Kocaeli Üniversitesi

Bilgisayar Mühendisliği Bölümü

Programlama Laboratuvarı I

Öğrenci Kayıt Sistemi

Mustafa KARA -- Aleyna ŞANLI

Kocaeli Üniversitesi Bilgisayar Mühendisliği

190201119@kocaeli.edu.tr

190201080@kocaeli.edu.tr

Projenin Özeti

Bu projenin amacı, C dilinde dosyalama fonksiyonlarının kullanılarak bir öğrenci kayıt sistemi içinde pratiğe dökülmesi ve etkin, hızlı bir şekilde kullanılması için tasarlanması olarak amaçlanmaktadır. Öğrenci kayıt sistemleri günümüzde milyonlarca öğrencilerin verilerinin tutulması, düzenlenmesi, gerektiğinde bulunup farklı birimlere gönderilmesi ve kullanılması gibi birçok amaç için önem arz etmektedir. Bu derece fazla verinin efektif bir şekilde kullanılması için, *server* ile *client* arasındaki veri akışının olabildiğince sorunsuz, atılan *web request* sayısının olabildiğince çabuk ve karşılanabilir düzeyde olması gerektiğinden, server kısmında bu görevi üstlenen yazılımın algoritmasında zaman, verim ve bilgi kaybı olmamalıdır.

1.Giriş

Projenin geliştirilmesi için C programlama dili kullanılmıştır. C programlama dili düşük seviyeli dillere oranla daha anlaşılır, çok yüksek seviyeli dillere oranla daha fazla esnek bir dil olması sebebiyle arada bir noktada diyebiliriz. C dili makine diline yakın olması sebebiyle hem donanıma hitap eder hem de uygulama geliştirmede kullanılır. C ile programlamada bilgisayara daha fazla hakimiyet vardır. Daha az hazır fonksiyon ve kütüphane vardır. Bundan dolayı donanım hesaba katılarak programlama hesabı yapılması gerekmektedir.

2.Temel Bilgiler

2.1 Projenin İsterleri

Projenin isterlerinde belli başlı fonksiyonlar bulunmaktadır. İstenilen fonksiyonlar;

{

indexDosyasiOlustur;

kayitEkle;

kayitBul;

kayitSil;

kayitGuncelle;

veriDosyasiniGoster;

indeksDosyasiniGoster;

indeksDosyasiniSil;

}

Şeklinde ifade edilmiştir.

Şekil 2.1'de öğrenci verilerinin olması gereken struct yapı gösterilmiştir.

```
struct account{
    long studentID;
    int lessonID;
    int score;
```

Şekil 2.1

Projede istenilen, 50 adet Struct yapıda öğrencinin verilerini binary formatta saklar iken, aynı zamanda bu verilerin offset numaraları ve öğrenci numaralarının bir Txt uzantılı dosyada öğrenci numarasına göre sıralı bi şekilde yazılı istenmesidir. Böylece bu sıralı verilerin, *binary search* algoritması ile işlem süresini neredeyse yarısından fazla kısaltarak offsetin bulunup, ilgili binary dosyada dosyanın baştan sona okunmasına gerek kalmadan bellekten erişebilmesine olanak sağlar.

3. Main Metodu

Main fonksiyonumuz girişte Şekil 3.1'deki while dögüsü ile başlamaktadır.

```
int main() {

    FILE *fptr;
    if ((fptr = fopen( Filename: "data.bin",  Mode: "rb+")) == NULL) {
        printf( Format: "Error while opening the existing data.bin file");    // This is the proccess of reading the existing binary file for using sorting
        exit( Code: 1);                                                       // searching, deleting methods over memory via struct variables.
    }
    while(fread(&student[studentCount], sizeof (struct account),  Count: 1,fptr)==1){
        offsets[studentCount].studentID = student[studentCount].studentID;
        offsets[studentCount].offset = studentCount * sizeof( struct account );
        studentCount++;
    }
```

Şekil 3.1

Bu fonksiyonun işlevi kodun kritik parçalarından biri olmakla beraber açıklanması gerekirse, *studentcount* değişkenimiz, global olarak "0"tanımlı olmakla beraber, her binary dosyasında okuduğu değer için bu değişkenin değerini 1 arttırır. Bu değişken diğer metodlarda önemli bir noktaya sahiptir. Böylece öğrenci verisinin kaç tane olduğu ilgili bil global değişkende sonradan kullanılmak üzere hazır olmuş olur. Bir diğer yapılan işlem ise, her okunan öğrenci kaydı için offset belirlenerek bu değerler ayrı bi struct yapısında anahtar değeri ile kaydedilir.

Binary dosyayı okuma sonrasında kullanıcıya sorulan switch case yapısı gelmektedir. Burada kullanıcı klavyeden yapmak istediği seçeneğin kodunu girerek ilgili metodların çalışmasını sağlar.

