BAĞLI LİSTE İLE KELİME SAYMA PROJESİ

Aslı Özen, Beyza Hatip
Bilgisayar Mühendisliği Bölümü
Kocaeli Üniversitesi
asliiozen@hotmail.com, beyzahatip17@gmail.com

Özet

Bu projede dosyadan okuma yapılarak kelimeler bir char dizisinde tutuldu. Her kelimenin metinde kaç kere geçtiği bir fonksiyonda hesaplanarak bu değer ve kelimenin kendisi bağlı listeye ekleme yapmak için oluşturulan fonksiyona gönderildi. Adet sayısı fazla olan kelimeden az olan kelimeye doğru sıralanacak şekilde uygun ekleme fonksiyonları kullanılarak kelime ve adeti eklendi.

Giriş

Bu projede amaç; text içerisindeki kelimeleri sayarak bağlı listeye adet olarak büyükten küçüğe doğru eleman ekleme yaparak bizlerin bağlı liste yapısını anlaması ve çözüm sağlayabilmesidir. Öncelikle bağlı liste için bir struct yapısı oluşturduk; kelimeler için char tipinde dizi; adetler için int tipinde bir değişken tanımladık. Bir sonraki düğüme geçebilmek için ve ilk düğüm, yeni düğüm ve geçici değer tutabilmek için pointerlar tanımladık. İlk düğümü NULL yaparak listeyi başta boş olarak tanımladık. Dosya okuma için main()de dosya adını belirterek dosyayı açtık ve dosyanın sonuna kadar okuma yapan while döngüsü içerisinde char tipinde tanımlanan bir dizi ile fscanf() fonksiyonu ile aldığımız kelimelerimizi tuttuk. Her bir kelime bulunduğunda onu ve dosyayı metin içerisinde kaç tane o kelimeden olduğunu bulan fonksiyona gönderdik. Bu fonksiyonda da dosyayı açtıktan ve adet için int tipinde bir değişken tanımladıktan sonra yine dosyanın sonuna kadar okuma yapabilmesi için bir while döngüsü açtık. Bu döngü strcmp() fonksiyonu ile gelen kelimenin aynısına metinde rastlanırsa adeti 1 arttırarak o kelimenin kaç tane olduğunu bulmamızı sağladı. Bulunan adeti main'e return ettik. Bu şekilde önce metindeki kelimeyi tutup sonra onun adetini bulduktan sonra main()'de aynı while döngüsünün içinde bu adet ve kelimeler bağlı listeye eklenmek üzere add() fonksiyonuna gönderiliyor. Böylece her bir kelime teker teker adet sayısı bulunduktan sonra teker teker listeye eklenmiş oldu. add()fonksiyonu gelen kelimeyi başa sona veya araya ekleme fonksiyonlarına gönderir. İlk if daha önce struct yapısındaki tanımladığımız ve NULL'a eşitlediğimiz first değişkenini kontrol ediyor. Eğer first=NULL ise o kelimeyi ve adetini başa ekleme fonksiyonuna gönderiyor. Diğer durumları bir if altında gerçekleştiriyoruz. Bu if kelimenin daha önce listeye eklenip eklenmediğini kontrol eden bir fonksiyon 0 değerini return ettiğinde çalışıyor. Bu fonksiyon strcmp() ile kelimenin daha önce listeye eklenip eklenmediğini kontrol ediyor. Bunun için temp değişkenini önce sırayla first'e eşitliyoruz listenin sonuna kadar gitmek için while döngüsü içine first==NULL şartını yazıyoruz. strcmp() ile daha önce bu kelime düğümde varsa fonksiyonun başında tanımladığımız ve 0'a eşitlediğimiz ctrl değişkeni 1 artıyor. Eğer bulamazsa ctrl değişkeni 0 kalıyor. Bütün düğümler temp=temp->next ile kontrol ediliyor. ctrl değişkeni add() fonksiyonuna return ediliyor. Böylece sadece ctrl 0 iken çalışan if koşulu aynı kelimeden düğümde yoksa ekleme yapma fonksiyonlarına kelime ve adetini gönderir. Bu koşulun altındaki ilk koşul yeni gelen değer ilk düğümdeki değerden büyükse bu kelime ve değerini başa ekleme fonksiyonuna gönderir. Eğer değilse araya ekleme ve sona ekleme fonksiyonları için temp değişkenini first'e eşitiyoruz ve bir while döngüsü açıyoruz bu döngü listenin sonuna kadar dönüyor. Döngünün içindeki ilk if temp düğümünden sonraki düğüm boş olduğunda yani listenin son düğümüne

