# BAĞLI LİSTE UYGULAMALARI PROJESİ

İrem ÇELİKKANAT
Kocaeli Üniversitesi
Mühendislik Fakültesi
Bilgisayar Mühendisliği
*190202124@kocaeli.edu.tr*

Eliz KURTULUŞ
Kocaeli Üniversitesi
Mühendislik Fakültesi
Bilgisayar Mühendisliği
*190202015@kocaeli.edu.tr*

## I. Problem Tanımı

Bu proje genel olarak bağlı liste yapısını kullanarak verilen kelimeler.txt dosyasından veriyi çekerek bu verileri bağlı listeler yardımıyla sıralı yapıda tutmayı sağlar. Sonucunda veriler bağlı listeye büyükten küçüğe sıralı bir şekilde yüklenmiş bulunmaktadır.

### A. TEMEL BİLGİLER

Program C programlama dilinde gerçekleştirilmiş olup geliştirme ortamı olarak CodeBlocks kullanılmıştır.

### B. TASARIM

### A. Algoritma

Bağlı liste uygulamaları projesi kelime verilerini sistemde sıralı yapıda tutar. Text içerisindeki kelimeleri sayarak bağlı listeye adet olarak büyükten küçüğe doğru eleman ekler. Uygulamada her seferinde yeni bir kelimeye göre metin içerisinde arama yapar ve metin içerisinde geçen kelimeleri sayar. Uygulamada işlem sırası metin içerisinde kelimeyi bulup, adet sayısını hesaplayıp, büyükten küçüğe doğru bağlı listeye eklenmesi şeklindedir. Bağlı liste kelime ve adet bilgisi içerir. Bağlı liste içerisinde bir kelime sadece bir kez bulunur. Bağlı liste büyükten küçüğe doğru sıralalıdır. Bağlı listede başa, sona ve araya ekleme metotlarının tümü kullanılılmıştır.

## II. PROGRAM MİMARİSİ

Bu kısımda projede kullanılan araçlar ve yöntemler hakkında detaylı bilgi verilecektir.

### A. Bağlı Liste ve Düğüm

Bağlı liste, her elemanın bir değerinin yanında bir de referans içerdiği veri yapısıdır. Bağlı liste bir dinamik veri yapısıdır ve programın çalışması sırasında büyüyüp küçülebilir. Program struct linkList adlı bağlı listeyi ve *sonrasi adlı bir düğüm içeriyor. linkList bağlı listesi, txt dosyasından okutulup sayılarak sırayla eklenilen kelimeleri ve kelimelerin adet sayılarını içerisinde tutmaktadır. Sonrasi düğümüyle kendisine bağlanan komşu düğümü işaret ediyor. Struct linkList yapısı şu şekildedir;

```
struct linkList{

    int adet;

   char kelime[100];

   struct linkList *sonrasi;

};
```

Yapı içerisinde, düğümdeki kelimelerin tekrar sayısının saklanması için tamsayı türünde bir değişken (int adet) ,düğümde kelimelerin tutulduğu bir string (char kelime[100] ) ve bir sonraki düğümleri işaret etmesi için struct linkList tipinde bir işaretçi (struct linkLst *sonrasi; ) tanımlanmıştır. Oluşturulan *ilk ve *son düğümleri tek yönlü listenin ilk ve son elemanını işaret ederek, bellek işlemlerinde bu elemanlara kolayca ulaşmamızı sağlamaktadır. Proje, kullanılan bu bağlı liste yapısı ile gerçekleştirilmektedir. Bu yapı; çeşitli fonksiyonlara gönderilerek içerisine veri atamaları

yapılmakta ve bazı çeşitli fonksiyonlarda da bu yapılar cinsinden değerler döndürülmektedir. Bağlı listeler, genel olarak dizi veri yapısı üzerinde veya her bir düğüm için bellekten malloc() fonksiyonu ile dinamik şekilde bellek alanı ayırarak ve bunları işaretçi değişken alanları üzerinden birbirine bağlayarak tutulabilir. Tanımlanan struct linkList isimli bağlı listeye bakıldığında, bu yapıda istenilen türde istenildiği kadar eleman tanımlanabilir.

### B. Dosya İşlemleri

Projede, kelimeler.txt dosyasından veri çekilerek işlemler başlatılır. Öncelikle "kelimeler.txt" dosyası açılıyor ve fscanf() ile satır satır okuma işlemi gerçekleşiyor. Ardından satıra ait kelime bilgisi tanımladığımız char okunan[100] stringine atanıyor. Burada geçici olarak tutuluyor. Okunan stringi sayma() fonksiyonuna gönderilerek adet sayısı belirleniyor. Ardından listede aynı kelimeden bulunup bulunmadığı kontrol edilerek bağlı listeye ekleme yapılıyor. Tüm bu işlemlerin sonucunda metindeki adet sayısına göre büyükten küçüğe sıralanmış kelimeler elde ediliyor.

