Muhammed Halil Akkaynak
190202075
Bilgisayar Mühendisliği Bölümü
Kocaeli Üniversitesi
halilakkaynak71@gmail.com

BAĞLI LİSTE İLE KELİME SAYMA

Erce Uslu
190202083
Bilgisayar Mühendisliği Bölümü
Kocaeli Üniversitesi
erceonline@gmail.com

Özet - Bu proje, bağlı liste kullanılarak metin belgesinden alınan kelimelerin kullanım sayısını hesaplamak için yapılmıştır.

Anahtar Kelimeler - Bağlı Liste, .TXT, Düğüm, Iter.

I. GİRİŞ

Bu proje temelde projeyi yapan kişilerin bağlı liste kullanımını ve dinamik belleği anlaması amaçlanmıştır.

Bu projeye başladığımız anda bağlı listeler üzerine araştırmalar yaptık. Düğümler hakkında bilgi edindik. Dinamik bellek kullanımı kavradık.

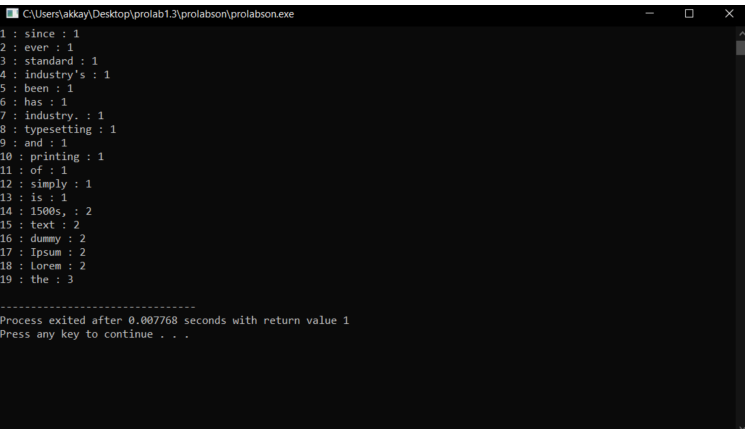
II. KARŞILAŞILAN PROBLEMLER

İlk karşılaştığımız problem manuel olarak bir sonraki düğümü null olarak kabul etmemiz gerektiğiydi, bunu araştırmalarımız sonucunda çözdük.

III. GENEL YAPI

KULLANICI KISMI

Proje başlatıldığında metin belgesinden bütün kelimeler tek tek alınır ve ekrana tekrar etme sayısı artacak şekilde basılır.



ALGORİTMA KISMI

Metin belgesinden okunan kelime düğümümüzün içinde yoksa düğüme kelime sayısı bir olmak üzere eklenir, eğer varsa düğümden bu kelime silinir. Kelime sayısına bir eklenerek tekrar düğüme eklenir ve en sonda ekrana basılır

```
Dugum* ekle(Dugum* dugum, char kelime[]) {
    int kelimeSayisi = sayac(dugum, kelime);
    if (kelimeSayisi != 1)
    {
        sil(dugum, kelime);
    }
    if (dugum == NULL)
    {
        dugum = (Dugum*)malloc(sizeof(Dugum));
        dugum->sonraki = NULL;
        dugum->kelime = malloc(strlen(kelime) + 1);
        strcpy(dugum->kelime, kelime);
        dugum->kelimeSayisi = 1;
        return dugum;
    }
    if (dugum->kelimeSayisi > kelimeSayisi)
    {
        Dugum* gecici = (Dugum*)malloc(sizeof(Dugum));
        gecici->kelimeSayisi = kelimeSayisi;
        gecici->kelime = malloc(strlen(kelime) + 1);
        strcpy(gecici->kelime, kelime);
        gecici->sonraki = dugum;
        return gecici;
    }
    Dugum* iter = dugum;
    while (iter->sonraki != NULL && iter->sonraki->kelimeSayisi < kelimeSayisi)
    {
        iter = iter->sonraki;
    }
    Dugum* gecici = (Dugum*)malloc(sizeof(Dugum));
    gecici->sonraki = iter->sonraki;
    iter->sonraki = gecici;
    gecici->kelimeSayisi = kelimeSayisi;
    gecici->kelime = malloc(strlen(kelime) + 1);
    strcpy(gecici->kelime, kelime);
    return dugum;
}
```

Ekle fonksiyonunda kelime sayısı döner. Eğer kelime sayısı bire eşit değilse bu kelime sil fonksiyonu yardımıyla silinir ve yeni kelime sayısıyla eklenir.