3.1. Void sort() Metodu

Bu metod, geçici bir struct değişkenine *data.bin* dosyasındaki verileri okuyarak bir sıralama algoritmasında *Struct account student* değişkenindeki *studentCount* değişkeni kadar

veriyi sıralamayı hedefler. Şekil 3.1.1'de sıralama algoritması gösterilmiştir.

```
for (int i = 0; i < studentCount; ++i) {
    for (int j = 0; j <studentCount; ++j)
    {
        if(student[i].studentID>student[j].studentID)
        {
            temp2.studentID = student[j].studentID;
            temp2.lessonID = student[j].lessonID;
            temp2.score = student[j].score;

            student[j].studentID= student[i].studentID;
            student[j].lessonID= student[i].lessonID;
            student[j].score= student[i].score;

            student[i].studentID= temp2.studentID;
            student[i].lessonID = temp2.lessonID;
            student[i].score= temp2.score;
        }
```

Şekil 3.1.1

3.1.2 Void createIndex() Metodu

Bu metotta sıralanmış student değişkenindeki anahtar değerlerin, main fonksiyondaki while döngüsünde atanan offset değerleriyle beraber yazdırılması hedeflenmiştir. Şekil 3.1.2'deki kod parçacığı temsillendirilmiştir.

```
void createIndex(){
    FILE *fptr;
    fptr = fopen ( _Filename: "indexFile.txt",  _Mode: "w+");
    for (int i = 0; i < studentCount; ++i) {
        fprintf(fptr,  _Format: "%3d %ld\n",findOffset(student[i].studentID),
                student[i].studentID);
    }
    fclose(fptr);
```

Şekil 3.1.2

3.1.3 void findStudent() Metodu

Bu metodun başlıca görevi, girilen bir öğrenci numarasının *indexFile.txt* dosyasında mevcut olup olmadığını kontrol etmektir. Bu metodun başlıca özelliği, *binary search* algoritmasını kullanarak mevcut işlemi kısaltmayı amaçlamaktadır. Aşağıda Şekil 3.1.3.a ve 3.1.3.b'de bu algoritmanın kısa örneklendirmesi gösterilmiştir.

```
struct binOffset temp[studentCount];
long first = 0 ;
long last = studentCount-1;
long middle = (first + last) / 2;
```

Şekil 3.1.3.a

```
while (first <= last) {
    if (temp[middle].studentID > userID_input) {
        first = middle + 1;
    }
    else if (temp[middle].studentID == userID_input) {
        printf( _Format: "\n%ld ID Found at Offset %d.\n",userID_input,temp[middle].offset);
        break;
    } else
        last = middle - 1;
        middle = (first + last) / 2;
}
if (first > last)
    printf( _Format: "Not found! %ld\n", userID_input);
```

Şekil 3.1.3.b

3.1.4 Void addStudent() Metodu

Bu metodun amacı, binary dosyanın sonuna klavyeden yeni girilecek bir öğrencinin kayıtlarını eklemektir. *studentCount* değişkeni burada "1"değer kadar artırılıp memoryde tutulup ilgili fonksiyonlar çağırılarak işlemler tamamlanır. Dosyanın append modda açılması gerekir. Şekil 3.1.4.1'de ilgili kod gösterilmiştir.

```
studentCount++;
printf( _Format: "Enter ID:");
scanf( _Format: "%ld",&student[studentCount].studentID);
printf( _Format: "Enter LessonId:");
scanf( _Format: "%d",&student[studentCount].lessonID);
printf( _Format: "Enter Point:");
scanf( _Format: "%d",&student[studentCount].score);
fseek(fptr, _Offset: 12, SEEK_END);
fptr = fopen( _Filename: "data.bin", _Mode: "ab+");

fwrite(&student[studentCount], _Size: 12, _Count: 1,fptr);
fclose(fptr);
writenewOffset_of_newStudent();
sort();
createIndex();
```

Şekil 3.1.4.1

3.1.5 Void writenewOffset_of_newStudent() Metodu

Bu fonksiyonun amacı, her çalışan void addStudent metodunda girilen yeni bir öğrenciye ait yeni bir offset değeri verilmesidir. Aşağıda Şekil 3.1.5.1'de kodun ilgili kısmı gösterilmiştir.

```
void writenewOffset_of_newStudent(){
    FILE *fptr;
    if ((fptr = fopen( _Filename: "data.bin", _Mode: "rb+")) == NULL) {
        printf( _Format: "Error while opening the existing data.bin file");
        exit( _Code: 1);
    }
    for (int i = 0; i < studentCount; ++i) {
        fread(&student[i],sizeof(struct account), _Count: 1,fptr);
    }
    offsets[studentCount].offset= studentCount*sizeof (struct account);
    offsets[studentCount].studentID=student[studentCount].studentID;
}
```

Şekil 3.1.5.1

3.1.6 updateBinary() Metodu

Bu metodun amacı, ilerde göreceğiniz *updateScore* metodunun bir parçası olup ilgili offsetin bulunup puan değerinin değiştirilmesini sağlar. Şekil 3.1.6.1 de ilgili kod parçası gösterilmiştir.

```
void updateBinary(long arrayOffset,int offset,int score)
 {

     FILE *fptr;
     if ((fptr = fopen( _Filename: "data.bin",  _Mode: "rb+")) == NULL) {
         printf( _Format: "Error while opening the existing data.bin file");
         exit( _Code: 1);
     }

     fseek(fptr,offset,SEEK_SET);
     student[arrayOffset].score=score;
     fwrite(&student[arrayOffset], sizeof (struct account), _Count: 1,fptr);
     fclose(fptr);
}
```

Şekil 3.1.6.1

Bu metodun daha detaylı açıklanması gerekirse, aşağıda göreceğiniz updateScore metodunda kullanıcının girmiş olduğu studentID değerine ait offsetin hızlı bir şekilde bulunup, bu offsetin, yeni puanının fseek fonksiyonu ile bulunup değiştirilmesi amaçlanmaktadır. Mevcut binary dosya böylece değişikliğe uğramadan ilgili puan değişir.