geldiğimizde ve gelen değer son düğümün değerinden küçük veya eşit olduğunda bu değer ve kelimeyi sona ekleme fonksiyonuna gönderiyor. Diğer if koşulu ise; gelen değer temp'ten sonraki düğümdeki amount değişkeninden büyük olduğunda değeri ve kelimeyi araya ekleme fonksiyonuna gönderiyor. Düğümler arasında gezmek için temp=temp->next yaptırıyoruz. Bu ekleme fonksiyonları gerekli işlemleri yaptıktan sonra main()de while döngüsü içinde bunları yazdırıyoruz. Ekleme fonksiyonları raporun **yöntem** kısmında detaylı olaraka açıklanacaktır. main()'de bir list pointerı oluşturuyor ve onu first'e eşitliyoruz. while döngüsü içinde list null olana kadar düğümlerin kelime ve adetlerini yazdırıyoruz ve list=list->next diyerek bütün düğümleri yazdırmış oluyoruz.

Yöntem

Bu projede dosyadan okuma yapabilmek için öncelikle fgets() fonksiyonu kullanıldı fakat bu şekilde boşlukları da aldığı için fscanf() fonksiyonuna geçildi. while döngüsü içinde !feof(f1) diyerek dosya sonuna kadar okuma yapıldı. Her seferinde char tipinde word1[] dizisi yeniden tanımlanarak her seferinde farklı kelimelerin alınması sağlandı. Metindeki kelimeden kaç tane olduğunu bulan calAmount() fonksiyonunda amount değişkeni 0' eşitlendi. Ve dosya sonuna kadar dönen while döngüsü açıldı. char tipinde read[] dizisi her seferinde yeniden tanımlanarak fscanf() ile her seferinde dosyadaki diğer kelime ile main()'de bu fonksiyona gönderilen kelimeyi karşılaştırmamız için oluşturuldu. strcmp() fonksiyonu ile main() den gelen kelime ile her seferinde read[]'deki kelime karşılaştırıldı. Kelime aynı ise 0 sonucunu veren strcmp() fonksiyonu 0 sonucunu verdiğinde amount değerini 1 arttırdık ve döngü bittiğinde amount değerini main()'e return ettik. calAmount() fonksiyonunda ilk başta dosyayı açmamıştık fakat böyle olunca sadece ilk kelime adedini bulduğunu fark edince burada da dosyayı açmak için main()' den dosyayı da gönderdik. main()'e return edilen amount'u ve kelimeyi add() fonksiyonuna gönderdik. Bu add() fonksiyonu belirli şartlara göre gelen kelime ve adedini 3 adet ekleme fonksiyonundan birine gönderiyor. İlk if bağlı liste ilk başta boşken başa ekleme fonksiyonuna(addToTheHead()) kelime ve adedini göndermemizi sağlıyor. struct yapısı kurarken tanımladığımız ve NULL'a eşitlediğimiz first değişkeni ilk başta boş. Bu fonksiyon da içinde bir (first == NULL) koşulu barındırıyor. Bunun sebebi başa eklenecek başka bir durumda farklı işlemler yapılacak olmasıdır. Bu koşulun altında malloc() kullanılarak bir first düğümü oluşturuluyor, gelen amount değeri atanıyor ve memcpy() fonksiyonu ile kelime de düğüme eklenmiş oluyor. Bir sonraki düğüm ise first->next = NULL denilerek NULL'a eşitleniyor.Böylece ilk düğümü oluşturmuş ve gerekli atamaları yapmış oluyoruz.add() fonksiyonundaki bir diğer koşul (searchLinkedList(word1)==0) ise aynı kelimeyi tekrar eklememek için kullandığımız bir şart. İlk kelime eklenirken bu şartı koymamıza gerek yoktu fakat diğer kelimeler için kontrol ediyoruz. searchLinkedList() fonksiyonunda daha önce struct yapısında tanımladığımız temp'i önce first'e eşitliyoruz ve int tipinde ctrl değişkenini 0 yapıyoruz. while (temp != NULL) döngüsü ile temp değişkeni NULL olana kadar dönüyoruz. Sonuç olarak tüm düğümler üzerinde temp = temp->next diyerek geziniyoruz. strcmp() fonksiyonu ile gelen kelime ile düğümlerdeki kelimeler karşılaştırılıyor. Eğer 0 ise bu aynı kelimeden olduğu anlamına geldiği için; if (!strcmp(word1,temp->word) bu koşul gerçekleştiğinde ctrl değişkenini 1; gerçekleşmediğinde 0 yapıyor ve ctrl'yi add() fonksiyonuna return ediyoruz. (searchLinkedList(word1)==0 olduğunda çalışan bu koşul böylece aynı kelime düğüme eklenmişse o kelimeyi eklemiyor.

