PROGRAMLAMA LABORATUVARI
PROJE 3 RAPORU

KOCAELİ ÜNİVERSİTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ
PROGRAMLAMA LAB.

Muhammed Enbiya Demir - 190202018
enbiyademir39@gmail.com

Çağkan Çağman CERAN - 190202028
190202028@kocaeli.edu.tr

1. Problem Tanımı

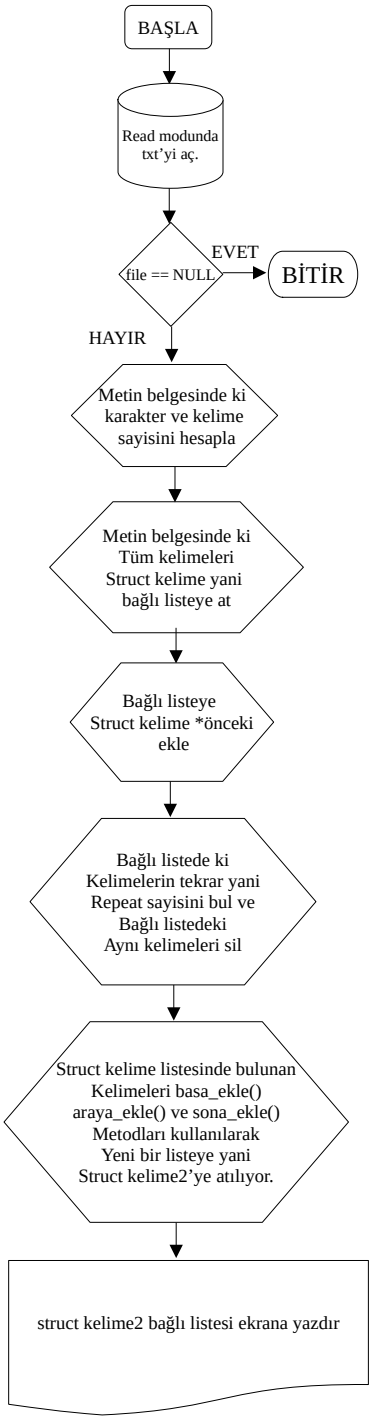
Proje, bağlı liste kullanılarak hiçbir şekilde dizi kullanılmadan bir metin içerisinde ki tüm kelimelerin metin içerisinde kaç defa geçtiğini büyükten küçüğe sıralayarak ekranda gösterilmesi şeklindedir.

2. Yapılan Araştırmalar

Proje için herhangi bir ek araştırmaya ihtiyaç duyulmamıştır. Sadece "veri yapıları ve algoritmaları" dersinde bize öğretilen bilgiler yeterli oldu.

3. Tasarım

3.1 Akış şeması




```
Struct kelime
{
int repeat; //txt de kac kez tekrar ettigi
int start;  //txt de kelimenin baslangic indexi
int end;    //txt de kelimenin bitis indexi
int uzunluk;//kelimenin uzunlugu
struct kelime *sonraki;
struct kelime *onceki;
};
```

```
struct kelime2
{
int repeat; //txt de kac kez tekrar ettigi
int start;  //txt de kelimenin baslangic indexi
int end;    //txt de kelimenin bitis indexi
struct kelime *sonraki;
};
```

3.2 Yazılım Mimarisi

Yapı olarak bilgiler txt dosyasından okunmuş gerekli işlemler döngüler sayesinde yapıldıktan sonra bağlı listede büyükten küçüğe metodlarla düzenlenmiştir.

4. Genel Yapı

Projede hiçbir şekilde dizi kullanılmamıştır. Kelimeleri struct yapısında tutmak için o kelimenin txt belgesinde bulunan başlangıç indexini int start; değişkeninde, bitiş indexini de int end; değişkeninde alındı. Bu şekilde çok kolay bir şekilde dizi kullanmadan kelimemizi konsola yazdırabiliriz. Sadece yapmamız gereken kelimeyi yazdırırken txt de int start; indexine gidip int end; indexine kadar olan her bir karakteri yazdırmak kelimemizi ekranda göstermiş olacaktır.
Aynı şekilde kelimemizin metinde kaç defa geçtiğini bulabilmemiz için kelimemizi diğer kelimeler ile karşılaştırmamız gerekir.

