Kocaeli Üniversitesi

Bilgisayar Mühendisliği Bölümü

Programlama Laboratuvarı 1

Proje 1

Muhammad Abdan SYAKURA (200201147) – Robera Tadesse GOBOSHO (190201141)

prof.syakur@gmail.com – robtad318@gmail.com

Özet

Bu rapor Programlama Laboratuvarı 1 dersi 1. Projesinin çözümünü açıklamak üzere hazırlanmıştır. Bu projede C dilinde dosyalama fonksiyonları, ikili dosyaları *(binary file)* ve metin dosyaları *(text file)* kullanarak basit Öğrenci Veri Sistemi geliştirilmiştir. Projede veri aramada aranan veriye daha hızlı ulaşmak için yoğun indeks *(dense index)* yapısıyla ikili arama *(binary search)* faydalanarak tasarlanmıştır. Öğrenci kayıtları ikili dosyası (.bin) olarak ve indeks dosyası metin dosyası (.txt) olarak kaydedilmektedir.

Giriş

Bu çalışmanın amacı ikili dosya ve metin dosyası ile C dilinde dosyalama fonksiyonlarının pratiğini yapmaktır. Bizden istenilen indexDosyasiOlustur, kayitEkle, kayitBul, kayitSil, kayitGuncelle, veriDosyasiniGoster, indeksDosyasiniGoster ve indeksDosyasiniSil fonksiyonları içeren basit bir Öğrenci Veri Sistemi programı yapmaktır.

Bu proje kapsamında veri dosyasında bir okuldaki öğrencilere ait notlar saklanmaktadır. Her öğrenci için aşağıdaki struct yapısında bilgi saklanmaktadır. Bu yapıda ogrNo, dersKodu ve puan alanları bulunmaktadır. Ancak projeyi gerçekleştirirken ihtiyaç duyulabileceği başka alanlar da tanımlanabilmektedir.

```
struct kayit{
        int ogrNo;
        int dersKodu;
        int puan;
        .
        .
}
```

Bir öğrenci birden fazla derslere katılabildiği için birden çok kayıt olabileceğini göz önünde bulundurarak koddaki fonksiyonları da ona göre yazdık.

Öğrencilerin kayıtlarını saklayan veri dosyası *binary file (kayitlar.bin)* olarak tanımlanmıştır. Bu dosyanın içinde istenilen gibi tanımladığımız struct kullanılarak oluşturulmuş 50 tane öğrencinin kaydı bulunmaktadır.

Bu projede de *dense index* yapısı kullanılması istenmiştir. Bu indeks yapısında ikili dosya olarak gerçekleştirilen veri dosyasındaki her kayıt için indeks dosyasında bir girdi saklanmaktadır. İndeks dosyasında

kayıtlar “anahtar” ve “adres” şeklinde saklanmaktadır. Adres değeri anahtarla ilgili bilginin veri dosyasında hangi offsette saklandığını göstermektedir. Veri dosyasında aynı anahtarla ilgili birden çok kayıt olabileceği için indeks dosyasında aynı anahtara ait birden fazla kayıt bulunabilir. Aşağıdaki görselde *dense index* yapısı için bir örnek verilmektedir. Bu programda ogrNo anahtar olarak tanımlanmıştır. Her anahtar indeks dosyasında yazılmaktadır.

Anahtar	Adres
100	2x
101	0
101	3x
101	6x
102	4x
103	5x
105	x
.	.
.	.

Şekil 1 Indeks Dosyası

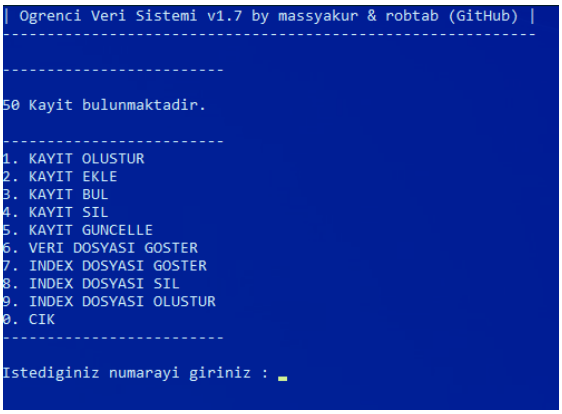
Offset	Anahtar
0	101
x	105
2x	100
3x	101
4x	102
5x	103
6x	101
.	.
.	.