3.1.7. updateScore() Metodu

Bu metodun amacı, index dosyasını geçici bir struct değişkenine geçirdikten sonra hızlı bir şekilde ilgili anahtar değerinin bulunup offsetin *updateBinary* ve yeni *score* değerinin yollanmasını sağlar. Aynı zamanda kullanıcı bu metodun içinde yeni puan değerini girmelidir. Şekil 3.1.7.1'de ilgili kod gösterilmektedir.

```
while (first <= last) {
    if (temp[middle].studentID > userID_input) {
        first = middle + 1;
    }
    else if (temp[middle].studentID == userID_input) {
        printf( _Format: "\nEnter new score: ");
        scanf( _Format: "%d", &newscore);
        int offsetReferance= findOffset(userID_input);
        updateBinary(middle, offsetReferance,newscore);
        break;
    } else
        last = middle - 1;
    middle = (first + last) / 2;
}
if (first > last)
    printf( _Format: "\nNot found! %ld\n", userID_input);
```

Şekil 3.1.7.1

Bu metodun amacına ulaşmasına rağmen, ilgili offsete ait öğrenci numarası listesinde sebebi anlaşılamayan yer değişiklikleri gerçekleşmektedir. İlgili sorunun kaynağı maalesef bulunamamıştır.

3.1.8 deleteStudent() Metodu

Bu metodun amacı, sıralama algoritması sayesinde hızlı bir şekilde ilgili öğrenci numarasının kaydı bulunup, bu numaraya ait offset no ile binary dosyada bu kaydın silinmesini hedeflemektir. Proje devamı olarak bu son fonksiyon maalesef proje sahipleri tarafından çözümlenememiştir.

3. KARŞILAŞILAN PROBLEMLER

Karşılaşılan problemlerden başlıca bir tanesi, bilgisayara bu proje için yaptığımız rasgele öğrenci kayıtlarının birbiriyle çakışma olup olmadığını kontrol eden yapılar projeye entegre edilememiştir. Bu durumun ufak bir çözümü olarak, rand fonksiyonuyla gelen rasgele değerlerin aralığı olabildiğince büyütülmüştür. Büyüyen bu aralık sayesinde, eş gelme olasılığı azaltılmıştır ancak bu projenin sürdürülebilirliğini tehlikeye atmaktadır.

İkinci karşılaşılan problem, değiştirilmek istenen puan verisinin amaca ulaşmasına karşın, anlaşılamayan bir nedenden dolayı öğrenci numaralarının sıralamaları değişmektedir. Sorunun çözümü için yöntemler sonuçsuz kalmakla beraber projenin sürekliliği kesilmiştir.

Son mevcut problem ise silinmek istenen öğrenci kaydının offsetleri bir indexte

bulunmasına rağmen, akabinde bu verinin silinmesi kolay olup, devamında diğer öğrencilerin verilerinin listede kayması problemine çözüm getirilememiştir.

Proje başlarında gelen problemler, öğrenci verilerini çıktı olarak görmek ister iken, aslında dosyadan değil, mevcut ramdeki verilerin gösterilmesinden dolayı dosyayı ayrı platformlarda inceleme zorunluluğu doğmuştur. Bu sorun proje devamında çözülmüştür.

SONUÇ

C programlama dili günümüzde bu kadar güçlü olduğunu bu proje sayesinde pratiğe dökerek, uygulamalarını, yapılan projeleri inceleyerek öğrenmiş olduk. Yapılan işlemlerin basit olmasına rağmen bu derece farklı detayları tek tek hesaplayıp bunu bir gömülü sisteme dökmenin gerçekten zor ve meşakkatli olduğunu anladık.

KAYNAKÇA

- https://www.programiz.com/c-programming/c-file-input-output
- https://www.programiz.com/c-programming/c-file-examples
- https://hackr.io/blog/binary-search-in-c
- https://fresh2refresh.com/c-programming/c-function/
- https://fresh2refresh.com/c-programming/c-structure-using-pointer/
- https://www.programiz.com/c-programming/c-goto-statement
- https://www.bilgigunlugum.net/prog/cprog/c_stdkut/stdio/fseek
- https://overiq.com/c-programming-101/array-of-strings-in-c/
- https://www.youtube.com/watch?v=iB4EOLYr67c
- https://programs.programmingoneonone.com/2021/02/hackerrank-pointers-in-c-solution.html
- https://web.cs.hacettepe.edu.tr/~maydos/Docs/c/dosyalar.pdf