KOCAELİ ÜNİVERSİTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

PROGRAMLAMA LAB. I- I. Proje

Sude Yağmur Göçgün / 190202046 / sude_572000@hotmail.com

Canberk Tezcan / 190202033 / can-berk2000@hotmail.com

BAĞLI LİSTE İLE KELİME SAYMA

-PROBLEM TANIMI-

Projemizin amacı kullanıcı tarafından .txt dosyasına girilen bir metin içerisinde geçen kelimelerin her birinin adetini bulup, bu kelimeleri adetleri ile birlikte bir bağlı listeye, büyükten küçüğe olacak şekilde eklememiz ve devamında bu bağlı listeyi ekrana yazdırmamızdır.

Önemli olan nokta dosyadan kelimeleri birer birer okumak ve bir kelimeyi okuduktan sonra tüm dosyayı tarayıp adet sayısını bulup, adetine göre bağlı listeye BAŞAEKLE-SONAEKLE-ARAYAEKLE methodlarını kullanarak eklemektir. Tüm kelimeler ve adetleri bir yerde tutulup en son bağlı listeye **dizilmeyecektir.**

Bağlı liste kullanımı yerine dizi kullanımı kesinlikle olmayacaktır.

-YAPILAN ARAŞTIRMALAR-

Projede öncelikle ilk karşılaştığımız sorun dosyadan okuma işlemiydi. Bunun için C'deki hazır fonksiyonları araştırmaya başladık.

Dosyadan okuma işlemi için önce fgets() fonksiyonunu kullandık. Fgets() fonksiyonu ile tüm texti okuyup bir pointera attıktan sonra bu pointerı strtok() fonksiyonu ile kelimelerine ayırıyorduk. Fakat burada fgets() fonksiyonunun sadece tek bir satır okuyabilmesinden kaynaklı sorun yaşadık. Bu sebepten dosya okuma için fscanf() fonksiyonuna geçiş yaptık.

Karşılaştığımız diğer sorun, kodumuz dosyada kelime adetini tararken birbirini içinde barındıran kelimeleri örneğin: merhaba ve merhabalar eşit olarak görüyordu. Bunun sebebi kelimelerin eşitliğini harf harf kontrol etmemizdi. Merhaba kelimesini merhabalar kelimesine eşit mi diye kontrol ederken, son harfi a'da eşit çıktıktan sonra merhaba kelimesinin harfleri bittiği için kontrolü bitiriyor ve sonucu merhabalara eşit buluyordu.

Bu sorunun çözümü için kelimeleri harf harf kontrol etmek yerine , bütününü kontrol eden fonksiyonu araştırmaya başladık ve strcmp() fonksiyonuna ulaştık.

Önce dosyadan okuduğumuz ilk kelimeyi ve tüm metni tutup adet bulma fonksiyonuna gönderdik. Ama burada ilk kelimenin adetini bulduktan sonra diğer kelimeler için kod çalışmadı. Biz de bu

durumun çözümü olarak adet bulma fonksiyonuna tüm metni göndermek yerine sadece aranacak kelimeyi gönderdik. Daha sonra fonksiyonun içinde dosyayı açıp okuyup adeti saydırdıktan sonra dosyayı kapattık. Böylece fonksiyon her çağırıldığında dosya tekrar açıldı, okundu ve kapandı. Sorunumuz da çözüldü.

-TASARIM-

Yazılım Mimarisi :

Kodumuzu şu mimaride tasarladık:

- Struct tipinde linked list oluştur.
- Mainde okunacak dosyayı aç.
- Dosyadan birinci kelimeyi oku.
- Kelimeyi adet bulma fonksiyonuna gönder.
- Adet bulma fonksiyonu içinde gelen kelimeyi tüm dosyada tara ve buldukça sayısını arttır.
- Maine dönen adet sayısını okunan kelime ile beraber başa sona veya araya ekle methodlarına gönderip bağlı listeye ekle.
- Bağlı listeyi ekrana yazdır.