Şekil 2 Veri Dosyası

Bu indeks yönteminde veri dosyasından (kayitlar.bin) ogrNo (anahtar) ve anahtarın adresini sıralayarak index dosyasına yazılmaktadır. Görselden de anlaşıldığı gibi indeks dosyasındaki kayıtlar anahtara göre sıralı iken veri dosyasındaki kayıtlar sıralı değildir. Böyle bir indeksleme yapısı veri arama işlemlerinin hızını oldukça hızlandırmaktadır. İndeks dosyası kullanılmadan bir kayıt veri dosyasında aranacak olsa tüm kayıtlar dosya baştan sona okunması gerekir. Veri dosyasındaki kayıtların büyük ve çok sayıda kayıt olduğu düşünüldüğünde veri dosyasının baştan sona okunması çok fazla sayıda disk erişimi gerektirir. Oysaki indeks dosyası kullanıldığında, dosyalar arama anahtarına göre sıralı olduğundan ikili arama yapılabilir. Diğer taraftan indeks dosyasındaki kayıtlar veri dosyasındaki kayıtlara göre çok daha küçük olduğundan indeks dosyasının belleğe aktarılması çok daha az disk erişimi gerektirir. Bundan dolayı indeks dosyası yapmayı tercih edilmiştir.

Yöntem

Öncelikle program başladığında Resim 1’deki gibi bir konsol açılacaktır.



Resim 1

Daha önce 50 kayıtlık *kayitlar.bin* oluşturduğumuz için “50 Kayit bulunmaktadir.” yazıyor. Main fonksiyonun başında *counter()* diye bir fonksiyon yazdık. Bu fonksiyon *kayitlar.bin’deki* tüm kayıtları hesaplayıp integer tipinde count diye bir değişkene atandıktan sonra ekrana kaç tane kayıt olduğunu yazdırır. Bu değişken başka fonksiyonlarda kullanılabilsin diye public olarak tanımladık.

Ayrıca *binary files,* dizilerin yapısına çok benziyor. Yalnız binary files diskte, diziler bellekte tutulmaktadır. *Binary files* diskte olduğundan çok büyük sayıda koleksiyonlar oluşturulabilir (kullanılabilir disk alanınıza göre sınırlıdır). Bu dosya kalıcı ve her zaman kullanılabilirler. Tek dezavantajı, disk erişim

süresinden kaynaklanan yavaşlıktır. Üstelik metin dosyasına göre insanlar tarafından okunamadığı için ve yapısı bilinmiyorsa veri elde edilemeyebileceğinden içeriğin güvenliğine de katkıda bulunmaktadır (2).

Bundan sonra while döngüsü kullanarak 0 girene kadar menüler görüntüleyecektir. Switch case kullanarak menüleri seçilebilir (1999).

1. *kayıtOlustur()*

İlk olarak yeni kayıt oluşturmak için 1. menüyü seçebilirsiniz. Bu menü *kayitOlustur()* fonksiyonu çağırır. Bu fonksiyon da structtan s diye bir pointer tanımlar. Kullanıcıdan kaç tane öğrenci girileceğini sorup s'yi o sayıya göre calloc fonksiyonu ile dinamik olarak tanımlar. Sonra da daha önce tanımlanan file pointeriyle *kayitlar.bin* oluşturur. Burada binary file olduğu için "wb" kullandık. Kullanıcı da girilen sayıya göre öğrenci numarası, ders kodu ve puanı tek tek yazacaktır. Döngünün sonunda binary file olduğundan *fwrite* fonksiyonu ile bu girilen veri struct olarak *kayitlar.bin* dosyasına kaydedilir. Dosya *fclose* ile kapatılır. *counter()* ve *indexDosyasiOlustur()* fonksiyonları çağırılır. Böylece yeni veriler oluşturulduğunda otomatik olarak hesaplayıp sıralı olarak index.txt dosyası oluşturur.

2. *indexDosyasiOlustur()*

Bu programda indeks dosyası, veri dosyasından alınan öğrenci numaralarını (index olarak) ve offset(adreslerini) saklamak için kullanılır.

İndeksleme, bir veri dosyasındaki belirli bir kaydın konumunu kısa sürede bulmaya yardımcı olur. Dosyalarda saklanan gerekli bilgileri sağlayarak hızlı karar verilmesine yardımcı olur.