Yazılım Mimarisi

```
typedef struct myNode{

    int amount;
    char word[50];
    struct myNode *next;

}myNode;
```

Yukarıda görülen struct yapısı ile düğüm iskeletimizi oluşturmuş oluyoruz. amount değişkeni kelime adetini tutar. word değişkeni kelimeyi tutar. next değişkeni listenin bir sonraki elemanına ulaşımı sağlar.

```
myNode *newNode, *first = NULL, *temp;
```

Yeni düğümü, ilk düğümü ve geçici düğümü ifade eden düğümleri tanımladık ve ilk düğümü null değerine eşitledik.

Adet hesaplama işlememini ilk olarak main() içindeki while döngüsünün içine yazmıştık fakat bu mantık çalışmadı.

```
int calAmount(FILE *f1,char word[]){
     int amount=0;
    f1=fopen("sentence.txt", "r");
    while(!feof(f1)){
              char read[100];
              fscanf(f1,"%s",read);
            if (strcmp(read, word) == 0) {
                amount++;
        }
    }
    fclose(f1);
   return amount;
}
```

(1)

Daha sonra bu işlemi fonksiyon (1) içinde yapıp fonksiyonu main() içindeki while içinde çağırarak projeyi doğru bir şekilde çalıştırabildik.

Projenin yapım aşamasında adet hesaplamasının doğru olup olmadığını kontrol edilmesi için aşağıdaki kod yazılmıştır.

Bu kodun çıktısı aşağıdaki gibi olmuştur. Kodun bu kısmının doğruluğu kontrol edildikten sonra bu kod silinmiştir.

```
FILE *f1;
f1=fopen("sentence.txt", "r");
if (f1 == NULL) return 1;
while(!feof(f1))
{   char word1[100];
    fscanf(f1,"%s",word1);
    printf("%d inci kelime %s\n",i,word1);
    i++;
    printf("This file has %d words in it. \n", calAmount(f1,word1));
    add(calAmount(f1,word1),word1);

}

fclose(f1);
```

```
1 inci kelime *
This file has 5 words in it.
2 inci kelime *
This file has 5 words in it.
3 inci kelime *
This file has 5 words in it.
4 inci kelime *
This file has 5 words in it.
5 inci kelime *
This file has 5 words in it.
6 inci kelime En
This file has 3 words in it.
7 inci kelime sicak
This file has 3 words in it.
8 inci kelime kasim
This file has 4 words in it.
9 inci kelime rekoru
This file has 3 words in it.
10 inci kelime daha
This file has 4 words in it.
11 inci kelime once
This file has 3 words in it.
12 inci kelime 2016
This file has 3 words in it.
13 inci kelime ve
This file has 3 words in it.
14 inci kelime 2019
This file has 3 words in it.
15 inci kelime yillarinda
This file has 3 words in it.
```