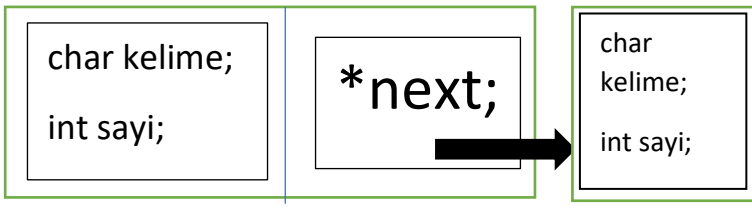
-GENEL YAPI-

Projemizin başlangıcında öncelikle bağlı listemiz için Structımızı oluşturduk.

İçinde kelimeyi ve kelimenin adetini tutuyor.

```
typedef struct n
{

    char kelime[100];
    int sayi;
    struct n* next;
} Node;
```



Bağlı listeye ekleme yapmamız için 3 adet fonksiyon isterimiz vardı. BaşaEkle SonaEkle ve ArayaEkle. Mainden bu fonksiyonlardan herhangi birini çağırabilmemiz için bağlılisteye eklenecek elemanı her seferinde bilmemiz ve ona göre gerekli fonksiyonu çağırmamız gerekiyordu. Bu mümkün olmayacağı için öncelikle ana bir Ekle fonksiyonu yazıp içinde kontroller gerçekleştirerek BaşaEkle,SonaEkle veya ArayaEkle fonksiyonlarından birini çağırdık.

Node * Ekle(Node *d,char kelimegelen[100],int x) Fonksiyonu :

Burada 3 kontrol sağlamamız gerekiyordu.

İf(bağlı liste boşsa)

Else if(bağlı listede tek eleman varsa)

Else(araya eklenecekse)

Ekle fonksiyonuna bir bağlı liste elemanı geldiğinde önce liste boş mu kontrolü yaptık. Eğer boşsa BasaEkle fonksiyonunu çağırdık.

```
if(d==NULL)
{
amamen bossa
    d=BasaEkle(d,kelimegelen,x);
    return d;
}
```

Diğer durumların hepsini bir else içinde kontrol ettik. Bu else içinde ilk yaptığımız işlem, gelen kelimenin bağlı listede olup olmama durumuydu. Varsa tekrar eklenmemesi gerekiyordu. Bunu tüm listeyinin kelimelerini strcmp() fonksiyonuyla gezerek kontrol ettik. Eğer varsa hiçbir değişiklik yapmadan gelen listeyi geri gönderdik.

```
while(gezen!=NULL)
{
    if(strcmp(gezen->kelime,kelimegelen
    {
        //ayni kelime kontrolu
        return d;
    }
```

Aynı kelime var mı? kontrolünden sonra, kodumuz ikinci durumumuz olan -listede tek eleman varsa- ihtimalini kontrol etti. Ve tek eleman olma durumunda tekrar ikiye ayrıldı

- If(gelen adet > liste elemanının adedi)
  Çağır -> BasaEkle();
- Else(gelen adet<liste elemanının adedi)

  Çağır->SonaEkle();

İkinci durumumuz kontrolünden sonra özel durumlar bitmiş oldu. Tek ihtimal gelen düğümün araya eklenmesiydi. Bu sebepten ötürü son olarak

Else {

Çağır->ArayaEkle() }

İşlemini gerçekleştirdik.

Node * BasaEkle(Node *d,char kelimegelen[100],int x) Fonksiyonu :

İki farklı durumda basaekle() çağırıldığı için

-tamamen boş olma-

-tek eleman olma-

Fonksiyonumuz içinde bu iki durumu tekrar kontrol ettik.

<u>Eğer liste boşsa:</u>

Gelen eleman için önce yer aldık.

d=(Node*)malloc(sizeof(Node));

d->next=NULL;

Gelen kelimeyi strcpy() yardımıyla

d->kelimesine kopyaladık.

d->sayisini da gelen sayiya eşitledik. d'yi fonksiyondan geri döndürdük.

<u>Eğer listede tek eleman varsa:</u>

Yeni eklenecek eleman için yer aldık.