örneğin, belirli bir kaydın aranması, güncellenmesi veya silinmesi gerektiğinde, onu bir indeks dosyasında aramak ve değişiklikleri uygulamak, aksi takdirde veri dosyalarındaki bir sürü veriyi gözden geçirerek boşa harcanacak zaman tasarrufu sağlar. Bir indeks dosyasında kayıt oluşturmak ve saklamak için: ilk olarak, 'ogrenci' tipinde bir struct dinamik olarak olusturulur. bundan sonra sırasıyla read(rb) ve write(w) modunda veri dosyası ve indeks dosyası açılır. Her kaydın öğrenci numarası ve ofseti veri dosyasından fread() fonksiyonu kullanılarak okunur ve bir while döngüsünde fprintf() fonksiyonu kullanılarak indeks dosyasına yazılır. Daha sonra indeks dosyasındaki kayıtlar, bubble sort sıralama algoritması kullanılarak sıralanmak üzere dinamik olarak başlatılan bir diziye aktarılır. sonunda, ofsetleriyle birlikte sıralanan indeksler, bir for döngüsünde fprintf() kullanılarak indeks dosyasına geri yazılır ve hem indeks hem de veri dosyaları kapatılır.

3. *kayitEkle()*

Bu fonksiyonda *kayitOlustur()* fonksiyonundaki tüm kodları kopyalayıp yapıştırdık. Yalnız "wb" yerine "ab" değiştirdik. Böylece var olan kayıtlar silinmeden en sonuna yeni kayıtlar eklenebilir.

4. *kayıtBul()*

kayıtBul() fonksiyonu, girilen öğrenci numarasının indeks dosyasındaki anahtar değerle eşleştiğinde veri dosyasından bir kayıt döndürür.

Bunu yapmak için, kayıtBul() fonksiyonu önce kullanıcıdan öğrenci numarasını alır ve değiştirilmiş ikili aramayı(modified binary search) kullanarak indeks dosyasında girilen öğrenci numarasını arar. İkili arama, aynı anahtara (öğrenci numarasına) sahip birden fazla kaydın olduğu koşullarda kullanılmak üzere değiştirildi. Belirtilen öğrenci numarası indeks dosyasında bulunursa, fonksiyon fseek() fonksiyonu ile indeks dosyasından alınan offset bilgisini kullanarak veri dosyasının dosya işaretçisini(file pointer) aranan kaydın konumuna taşır. Daha sonra fread() fonksiyonu kullanılarak veri dosyasından kayıt okunur. Son olarak, fonksiyon aranan kaydı biçimlendirilmiş bir şekilde görüntüler. Eğer girilen öğrenci numarası için herhangi bir kayıt bulunamazsa, fonksiyon "kayıt bulunamadı" mesajını yazdırır.

5. ***veriDosyasıGoster()***

VeriDosyasiGoster() fonksiyonu çağrıldığında veri dosyasındaki tüm kayıtları görüntüler.

Bunu yapmak için, bir dosya işaretçisi ve 'ogrenci' tipinde bir struct bildirilir. Daha sonra veri dosyası ikili okuma modunda (rb) açılır. Daha sonra fread() fonksiyonunu bir while döngüsünde koşul olarak kullanarak fonksiyon, veri dosyasındaki her kaydı döngüde printf kullanarak ekrana yazdırır. fread while döngüsünde kullanıldı, çünkü onu while(fread(&s1, sizeof(ogrenci), 1, file)) şekilde çağırmak dosya işaretçisini her seferinde struct ogrenci boyutuna göre ilerletir.

Deneysel Sonuçlar

Program ilk çalıştığında kayitlar.bin olup olmadığını kontrol eder. Dosya yoksa Resim 2'deki gibi gösterecek.

```
| Ogrenci Veri Sistemi v1.7 by massyakur & robtab (GitHub) |
------------------------------------------------------------

kayitlar.bin DOSYASI BULUNAMADI
0 Kayit bulunmaktadir.

-----------------------
1. KAYIT OLUSTUR
2. KAYIT EKLE
3. KAYIT BUL
4. KAYIT SIL
5. KAYIT GUNCELLE
6. VERI DOSYASI GOSTER
7. INDEX DOSYASI GOSTER
8. INDEX DOSYASI SIL
9. INDEX DOSYASI OLUSTUR
0. CIK
-----------------------

Istediginiz numarayi giriniz :
```