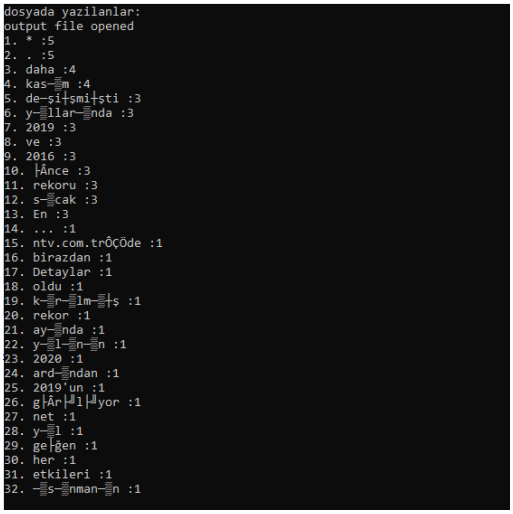
### C. Kullanılan Fonksiyonlar ve İşlevleri

- **void yazdir(struct linkList *b)** , bu fonksiyon bağlı listede bulunan yapıları ekrana yazdırır.
- **struct linkList *sona_ekle(struct linkList *r,int x,char *kelime)** , bu fonksiyon bağlı listenin sonuna eleman ekleme işlemini gerçekleştirerek bulunduğu düğümü döndürür.
- **struct linkList *basa_ekle(struct linkList *r,int x,char *kelime)** , fonksiyon bağğlı listenin başına ekleme yaparak listeyi döndürür.
- **struct linkList *araya_ekle(struct linklist *r,int x,char *kelime)** , fonksiyon bağlı listede yönlendirilen yere ekleme yapmaktadır. x değerine bağlı olarak içerisinde sona_ekle() fonksiyonu da barındırır ve bulunan bağlı listeyi döndürmektedir.
- **int sayma(char *okuma)** , bu fonksiyonda dosya açma kapama işlemleri gerçekleştirilir. Dosya açılarak kelimelr okutulur ve okuma stringiyle aynı olan kelimeler strcmp() fonksiyonu ile teyit edilerek sayısı belirlenir. Belirlenen sayı döndürülür.
- **struct linkList *siralama(struct linkList *liste,int x,char *kelime)** , fonksiyonu gelen bağlı listeyi x değerine göre belirlenen durumlar ile sona_ekle() , basa_ekle() ve araya_ekle fonksiyonlarına gönderir. Sonucunda ise karsilastirma() fonksiyonuna linkList yapısı döndürür.
- **struct linkList *karsilastirma(struct linkList *liste,int x,char *kelime)** , bu fonksiyon dosyadan okunan kelimenin daha önce bağlı listeye eklenip eklenmediğini strcmp() fonksiyonunu kullanarak kontrol eder. Bağlı listede bu kelimeden varsa ekleme yapılmaz eğer yoksa sıralama fonksiyonuna gönderilerek ekleme yapılır. Sonucunda da bağlı liste döndürülür.

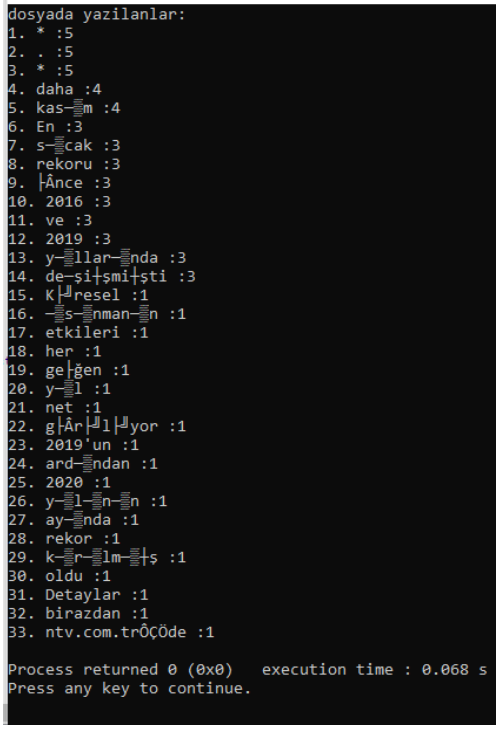
## III. PROJENİN GENEL YAPISI

Projede temel olarak txt dosyasından okutulan kelimeler adetleri sayılarak büyükten küçüğe bağlı listeye ekleme yapılmaktadır. Bu işlemler sırasıyla; dosyadan okuma, sayma, karşılaştırma, sıralama ve sona, başa, araya ekleme yapılarak gerçekleştirilmektedir. Sonucunda kelimeler ve adet sayıları sıralı olarak bağlı listeden konsola ve txt dosyasına yazdırılmaktadır.