Node * gecici=(Node*)malloc(sizeof(Node));

Gecicinin kelimesini ve sayisini gelen kelime ve sayiya eşitledikten sonra,

Geçicinin->next= d;

yaparak geçicinin 1. Eleman olmasını sağladık. Ve

yeni düğüm başlangıcımız artık geçici olduğundan

return geçici;

Node * ArayaEkle(Node *d,char kelimegelen[100],int x) Fonksiyonu :

Önce baş düğümümüzü kaybetmeden liste içinde gezmek için yeni bir değişken tuttuk.

Node * gezen=d;

While ile gezen'i nexti null olana kadar veya nextinin sayisi x'ten küçük olana kadar ilerlettim.

```
while(gezen->next!=NULL && gezen->next->sa
{
    gezen=gezen->next;
}
```

Fonksiyona gelen x'in listenin başındaki elemanın sayısından büyük olması durumunda başaekleme işleminin ayrıca yapılması gerekiyordu. Aynı kodları tekrar etmemek için burada BasaEkle() fonksiyonunu çağırdık.

```
if(d->sayi<x)
{
    d=BasaEkle(d,kelimegelen,x);
    return d;

}
```

Kalan durumlarda gezen'i araya ekleme işlemi yapılacak yere kadar ilerlettiğimiz için ve bağlı listenin başını (*d) kaybetmemek için araya ekleme işleminin kodunu gezen üzerinden yazdık.

Eklenecek eleman için önce yer aldık.

Node * gecici=(Node*)malloc(sizeof(Node));

Gezen tam ekleme yapılacak yerde durduğu için

gecici->next=gezen->next;

gezen->next=gecici;

strcpy(gecici->kelime,kelimegelen);

gecici->sayi=x;

return d;

Node * SonaEkle(Node *d,char kelimegelen[100],int x) Fonksiyonu :

Burada da işlemler çok benzerdi. Listemizin başı değişmeyeceği için içinde gezinmek için yeni bir değişken tuttuk.

Node * gezen=d;

Ve listenin sonuna gelene kadar yani

while(gezen->next!=NULL)

olana kadar gezeni birer ilerlettik. Gezen sona geldiğinde ekleme yapılacak eleman için yer aldık ve

Node * yeni=(Node*)malloc(sizeof(Node));

gezen->next=yeni;

yeni->next=NULL;

strcpy(yeni->kelime,kelimegelen);

yeni->sayi=x;

işlemlerini yaparak sona yeni elemanı eklemiş olduk.

return d;

Void EkranaBas(Node*d) Fonksiyonu :

Gelen bağlı listenin elemanlarını son eleman olana kadar(gezen!=null) döngü içinde ekrana yazdırdık.

```
void EkranaBas(Node *d)
{
    int i=0;
    Node * gezen;
    gezen=d;
    while(gezen!=NULL)
    {
        i++;
        printf("%d. kelime: %s  >> adet: %d \n",i,gezen->kelime,gezen->sayi);
        gezen=gezen->next;

    }
}
```

Bağlı liste işlemleri ve fonksiyonları bu kadardı. Mainde dosyadan okuduğumuz birinci kelimenin adetini hesaplamamız için ayrı bir fonksiyon daha açmak istedik ve okuduğumuz kelimeyi bu fonksiyona gönderdik.

int kelime_sayici(char kelime[]) Fonksiyonu :

Gelen kelimeyi dosyayı okutarak içinde tarayacağımız için fopen() ile textimizi 'r' formatında açtık. Değişkenlerimizi tanımladık.

```
FILE*f=fopen("deneme.txt", "r");
int i=0;
char c;
char kelimeler[1000000000];
```

while() döngüsü içinde fgetc() fonksiyonu ile dosya sonuna gelene kadar dosyanın içindeki her bir char karakterini kelimeler[] dizisine attık. Böylece texti elimizde tutuyoruz.

Hemen sonra for döngüsü içinde kelimeler[] dizisini strtok() fonksiyonu ile boşluk görene kadar okutuyoruz ve bu işlem döngüde kelimeler dizisinin son karakterine gelene kadar devam ediyor.