Resim 2 – kayitlar.bin bulunamadı

```
Istediginiz numarayi giriniz : 2
Kac tane ogrenci gireceksiniz : 1

Ogrenci numarayi giriniz : 100
Ders kodunu giriniz : 505
Puan giriniz : 87


-------------------------

  50 Kayit bulunmaktadir.
  1 tane kayit basariyla eklendi
  index.txt BASARIYLA OLUSTURULMUSTUR!

-------------------------
```

Resim 3 - kayıtEkle Fonksiyonu

```
-------------------------

Istediginiz numarayi giriniz : 3
ARAMAK ICIN OGRENCI NO YAZINIZ : 145

======================================
| Ogrenci No    Ders Kodu       Notu |
--------------------------------------
|     145          501           82  |
|     145          505           90  |
|     145          555          100  |
|     145          502           79  |
|     145          508           90  |
======================================
```

Resim 4 – kayıtBul öğrenci bulunduğunda listeler

```
Istediginiz numarayi giriniz : 3
ARAMAK ICIN OGRENCI NO YAZINIZ : 200

=======================================
| Ogrenci No    Ders Kodu       Notu |
---------------------------------------
|                                     |
|     200 NOLU OGRENCI BULUNAMADI!    |
|                                     |
=======================================
```

Resim 5 - Öğrenci bulunmadığında

```
Istediginiz numarayi giriniz : 4
SILMEK ICIN OGRENCI NO YAZINIZ : 100

100 numarali ogrenci basariyla silindi!

-------------------------

  49 Kayit bulunmaktadir.

  index.txt BASARIYLA OLUSTURULMUSTUR!

-------------------------
```

Resim 6 - Bir kayıt silindiğinde

```
Istediginiz numarayi giriniz : 4
SILMEK ICIN OGRENCI NO YAZINIZ : 100

=======================================
|                                     |
|     100 NOLU OGRENCI BULUNAMADI!    |
|                                     |
=======================================
```

Resim 7- Silmek için kayıt bulunmadığında

```
Istediginiz numarayi giriniz : 5
PUAN DEGISTIRMEK ICIN OGRENCI NO YAZINIZ : 145
555 nolu dersinin puani guncelle? (1 Evet/0 Hayir) : 1
Puani giriniz: 10

508 nolu dersinin puani guncelle? (1 Evet/0 Hayir) : 0
505 nolu dersinin puani guncelle? (1 Evet/0 Hayir) : 0
501 nolu dersinin puani guncelle? (1 Evet/0 Hayir) : 1
Puani giriniz: 20

502 nolu dersinin puani guncelle? (1 Evet/0 Hayir) : 1
Puani giriniz: 30
```

Resim 8 - Kayıt güncellerken

```
|     155          502          35  |
|     150          505          78  |
|     145          508          90  |
|     142          501          55  |
|     141          501          90  |
|     145          505          90  |
|     145          555         100  |
|     145          501          82  |
|     145          502          79  |
|     150          555          90  |
|     156          555          54  |
```

Resim 9 - Güncellemeden önceki veri

```
|     145          508          90  |
|     142          501          55  |
|     141          501          90  |
|     145          505          90  |
|     145          555          10  |
|     145          501          20  |
|     145          502          30  |
|     150          555          90  |
|     156          555          54  |
```

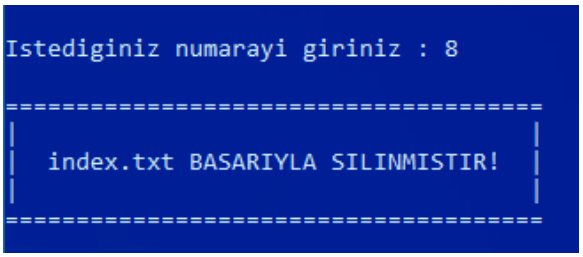
Resim 10 - Güncellendikten sonraki veri

```
-------------------------

Istediginiz numarayi giriniz : 6

---------------------------------------
| Ogrenci No    Ders Kodu       Notu |
---------------------------------------
|     102          505          97  |
|     105          501          45  |
|     106          502          90  |
|     107          555          95  |
```

