Kocaeli Üniversitesi

Bilgisayar Mühendisliği Bölümü

Programlama Laboratuvarı 1

Proje 1

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

[prof.syakur@gmail.com](mailto:prof.syakur@gmail.com) – [robtad318@gmail.com](mailto: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 |
| . | . |
| . | . |

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

Şekil 1 Indeks Dosyası Şekil 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.

Text

Description automatically generated

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* fonksiyonuile 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.

1. ***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 dosyası kapatılır.*

1. ***kayitEkle()***

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

1. ***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.*

1. ***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.*

Ekle

İndex, yöntem kısmında daha ayrıntılı anlatılacaktır

**Deneysel Sonuçlar**

**Sonuçlar**

Txt file her sayı 1 byte

Akış seması ektedir.

Binary search anlat logn

Dense index

Başka ide ve sistemlerde dene

Başka başarıyla oluşturulmuştur yaz

akış diyagramı veya yalancı kod içeren, özet, giriş, yöntem, deneysel sonuçlar, sonuç ve kaynakça

**Kaynakça**

1. <https://computer.howstuffworks.com/c39.htm> (Binary Files)
2. URL 2: <<https://www.codingeek.com/tutorials/c-programming/text-files-vs-binary-files-in-c-programming-language/>> Erısım Tarihi: 19.10.2021
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>