Kocaeli Üniversitesi Bilgisayar Mühendisliği Bölümü

Programlama Laboratuvarı I - Proje 1

İbrahim İnce - 200202102

1. Özet

İkili dosyaların ve text dosyalarının kullanıldığı bu projede, yoğun index yapısı kullanılarak bir okuldaki öğrencilerin numaralarının, aldığı derslerin ders kodlarının ve bu derslerden aldıkları puanların kaydedilmesi sağlanmıştır. Kullanıcının eklediği değerleri veri dosyasına eklemek, veri dosyasındaki kaydı ekrana yazdırmak, kullanıcının belirttiği kayıtları silmek ve güncellemek, index dosyası oluşturmak ve silmek, veri dosyasındaki kayıtları ekrana yazdırmak projedeki isterlerdir.

2. Giriş

Proje, C programlama dili ve CodeBlocks IDE'si kullanılarak hazırlanmıştır. Binary dosyası olarak öğrenci verilerinin depolanacağı "veri.bin" dosyası, text dosyası olarak da indexleme işlemi sonrası bilgilerin kaydedileceği "index.txt" dosyası kullanılmıştır.

3. Yöntem

İlk olarak "main()" fonksiyonu içerisinde isterleri gerçekleştirecek fonksiyonları çağıracak menü hazırlanmıştır. "Struct" yapısında öğrenci numaralarını tutacak "ogrNo", ders kodlarını tutacak "dersKodu" ve öğrencilerin o dersten aldıkları puanları tutacak "puan" değişkenleri tanımlanmıştır.

-"kayitEkle()"

Veri dosyası yoksa oluşturmak, varsa da dosyaya kayıt eklemek için kullanılan bu fonksiyonda kullanıcıdan eklemek istediği öğrencinin bilgileri alınır. Ardından "veri.bin" binary dosyası ekleme modunda (ab) açılır. Alınan bilgiler veri dosyasına "fwrite()" fonksiyonu ile yazdırılır ve dosya kapatılır.

-"veriDosyasiniGoster()"

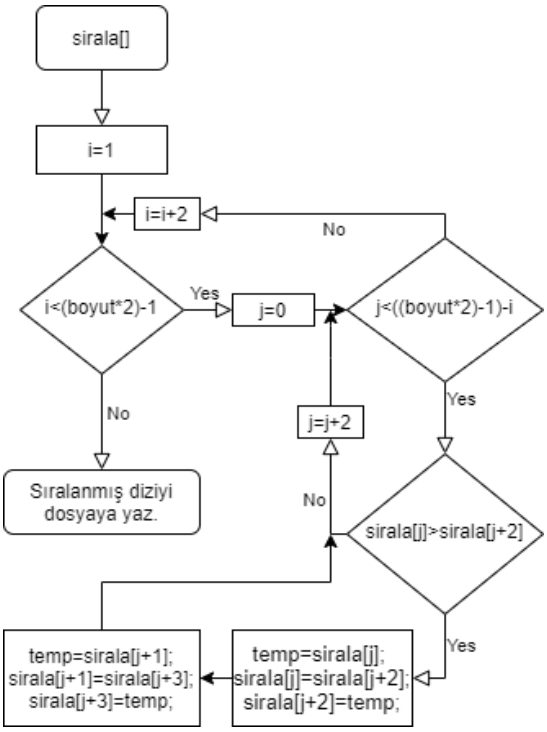
"veri.bin" dosyası okuma modunda (rb) açılır. "while()" döngüsü içerisinde "fread()" fonksiyonu ile veri dosyasındaki bilgi dizisi bir struct dizisine atanır. "if" fonksiyonu ile dosyadan bilgi okuma işleminin başarılı olduğunu kontrol ettikten sonra "printf()" fonksiyonu ile alınan bilgiler ekrana yazdırılır. Tüm bilgiler okunup ekrana yazdırıldıktan sonra dosya kapatılır.