## IV. YAPILAN ARAŞTIRMALAR




**Şekil 1 İlk Hali (Console Çıktısı)**




**Şekil 2 Fonksiyonlardaki değişiklikten sonra**

# V. YAKLAŞIMLAR

## A. BAĞLI LİSTELER

Bağlı listeler (linked list) bilgisayar programlarında kullanılan bir tür veri yapısıdır. Bağlı listeler bir zincirin halkaları gibi birbirine bağlı düğümlerden oluşur. Her düğüm iki kısımdan oluşur ilk kısım istenilen verinin kaydedileceği değişkenlerdir ikinci kısım ise bir sonraki ve tipine bağlı olarak bir önceki düğümün adresini içeren işaretçidir (pointer). Bağlı listedeki verilere sıra ile erişilebilir, ilk düğümden başlanarak önce veri okunur sonra işaretçinin gösterdiği adresteki bir sonraki düğüme geçilir, veri okunur ve sonraki düğüme geçlir bu işlem bütün düğümler tamamlana kadar devam eder.

Bağlı listelerin kullanım amacı sıralı verilerilerin bellekte erişilebilir bir düzende tutulmasıdır. Dizilerden (array) farklı olarak verilere erişim sıralıdır. Dizilerde iki elaman arasına bir başka eleman eklemek için dizinin yeniden düzenlenmesi gerekir bu işlem dizideki veri miktarına bağlı olarak belirli bir zaman alır, büyük dizilerde bu işlemi yapmak programın performansını önemli ölçüde düşürür.

## B. Tek Yönlü Bağlı Listeler

Her düğüm yanlızca bir sonraki düğümün adresi barındırır bunun sonucu olarak tek yönde okuma yapılabilir, bir düğümden önceki düğüme erişilemez. Düğüm yapısı C programlama dilinde struct kullanılarak yapılır