Düğüm boşsa bellekte yer ayrılır düğüm oluşturulur, düğümün ilk elemanına kelime eklenir ve kelime sayısı bir olur.

Eğer yeni kelimenin sayısı, ilk düğümün kelime sayısından küçükse yeni kelime başa eklenir.

Sonraki düğüm boş değilse ve sonraki düğümün kelime sayısı şimdiki kelime sayısından küçükse iter sonraki düğüme geçer.

Geçici düğüm tanımlanır ve bunun sayesinde kelime duruma göre araya veya sona eklenir.

```c
int sayac(Dugum* dugum, char kelime[]) {
    Dugum* iter = dugum;
    int kelimeSayisi = 1;
    if (strncmp(iter->kelime, kelime, 100) == 0) {
        kelimeSayisi = iter->sonraki->kelimeSayisi;
        kelimeSayisi++;
    }
    while (iter->sonraki != NULL) {
        if (strncmp(iter->sonraki->kelime, kelime, 100) == 0)
        {
            kelimeSayisi = iter->sonraki->kelimeSayisi;
            kelimeSayisi++;
        }
        iter = iter->sonraki;
    }
    return kelimeSayisi;
}
```

Sayaç fonksiyonunda o kelime mevcut düğümde daha önceden eklenme durumu kontrol edilir.

Eğer daha önceden mevcut olan düğümümüzde bu kelime varsa kelime sayısı bir arttırılır, fonksiyonun sonunda da kelime sayısı döndürülür.

```c
Dugum* sil(Dugum* dugum, char kelime[]) {
    Dugum* gecici;
    Dugum* iter = dugum;
    if (strncmp(dugum->kelime, kelime, 100) == 0)
    {
        gecici = dugum;
        dugum = dugum->sonraki;
        free(gecici);
        return dugum;
    }
    while (iter->sonraki != NULL)
    {
        if (strncmp(iter->sonraki->kelime, kelime, 100) == 0)
        {
            gecici = iter->sonraki;
            iter->sonraki = iter->sonraki->sonraki;
            free(gecici);
            return dugum;
        }
        iter = iter->sonraki;
    }
}
```

Sayac fonksiyonunda bir dönmüyorsa bu fonksiyon çalıştırılır.

Başta düğümünün ilk kelimesi bu kelimeye eşit mi diye bakılır, eğer eşitse bu eleman silinir ve fonksiyondan çıkılır.

Iter, sırayla bütün düğümleri gezer. Eğer sonraki düğümdeki kelime bu kelimeye eşitse bu düğüm silinir ve fonksiyondan çıkılır.

```c
void EkranaBas(Dugum* dugum) {
    Dugum* iter = dugum;
    int i=1;
    while (iter)
    {
        if (iter->kelime != "")
        {
            printf("%d : %s : %d\n", i, iter->kelime, iter->kelimeSayisi);
            i++;
        }
        iter = iter->sonraki;
    }
}
```

En sonda da bu fonksiyon ile iter bütün düğümleri gezer ve o düğümdeki istenilen verileri ekrana basar.

IV. KAYNAKÇA

https://notpast.com/c_programlama/Bagli-Listeler-72.html

https://www.youtube.com/watch?v=r3uOBb3BM-0&list=PLh9ECzBB8tJN9bckI6FbWB03HkmogKrFT

http://embedded.kocaeli.edu.tr/?page_id=91 (Bağlı Listeler PDF'i)


BAŞLA
Metin belgesinden kelimeler okunur.
Kelimenin duğümde olup olmadığı kontrol edilir
True
Kelime sayisi artırılır
False
Kelime sayısı aynı kalır
Kelime sayısı bire eşit olup olmadığı kontrol edilir
Kelime silinir ve yeni kelime sayisiyla tekrardan eklenir
Kelime eklenir
Ekrana kelimeleri ve kelime sayilarini bas
SON