-"indexDosyasiOlustur()"

"veri.bin" dosyası okuma modunda (rb) açılır. "index.txt" dosyası da yoksa oluşturulması, varsa da içindeki bilgilerin silinip yenilerinin yazılabilmesi için yazma modunda (w) açılır. Dosyaların başarılı bir şekilde açıldığı kontrol edildikten sonra "while()" döngüsü içerisinde veri dosyasındaki bilgi dizilerinin kaç kez başarılı bir şekilde okunduğu sayılır ve bu değer "boyut" değişkenine atanır.

“malloc()” fonksiyonu kullanılarak gerekli boyutta “sirala[]” dizisi oluşturulur. “fseek()” fonksiyonu ile dosya işaretçisi dosyanın başına alınır. Dosyadan bilgi okuma işlemi tekrar yapılarak okunan struct dizilerinden “ogrNo” bilgisi “sirala[]” dizisinin sıfırıncı indexine atanır. “ftell()” fonksiyonu ile dosya işaretçisinin konumu alınır ve konum okunan bilginin başını gösterecek şekilde düzenlenip dizinin birinci indexine atanır. Dosyadaki tüm veriler okunur ve bu şekilde gerekli bilgiler “sirala[]” dizisine atılmış olur. Veri dosyası kapatılır.

“Bubble sort” algoritması kullanılarak dizideki çift sayılı indexler birbirleri ile karşılaştırılarak küçükten büyüğe sıralanır. Anahtar ve offset bağlantısının korunması için değiştirilen değerlerden birer sonraki değerler de kendi aralarında değiştirilir. “for()” döngüsü ile dizideki anahtar ve offset bilgileri “index.txt” dosyasına yazılır. “sirala[]” dizisine ayrılan bellek alanı serbest bırakılır ve index dosyası kapatılır.



-“indexDosyasiniGoster()”

“index.txt” dosyası okuma modunda (r) açılır. “while()” döngüsü içerisinde “fscanf()” fonksiyonu ile okunan anahtar ve offset değerleri ekrana yazdırılır. İndex dosyası kapatılır.

-“indexDosyasiniSil()”

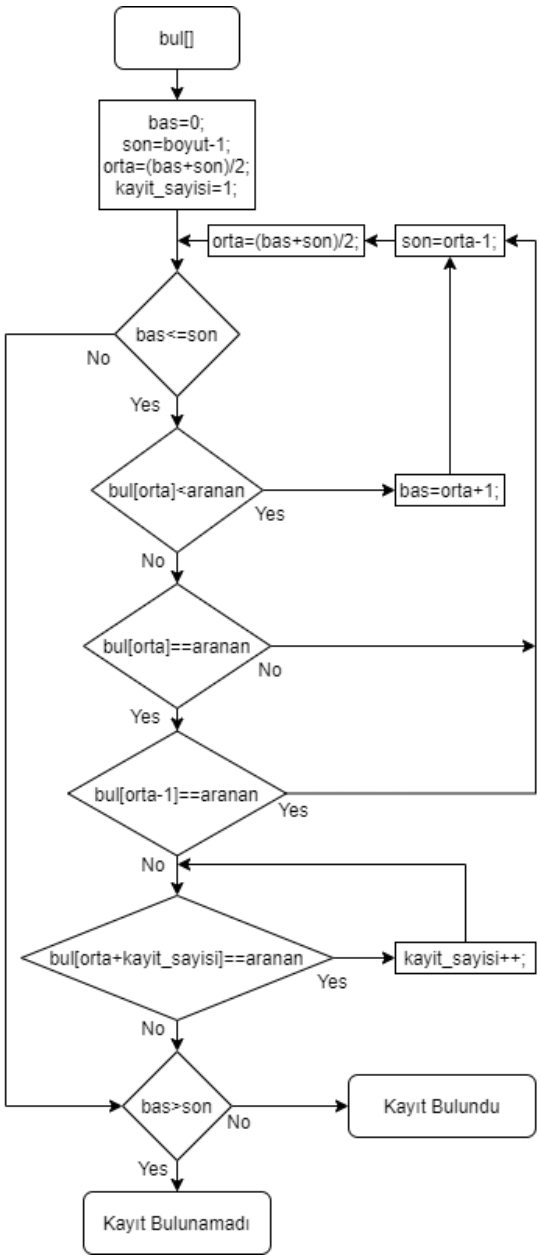
“remove()” fonksiyonu ile “index.txt” dosyası silinir. “if()” fonksiyonu ile işlemin başarılı olduğu kontrol edilir.