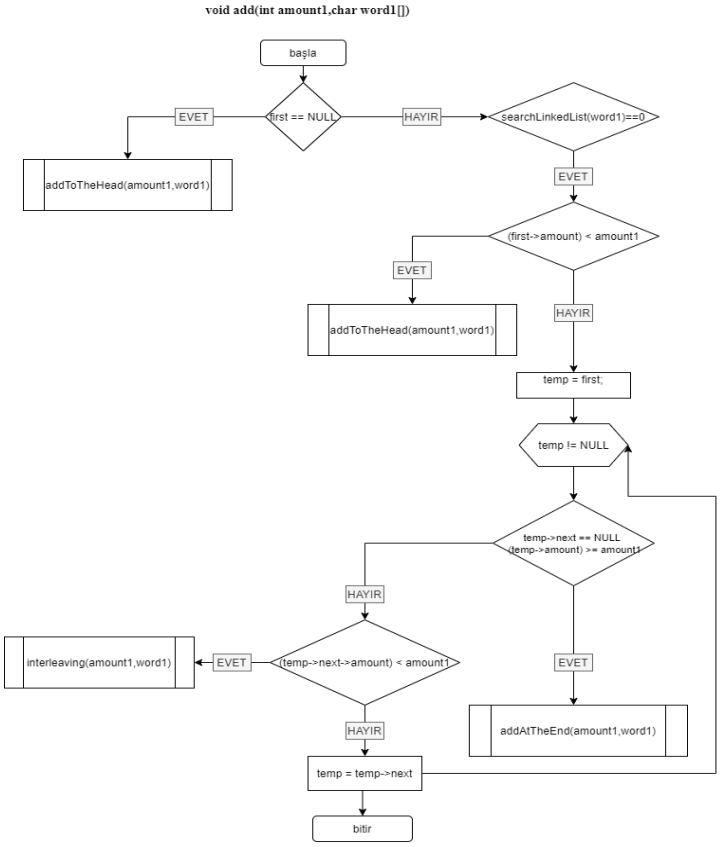
Sonuçlar

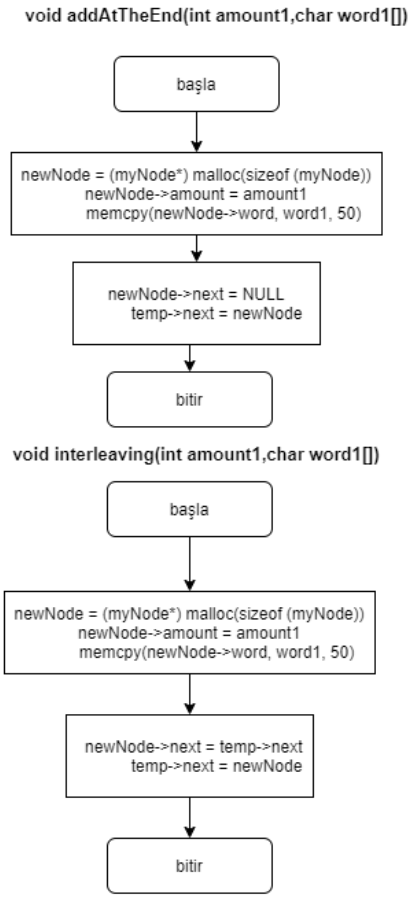
Kod birden fazla paragraf için denendi ve altta eklenen çıktılara ulaşılarak kodun doğruluğu kontrol edilmiştir.