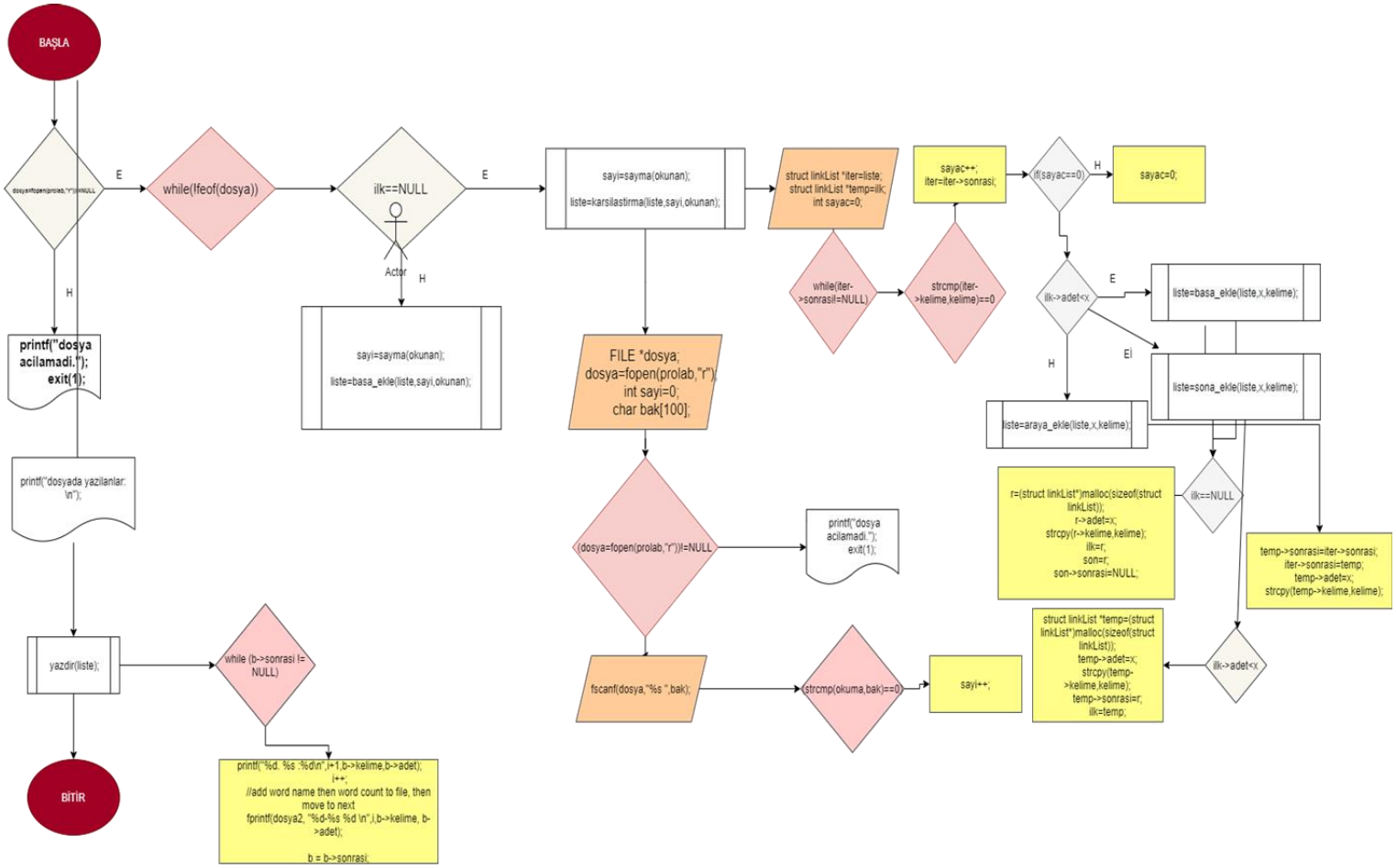
# VI. KAZANIMLAR

C dilinin en önemli konularımdan biri olan işaretçilerin(pointer) kullanımı ve bellek yönetimi gibi konularda oldukça fazla tecrübe elde ettik. Yapısal programlama mantığıyla modelimizi tekrar oluşturduk. Bu da her iki programlama geleneğinin farklarını çok net görmemizi sağladı. Programın çeşitli kısımlarında farklı bellek işlemlerini gerçekleştirdik böylece dinamik bellek yönetimi, işaretçi aritmetiğini vs. oldukça sık kullandık ve bu yöntemleri daha iyi tecrübe etme fırsatını yakaladık.

# VII. REFERANSLAR


1) David C. Rankin https://stackoverflow.com/questions/36323700/counting-words-from-text-file-into-linked-list-in-c(18.12.2020)
2) Sadi Evren Şeker https://www.youtube.com/watch?v=P976s2Qbpc4 (21.12.2020)
3) Sadi Evren Şeker https://www.youtube.com/watch?v=wDAf9Er6Qq8&t=203s (18.12.2020)
4) Sadi Evren Şeker https://www.youtube.com/watch?v=r3uOBb3BM-0&list=PLh9ECzBB8tJN9bckI6FbWB03HkmogKrFT&index=1&t=31s( 21.12.2020)
5) Sadi Evren Şeker https://www.youtube.com/watch?v=DGi_gSfYfKo&list=PLh9ECzBB8tJN9bckI6FbWB03HkmogKrFT&index=2( 02.01.2021)
6) Web site https://notpast.com/c_programlama/Bagli-Listeler-72.html (04.01.2021) 7)
7) http://embedded.kocaeli.edu.tr/?page_id=91(16.12.2020)


# VIII. AKIŞ DİYAGRAMI


BAŞLA
dosya=fopen(prolab,"r")!=NULL
E
H
while(!feof(dosya))
ilk==NULL
E
Actor
H
sayi=sayma(okunan);
liste=basa_ekle(liste,sayi,okunan);
sayi=sayma(okunan);
liste=karsilastirma(liste,sayi,okunan);
struct linkList *iter=liste;
struct linkList *temp=ilk;
int sayac=0;
sayac++;
iter=iter->sonrasi;
if(sayac==0)
H
sayac=0;
while(iter->sonrasi!=NULL)
strcmp(iter->kelime,kelime)==0
ilk->adet<x
E
liste=basa_ekle(liste,x,kelime);
H
El
liste=araya_ekle(liste,x,kelime);
liste=sona_ekle(liste,x,kelime);
printf("dosya acilamadi.");
exit(1);
FILE *dosya;
dosya=fopen(prolab,"r");
int sayi=0;
char bak[100];
printf("dosyada yazilanlar: \n");
(dosya=fopen(prolab,"r"))!=NULL
printf("dosya acilamadi.");
exit(1);
r=(struct linkList*)malloc(sizeof(struct linkList));
r->adet=x;
strcpy(r->kelime,kelime);
ilk=r;
son=r;
son->sonrasi=NULL;
ilk==NULL
temp->sonrasi=iter->sonrasi;
iter->sonrasi=temp;
temp->adet=x;
strcpy(temp->kelime,kelime);
yazdir(liste);
while (b->sonrasi != NULL)
fscanf(dosya,"%s ",bak);
(strcmp(okuma,bak)==0)
sayi++;
struct linkList *temp=(struct linkList*)malloc(sizeof(struct linkList));
temp->adet=x;
strcpy(temp->kelime,kelime);
temp->sonrasi=r;
ilk=temp;
ilk->adet<x
BİTİR
printf("%d. %s :%d\n",i+1,b->kelime,b->adet);
i++;
//add word name then word count to file, then move to next
fprintf(dosya2, "%d-%s %d \n",i,b->kelime, b->adet);
b = b->sonrasi;