-“kayitBul()”

“scanf()” fonksiyonu ile kullanıcıdan bulmak istediği kaydın anahtar değeri (öğrenci numarası) alınıp “aranan” değişkenine atanır. “index.txt” dosyası okuma modunda (r) açılır. “while()” döngüsü ile text dosyasındaki bilgiler ikişerli okunur. Her başarılı okuma sonrası “boyut” değişkeni bir arttırılarak okunan veri sayısı bulunur.

“malloc()” fonksiyonu kullanılarak gerekli boyuttaki “bul[]” ve “os[]” dizileri oluşturulur. “fseek()” fonksiyonu ile dosya işaretçisinin konumu dosyanın başına alınır. “while()” döngüsü içerisinde index dosyasından ikişerli okunan değerlerden anahtar değeri “bul[]” dizisine, offset değeri de “os[]” dizisine atanır. Tüm değerler okunup, dizilere atanıp, “Binary Search” algoritması ile “bul[]” dizisindeki kayıtlardan aranan numara bulunur. Bulunan numaranın bulunduğu indexten bir önceki indexte de aynı numaranın olup olmadığı kontrol edilir. Eğer aynı numara varsa arama döngüsü aranan numarayı henüz bulmamış gibi devam eder. Bu sayede o öğrenciye ait ilk kayıt bulunmuş olur. Aranan numaranın bulunduğu indexin devamındaki indexler de taranarak

bu numaraya ait kaç kayıt olduğu bulunur ve "kayit_sayisi" değişkeninde saklanır. Aranan numaranın bulunabildiği kontrol edildikten sonra veri dosyası okuma modunda (rb) açılır. "for()" döngüsü içerisinde aranan numaraya ait kayıtlar ekrana yazdırılır. Ekrana yazdırılacak kaydın veri dosyasındaki yeri "fseek()" fonksiyonu ve "os[]" dizisindeki offset değeri kullanılarak bulunur. Sonrasında diziler için ayrılan bellek alanı serbest bırakılır ve dosyalar kapatılır.



-"kayitSil()"

"gecici.bin" dosyası yazma modunda (wb) açılır. "kayitBul()" işlemindeki gibi aranan kayıtlar bulunur. Kullanıcı tüm kayıtları silmek veya tek bir ders kaydını silmek için seçim yapar. Tüm kayıtların silinmesi istendiğinde "while()" döngüsü içerisinde "fread()" fonksiyonu ile "veri.bin" dosyasındaki kayıtlar "fread()" fonksiyonu ile okunur. Okunan kayıtlar "fwrite()" fonksiyonu ile "gecici.bin" dosyasına yazdırılır. Eğer okunan kayıtlarda aranan öğrenci numarasına ait kayıt var ise dosyaya yazdırma işlemi yapılmadan döngü devam eder. Tek bir ders kaydının silinmesi istendiğinde ise kullanıcıdan silmek istediği ders kodu alınır. Alınan değer "dk" değişkenine atanır. Aynı işlemler sırasıyla devam eder. Ancak okunan kayıtlarda aranan öğrenci numarasına ait kayıt var ise ek olarak bu kayıttaki ders kodu ile "dk" değişkenindeki ders kodunun da aynı olup olmadığı kontrol edilir. Eğer aynıysa "gecici.bin" dosyasına yazdırma işlemi yapılmadan döngü devam eder.