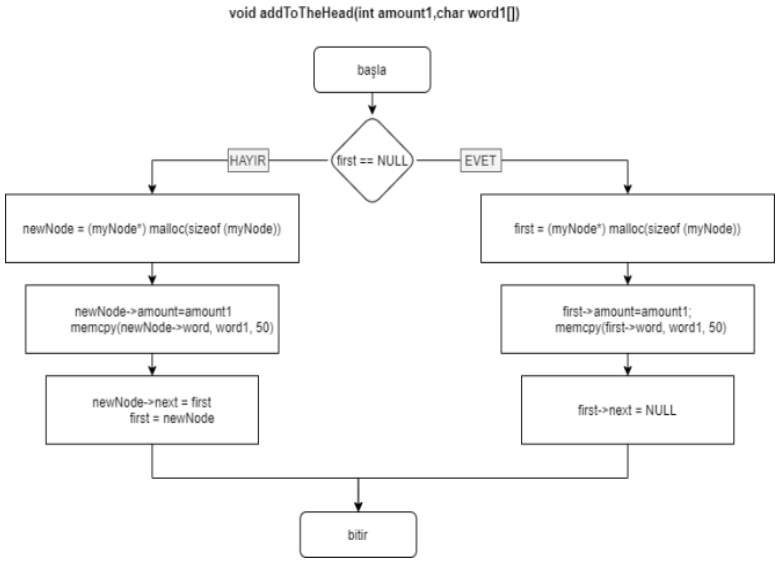
```
1. 5 : *
2. 5 : .
3. 4 : kasim
4. 4 : daha
5. 3 : En
6. 3 : sicak
7. 3 : rekoru
8. 3 : once
9. 3 : 2016
10. 3 : ve
11. 3 : 2019
12. 3 : yillarinda
13. 3 : degismisti
14. 1 : Kuresel
15. 1 : isinmanin
16. 1 : etkileri
17. 1 : her
18. 1 : gecen
19. 1 : yil
20. 1 : net
21. 1 : goruluyor
22. 1 : 2019'un
23. 1 : ardindan
24. 1 : 2020
25. 1 : yilinin
26. 1 : ayinda
27. 1 : rekor
28. 1 : kirilmis
29. 1 : oldu
30. 1 : Detaylar
31. 1 : birazdan
32. 1 : ntv.com.tr'de
33. 1 : ...

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

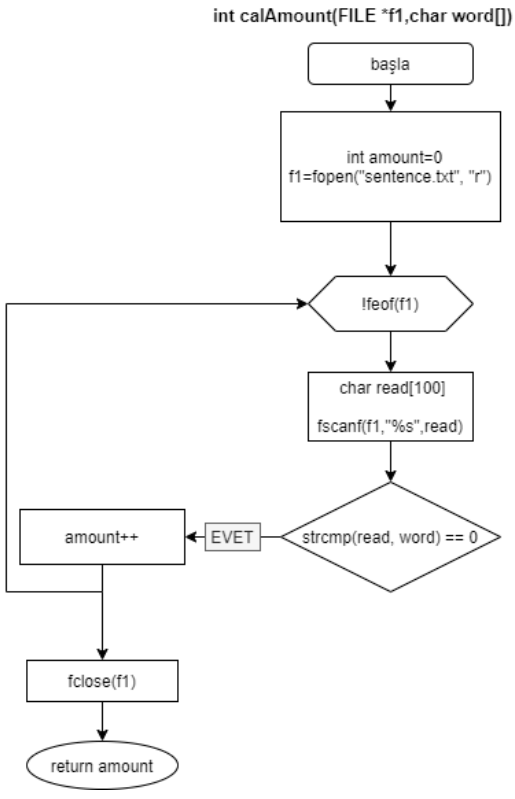
```
1. 10 : ,
2. 10 : the
3. 8 : of
4. 4 : *
5. 4 : and
6. 4 : her
7. 4 : as
8. 3 : +
9. 3 : .
10. 3 : in
11. 2 : also
12. 2 : is
13. 2 : underworld
14. 2 : abduction
15. 2 : with
16. 2 : vegetation
17. 2 : spring
18. 1 : In
19. 1 : Greek
20. 1 : mythology
21. 1 : Persephone
22. 1 : called
23. 1 : Kore
24. 1 : or
25. 1 : Kora
26. 1 : daughter