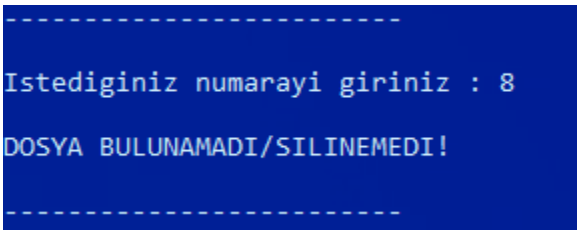
Resim 11 - Veri dosyası gösterimi

```
-------------------------

Istediginiz numarayi giriniz : 7

-------------------------
| Ogrenci No    Adres  |
-------------------------
|     100         624  |
|     101         640  |
|     102           0  |
|     103         656  |
|     104         672  |
```

Resim 12- İndeks dosyası gösterimi



Resim 13- index.txt sil fonksiyonu



Resim 14 - index.txt yoksa

Sonuçlar

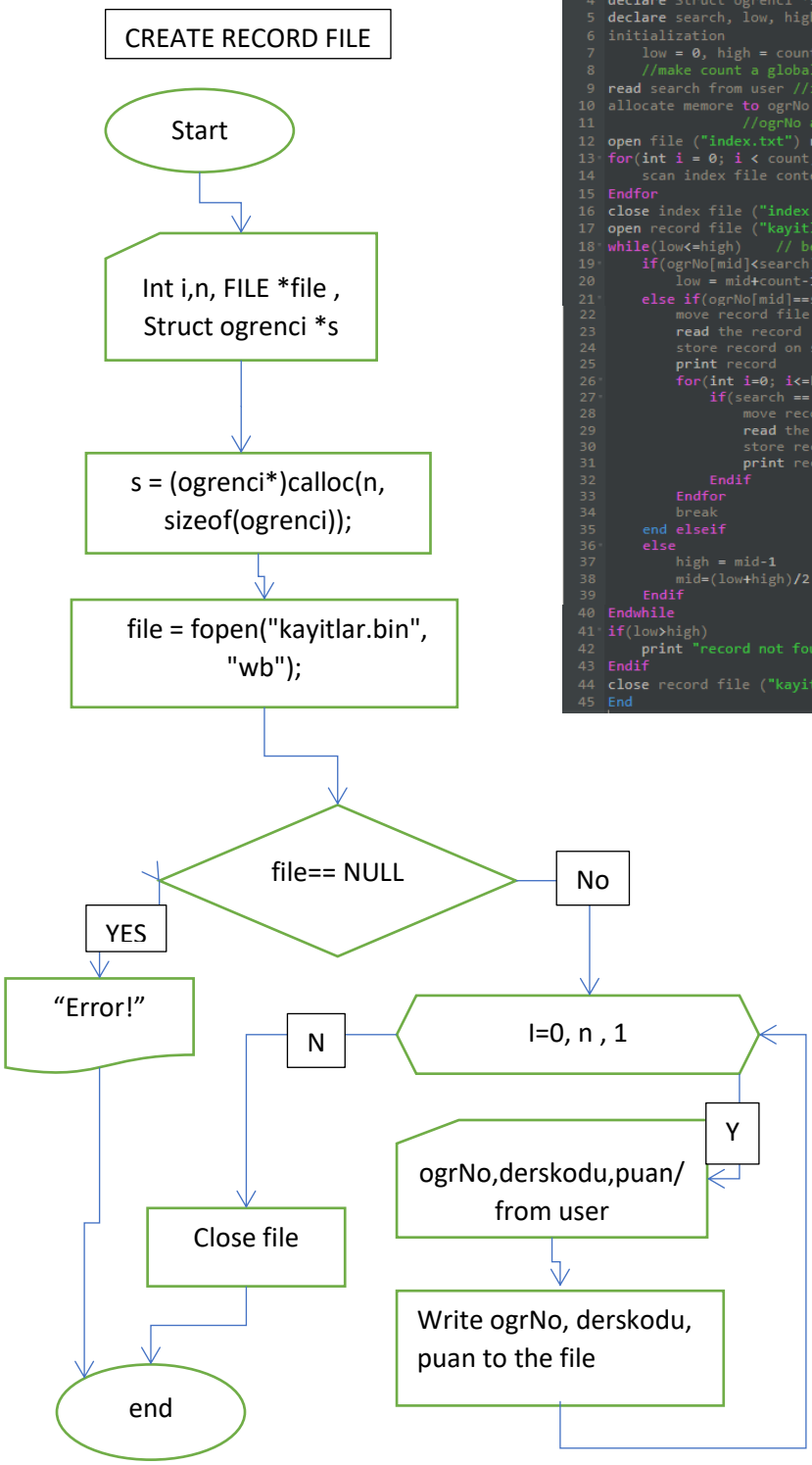
Program, farklı dosya işleme komutları ve yoğun indeksleme kullanarak dosyalar üzerinde çalışmaktadır. Program veri dosyası ve indeks dosyası oluşturabilir, veri dosyasındaki herhangi bir kaydı arayabilir ve görüntüleyebilir, yeni kayıtlar eklenebilir, belirli kayıtlar aranabilir, güncellenebilir veya silinebilir. Ayrıca program hem veri dosyasını hem de indeks dosyasını kullanıcıya gösterebilir. Üstelik indeks dosyası diskten silinebilir. Yukarıda bahsedilen prosedürü gerçekleştirmek için program, kullanıcının istenen komutu uygulamayı seçebileceği menüyü görüntüler.