"veri.bin" dosyası "remove()" fonksiyonu ile silinir. "rename()" fonksiyonu ile "gecici.bin" dosyasının adı "veri.bin" olarak değiştirilir. Diziler için ayrılan bellek alanı serbest bırakılır ve dosyalar kapatılır. İndex dosyası oluşturma fonksiyonu çağrılır.

-"kayitGuncelle()"

Kullanıcıdan alınan yeni puan değeri "yenipuan" değişkenine atanır. "kayitSil()" işlemindeki tek bir ders kaydının silinmesi olayı gerçekleştirilir. Ancak aranan öğrenci kaydı ve ders kodu bulunduğunda kaydı yazdırmadan döngüye devam etmek yerine dosyaya yazılacak

olan kaydın puanı kullanıcıdan alınan değer ile değiştirildikten sonra kayıt dosyaya yazılır.

4. Deneysel Sonuçlar

"indexDosyasiOlustur()" işleminde "sirala[]" dizisi "Bubble Sort" algoritması ile küçükten büyüğe sıralanırken, doğru sonucu almak için algoritmada düzenlemeler yapılmıştır. "sirala[]" dizisinde tutulan değerler birbirleriyle bağlantılı ikili gruplar şeklinde olduğu için normal bir sıralama yöntemi doğru sonucu vermemiştir. Sıralama algoritmasının sadece çift sayıdaki indexleri kontrol etmesi gerektiği için "i" ve "j" kontrol değişkenleri her döngü sonunda iki birim arttırılmış ve ilk döngünün kontrol değişkeni olan "i" değerine başlangıçta "1" değeri atanmıştır. Böylece doğru sonuç elde edilebilinmiştir. Bunun yerine "kayitBul()" işlemindeki gibi birbiri ile bağlantılı iki değer, iki ayrı dizide aynı indexte tutulabilir ve sıralanmak istenen dizi "Bubble Sort" algoritması ile doğru bir şekilde sıralanabilirdi. Fakat iki yöntem de doğru sonucu verdiğinden ve zaten ikinci yöntem başka bir fonksiyonda kullanıldığından, birinci yönteme de projede yer verilmek istenmiştir.

"kayitBul()" işleminde bulunmak istenen öğrenci numarasına ait kayıt bulunurken "Binary Search" algoritması doğru kaydı buluyordu fakat çoğunlukla bulduğu kayıt öğrenciye ait ilk kayıt olmuyordu. Algoritmada "Dizinin ortasındaki indexte bulunan değer aranan değere eşit mi?" kontrolünün başarılı olması durumunda bulunan indexten bir önceki indexte de aynı değer olup olmadığı kontrol edilmiştir. Eğer aynı değer var ise, algoritmanın henüz aranan değeri bulamamış gibi çalışmaya devam etmesi sağlanmıştır.

5. Sonuç

Yoğun index yapısı kullanılarak veri depolayan bir program hazırlandı. Çok sayıda ve sırasız bir halde veriler bulunan dosyadan, kolay ve verimli bir şekilde kayıt bulup değiştirilebilmesi sağlandı. Var olan hazır algoritmalar ihtiyaca göre düzenlendi ve kullanıldı.

6. Kaynakça

[1] GeeksforGeeks. (2021). Binary Search, Erişim Tarihi: 17.10.2021, https://www.geeksforgeeks.org/binary-search/

[2] GeeksforGeeks. (2021). Bubble Sort, Erişim Tarihi: 17.10.2021, https://www.geeksforgeeks.org/bubble-sort/

[3] bilgigunlugum.net. C Programlama, Erişim Tarihi: 17.10.2021, https://www.bilgigunlugum.net/prog/cprog

[4] Akış Şeması, https://app.diagrams.net/