Döngü her bir kere döndüğünde, strtok ile diziden okunan kelime, fonksiyona gelen(eşitliği kontrol edilecek olan kelime) bir if içerisinde strcmp() fonksiyonu ile kontrol edilerek her eşitlik sağlandığında kelime adeti değişkenini bir arttırdık. For döngüsü sonlandığında fonksiyona gelen kelimenin adeti hesaplanmış oldu ve geriye adet bilgisini döndürdük. Kodumuz:

```
char *okunankelime=NULL;
int sayac=0;


for(okunankelime = strtok(kelimeler," "); okunankelime != NULL; okunankelime = st
{
    if(strcmp(kelime,okunankelime)==0)
    {
        sayac=sayac+1;
    }
}

fclose(f);

return sayac;
```

int main() :

Dosyamızı fopen() ile r modunda açtıktan sonra bir while döngüsü içinde dosya sonuna gelene kadar fscanf() ile her okunan kelimeyi bir değişkene attıktan sonra önce kelime_sayici fonksiyonuna göndererek adetini hesaplattık. Dönen adet bilgisini ve okunmuş olan kelimeyi Ekle methoduna göndererek bağlı listeye ekledik. Döngü sona erdiğinde ise bağlı listemizi EkranaBas fonksiyonuna göndererek ekrana yazdırdık. Ve dosyayı kapattık.

```
int main()

{

   Node* root;

   root=NULL;

   FILE*dosya=fopen("deneme.txt", "r");

   char word[100];

int sayi;

   while(fscanf(dosya,"%s",word)!=EOF) //tek tek kelimelei tutup bağlılisteye ekleme

   {

      sayi=kelime_sayici(word);

      root=Ekle(root,word,sayi);

   }

   fclose(dosya);

   EkranaBas(root);

   return 0;

}
```

ÖRNEK TEXT VE ÖRNEK ÇIKTI:

Text:

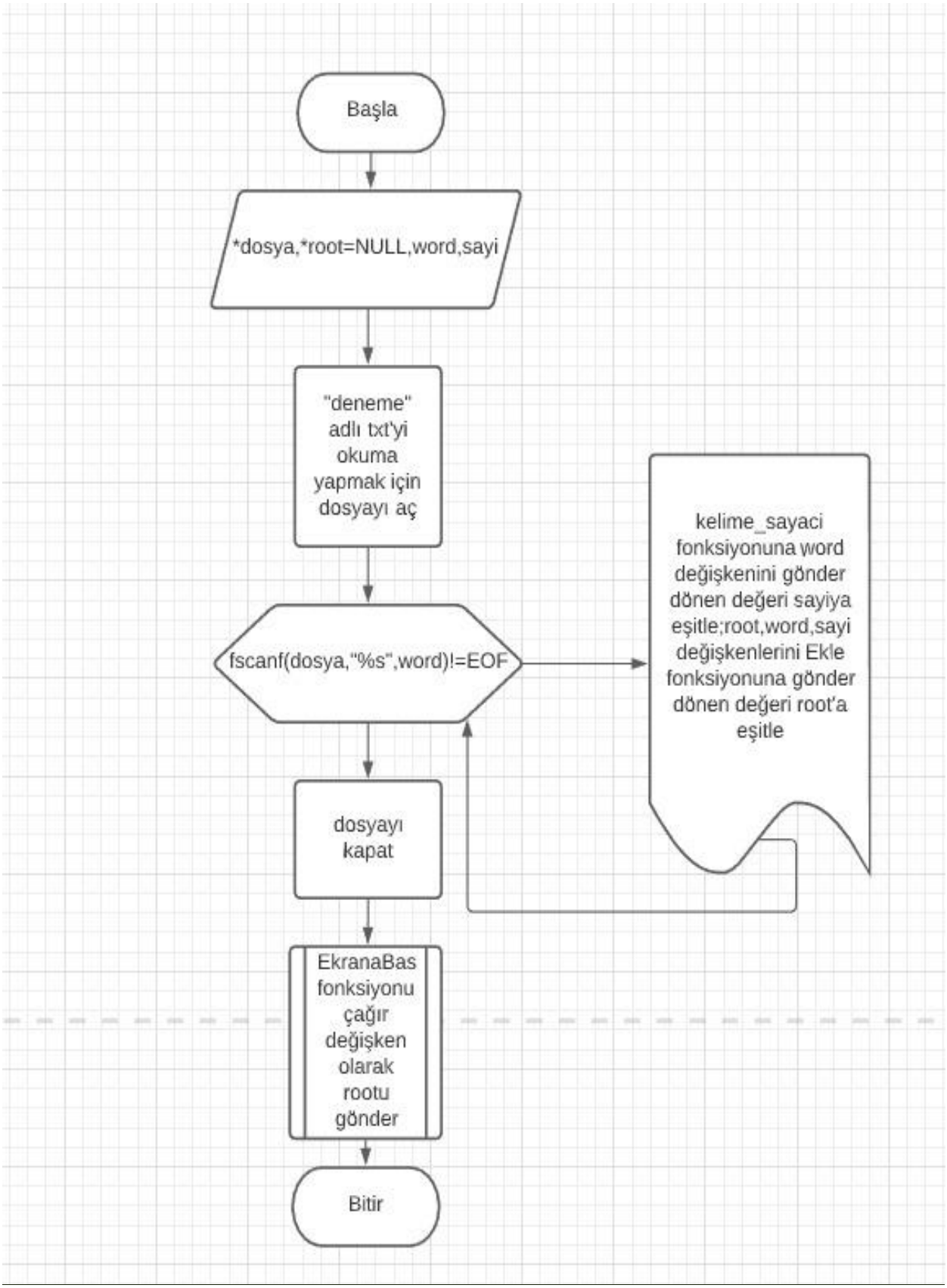
makas , makas , makas ,
ali dayı , noktaya bas ,
trampetler çalınıyor ,
yüzbaşılar darılıyor ,
darılmayın yüzbaşılar ,
can yürekten ayrılmıyor .