Program Windows 10 işletim sisteminde CodeBlocks 20.03 üzerinden geliştirildi. GNU GCC ile derlenerek çeşitli testler yapılarak herhangi bir hata vermediği tespit edildi. Projenin gelişimi ve versiyonlarını takip edebilmek için Git versiyon kontrol sistemi kullandık.

Akış Şemaları ve Yalancı Kodları

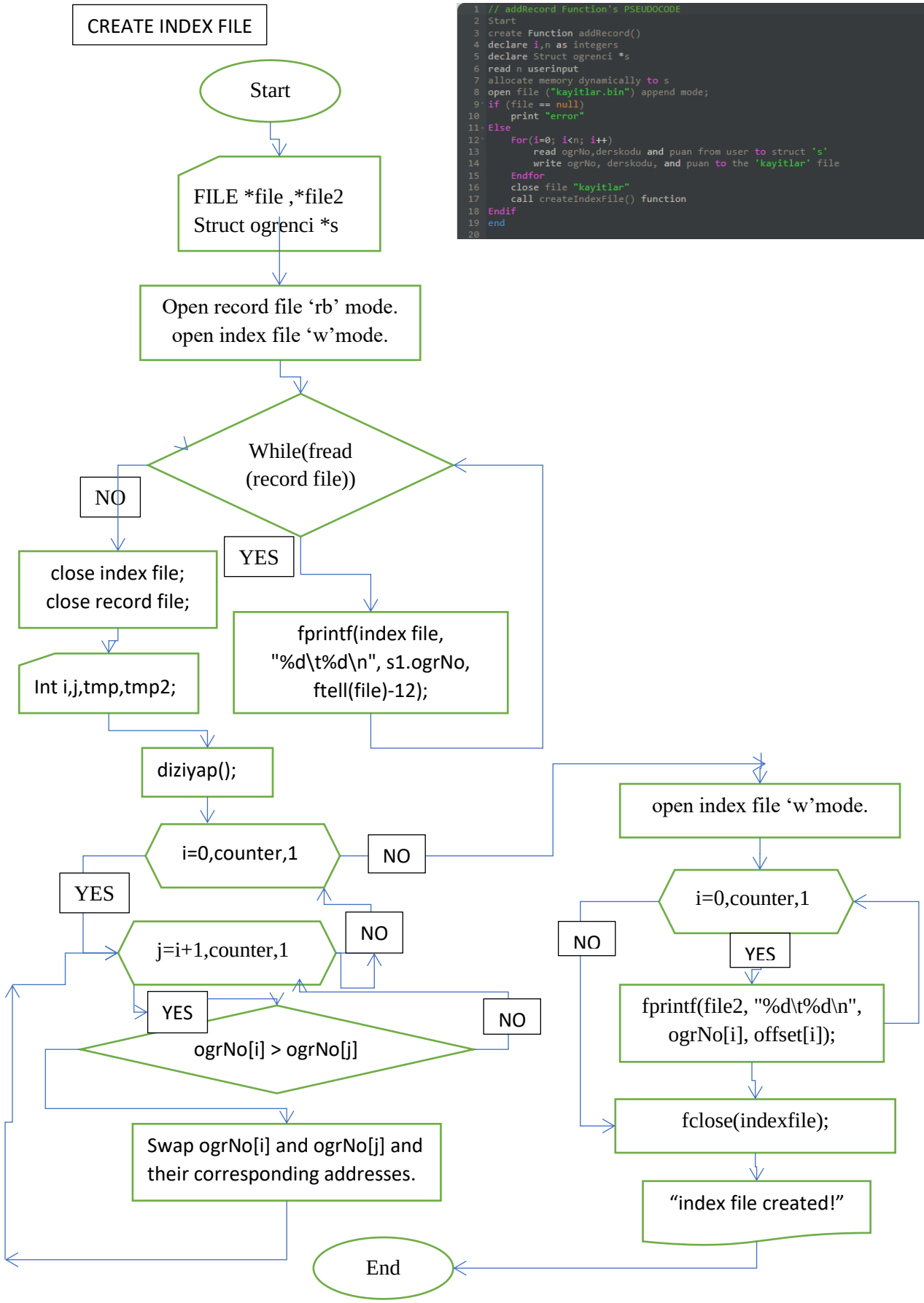
Resim 15 - kayitBul yalancı kodu

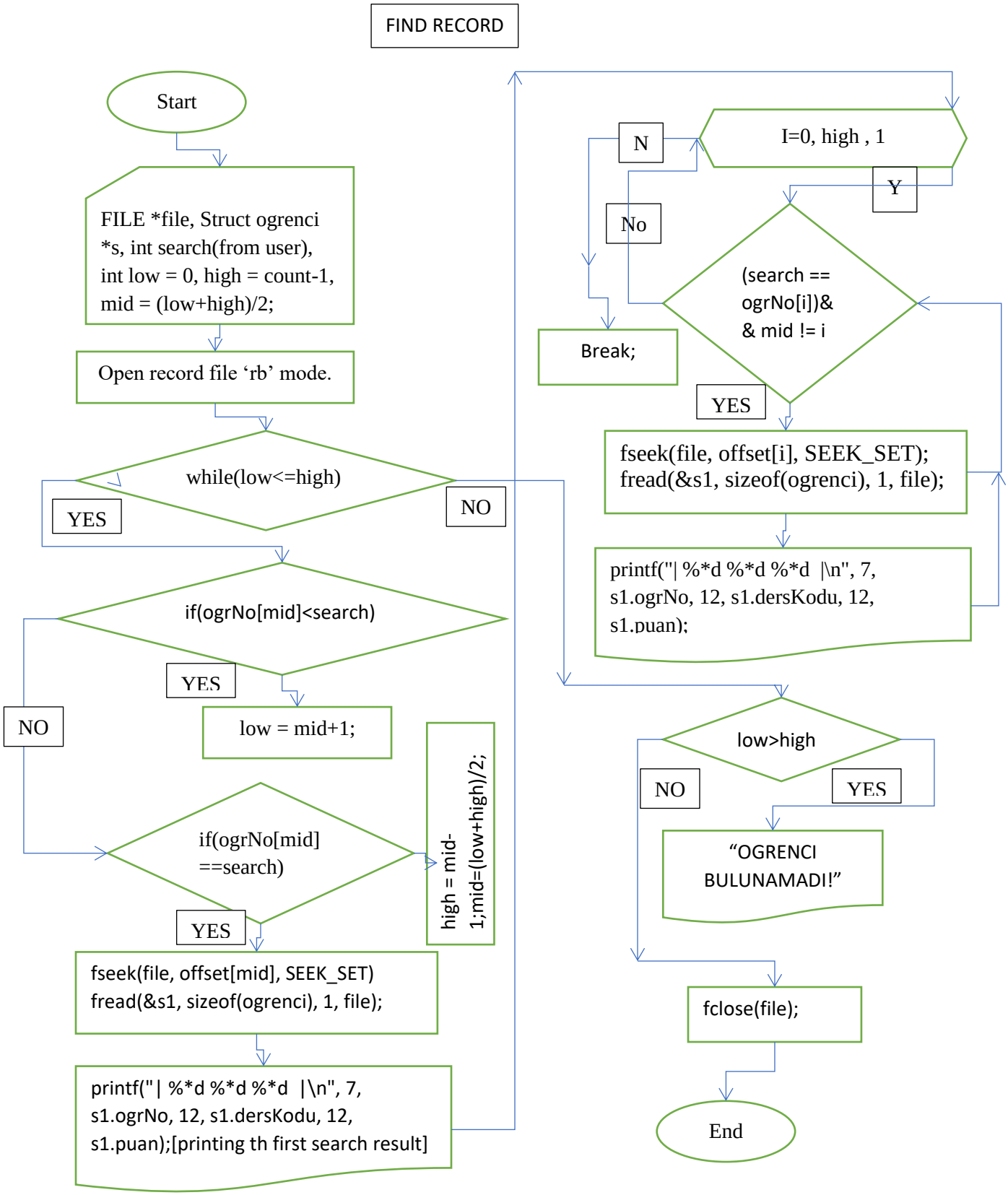
```
// findRecord Function's PSEUDOCODE
Start
declare filePointer
declare Struct ogrenci *s
declare search, low, high, mid as integers
initialization
    low = 0, high = count-1, mid = (low+high)/2
    //make count a global variable and store no of records on it
read search from user //it holds ogrNo to be searched
allocate memore to ogrNo and offset dynamically (count*sizeof(int))
                //ogrNo and offset are global variables
open file ("index.txt") read mode
for(int i = 0; i < count; i++)
    scan index file contents and store them on ogrNo[i] and offset[i]
Endfor
close index file ("index.txt")
open record file ("kayitlar.bin") read mode
while(low<=high)    // beginning of binary search algorithm
    if(ogrNo[mid]<search)
        low = mid+count-1
    else if(ogrNo[mid]==search)
        move record file pointer beginning of file by the offset of offset[mid]
        read the record
        store record on struct s
        print record
        for(int i=0; i<=high; i++) //incase there are multiple records with the same
            if(search == ogrNo[i] && mid != i)
                move record file pointer beginning of file by the offset of offset[mi
                read the record
                store record on struct s
                print record
            Endif
        Endfor
        break
    end elseif
    else
        high = mid-1
        mid=(low+high)/2
    Endif
Endwhile
if(low>high)
    print "record not found"
Endif
close record file ("kayitlar.txt")
End
```