27. 1 : Zeus
28. 1 : Demeter
29. 1 : She
30. 1 : became
```

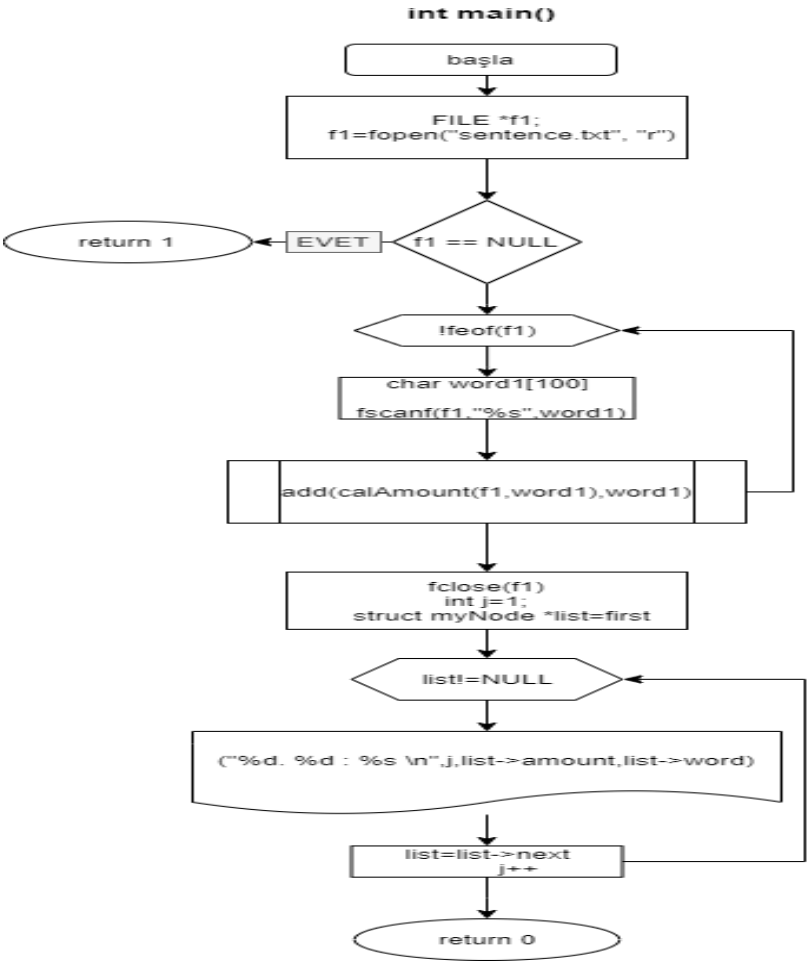
Akış Şemaları:











Şema çizimleri draw.io üzerinden yapılmıştır.

Referanslar

Yorulmaz M., Yorulmaz S. (2018). Programlamayı C ile Öğreniyorum. Ankara: Palme Yayınevi.

Website

https://www.bilgigunlugum.net/prog/cprog/c_stdkut/string/strcmp

https://stackoverflow.com/questions/2440420/compare-equality-of-char-in-c#:~:text=Check%20them%20in%20a%20for,change%20they're%20not%20equal.&text=You%20are%20checking%20the%20identity,character%20in%20the%20other%20array

https://www.geeksforgeeks.org/basics-file-handling-c/

https://fresh2refresh.com/c-programming/c-file-handling/

https://medium.com/@sddkal/c-veri-yap%C4%B1lar%C4%B1-generic-ba%C4%9Fl%C4%B1-liste-d5251d3b742f

http://bilgisayarkavramlari.com/2007/05/03/linked-list-linkli-liste-veya-bagli-liste/

http://bilgisayarkavramlari.com/2010/12/06/dosya-ve-bagli-liste-uygulamasi/