Çıktı:

```
"C:\Users\sude ya-mur\Desktop\prolab1proje3\main.exe"
1inci:   ,  >>>  8  adet var
2inci:   makas  >>>  3  adet var
3inci:   yuzbasilar  >>>  2  adet var
4inci:   .  >>>  1  adet var
5inci:   ayrilmiyor  >>>  1  adet var
6inci:   yurekten  >>>  1  adet var
7inci:   can  >>>  1  adet var
8inci:   darilmayin  >>>  1  adet var
9inci:   dariliyor  >>>  1  adet var
10inci:   caliniyor  >>>  1  adet var
11inci:   trampetler  >>>  1  adet var
12inci:   bas  >>>  1  adet var
13inci:   noktaya  >>>  1  adet var
14inci:   dayi  >>>  1  adet var
15inci:   ali  >>>  1  adet var

Process returned 0 (0x0)   execution time : 0.056 s
Press any key to continue.
```

-Akış Diyagramı-



-KAZANIMLAR-

Projeyi gerçekleştirmeye çalışırken dosyadan okuma işlemlerini uzunca araştırdık ve bu konuda birçok bilgi edindik.

C dili hazır fonksiyonları hakkında da birçok araştırma yaparak büyük ölçüde konuya hakim olduk.

Bağlı Liste konusunu ve işleyiş mantığını araştırarak ve proje üzerinde uygulama fırsatı bularak çok iyi bir biçimde kavramış olduk.

-KAYNAKÇA-

1. https://www.bilgigunlugum.net/prog/cprog/c_stdkut/stdio/fscanf

2. https://www.bilgigunlugum.net/prog/cprog/c_stdkut/stdio/fgetc

3. http://bilgisayarkavramlari.com/2008/10/22/c-ile-dosya-islemleri/

4. https://www.geeksforgeeks.org/search-an-element-in-a-linked-list-iterative-and-recursive/

5. https://www.tutorialspoint.com/data_structures_algorithms/linked_list_algorithms.htm

6. https://www.youtube.com/watch?v=r3uOBb3BM-0&t=2s