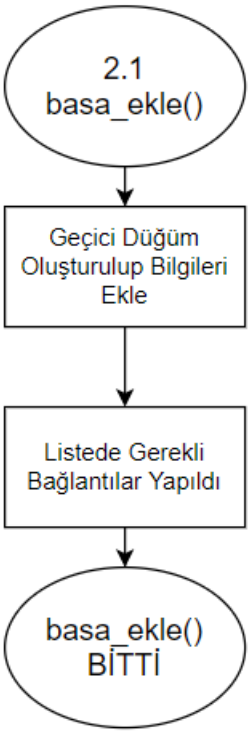
IX. AKIŞ DİYAGRAMLARI

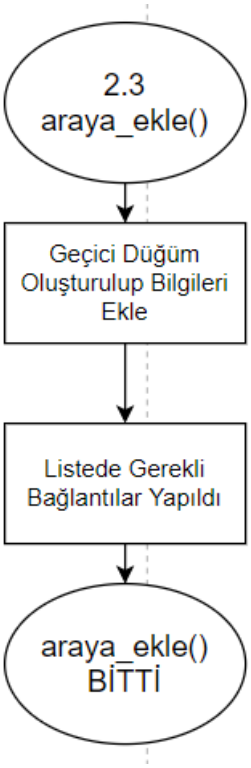


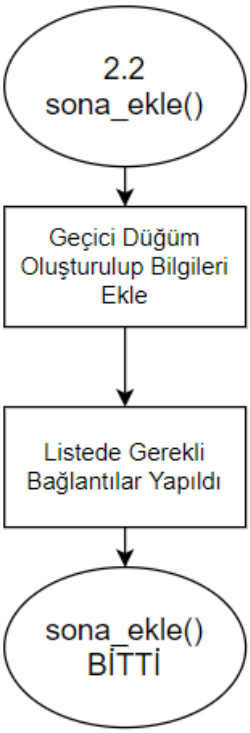


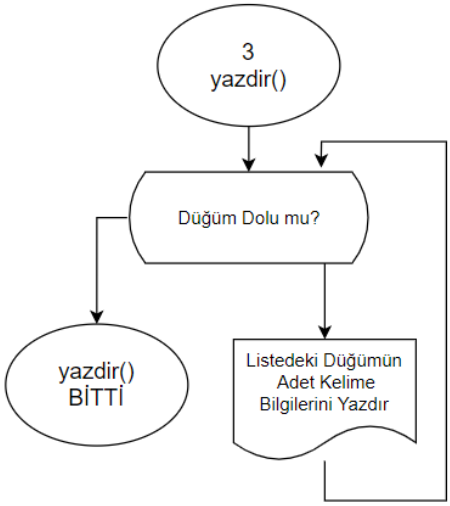





2.1
basa_ekle()
Geçici Düğüm Oluşturulup Bilgileri Ekle
Listede Gerekli Bağlantılar Yapıldı
basa_ekle() BİTTİ


2.3
araya_ekle()
Geçici Düğüm Oluşturulup Bilgileri Ekle
Listede Gerekli Bağlantılar Yapıldı
araya_ekle() BİTTİ


2.2
sona_ekle()
Geçici Düğüm Oluşturulup Bilgileri Ekle
Listede Gerekli Bağlantılar Yapıldı
sona_ekle() BİTTİ


3
yazdir()
Düğüm Dolu mu?
yazdir() BİTTİ
Listedeki Düğümün Adet Kelime Bilgilerini Yazdır