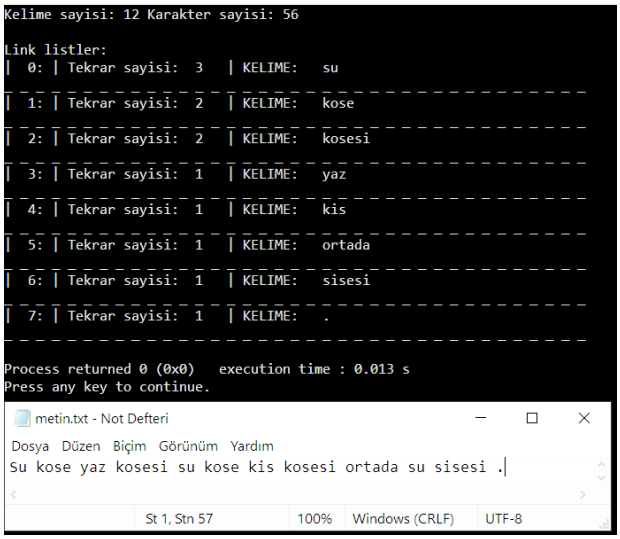
Öncelikle yazdığımız programda karşılaştıracağımız kelimenin uzunluğu elimizdeki mevcut kelimenin uzunluğuna eşit değilse doğal olarak bu karşılaştıracağımız kelimeler farklı kelimeler olacaktır. Eğer uzunlukları aynı ise bu sefer her iki kelimenin ilk karakterinden başlayıp teker teker her bir karakterini karşılaştırır.

Program karşılaştırırken farklı iki karakteri(char) karşılaştırdığını gördüğü anda break; ile karşılaştırmayı bırakır ve yine farklı kelimeler olduğuna karar verir ardından sonraki kelimeye geçer. Eğer tüm karakterleri(char) aynı ise ikisi de aynı kelimedir. Bu durumda struct yapısında bulunan int repeat'i bir arttırır ve karşılaştırdığı kelimeyi link list yapısından siler çünkü struct yapısında birden fazla aynı kelimenin olmasını istemiyoruz.

Tüm bu işlemlerin ardından elimizde birbirinden farklı kelimelerin olduğu ve bu kelimelerin repeat sayısının olduğu bir link list elde etmiş oluyoruz. Fakat bu link list yapısında kelimeler repeat sayısına göre büyükten küçüğe doğru link liste yerleşmemiş olacaktır. Bunu yapmak için ikinci bir link liste ihtiyaç duyuyoruz. Birinci link listten ikinci link liste basa_ekle(), sona_ekle(), araya_ekle() metotları sayesinde ikinci link liste yerleştirmiş oluyoruz. Burada bir şeye dikkat çekmek istiyorum. Bu metotlar "veri yapıları ve algoritmaları" dersinde link list mantığını anlatmak için derste gösterilmiştir. Bizde dersteki metotların aynısını almış bulunmaktayız.

Sadece doğal olarak değişkenler için değişiklikler yapılmıştır. Yani burada farklı bir araya_ekle(), basa_ekle(), sona_ekle()'den bahsetmiyoruz. Bütün bunların ardından ikinci bağlı listeye repeat sayılarına göre listelenmiş olacaktır. En son olarak ikinci listeyi ekrana yazdırarak program sonlanıyor.

4.1 Ekran Görüntüsü



5. Referanslar

Kocaeli Üniversitesi Veri Yapıları Ve Algoritmaları Dersi Notları

Web sitesi

http://embedded.kocaeli.edu.tr/?page_id=91