Resim 16 - kayitEkle fonksiyonu yalancı kodu

```
// addRecord Function's PSEUDOCODE
Start
create Function addRecord()
declare i,n as integers
declare Struct ogrenci *s
read n userinput
allocate memory dynamically to s
open file ("kayitlar.bin") append mode;
if (file == null)
    print "error"
Else
    For(i=0; i<n; i++)
        read ogrNo,derskodu and puan from user to struct 's'
        write ogrNo, derskodu, and puan to the 'kayitlar' file
    Endfor
    close file "kayitlar"
    call createIndexFile() function
Endif
end

```




FIND RECORD
Start
FILE *file, Struct ogrenci *s, int search(from user), int low = 0, high = count-1, mid = (low+high)/2;
Open record file 'rb' mode.
while(low<=high)
YES
NO
if(ogrNo[mid]<search)
YES
NO
low = mid+1;
high = mid-1;mid=(low+high)/2;
if(ogrNo[mid] ==search)
YES
fseek(file, offset[mid], SEEK_SET)
fread(&s1, sizeof(ogrenci), 1, file);
printf("| %*d %*d %*d |\n", 7, s1.ogrNo, 12, s1.dersKodu, 12, s1.puan);[printing th first search result]
I=0, high , 1
N
Y
No
(search == ogrNo[i])& & mid != i
Break;
YES
fseek(file, offset[i], SEEK_SET);
fread(&s1, sizeof(ogrenci), 1, file);
printf("| %*d %*d %*d |\n", 7, s1.ogrNo, 12, s1.dersKodu, 12, s1.puan);
low>high
NO
YES
"OGRENCI BULUNAMADI!"
fclose(file);
End

Kaynakça

1. https://computer.howstuffworks.com/c39.htm (Binary Files)
2. https://www.codingeek.com/tutorials/c-programming/text-files-vs-binary-files-in-c-programming-language/
3. https://www.tutorialspoint.com/c_standard_library/c_function_fseek.htm
4. https://www.tutorialspoint.com/cprogramming/c_file_io.htm
5. https://www.programiz.com/c-programming/c-file-input-output
6. Tosun, N. (1998). *C dilinde kullanılan veri tabanları ve performansları* (Master's thesis, Fen Bilimleri Enstitüsü).
7. https://www.tek-tips.com/viewthread.cfm?qid=260813
8. https://pseudoeditor.com/ (Yalancı Kod oluşturmak için kullanıldı)
9. https://stackoverflow.com/
10. https://www.tutorialcup.com/dbms/indexing.htm
11. https://www.guru99.com/indexing-in-database.html
12. https://www.javatpoint.com/indexing-in-dbms