Yıldız Teknik Üniversitesi Elektrik-Elektronik Fakültesi Bilgisayar Mühendisliği Bölümü

BLM1031

Yapısal Programlama

Gr: 2

Doç. Dr. Hafiza İrem TÜRKMEN ÇİLİNGİR

İsim : Şevval ÇABUK

No: 24011107

E-posta : sevval.cabuk@std.yildiz.edu.tr

Ön Bilgi

Bu projede, ilişkisel veritabanlarının temel çalışma mantığını modelleyen, dosya tabanlı ve bağlantılı liste (linked list) yapıları üzerine kurulu bir *kütüphane otomasyon sistemi* geliştirilmiştir.

Sistem; yazar, öğrenci ve kitap bilgilerinin ayrı ayrı yönetilebildiği; bu nesneler arasında anlamlı ilişkilerin kurularak saklandığı modüler bir yapıya sahiptir. Öğrenciler çift yönlü, yazarlar ve kitaplar ise tek yönlü bağlı listeler aracılığıyla tutulmuş; tüm veriler *.csv* dosyalarına eşzamanlı olarak işlenmiştir. Ayrıca kitapların ödünç alınıp teslim edilme süreçleri, raf durumlarının güncellenmesi ve öğrenci puanlarının takibi gibi dinamik işlemler, kullanıcı dostu bir menü aracılığıyla yönetilmektedir.

Projede, dinamik bellek yönetimi ve fonksiyonel soyutlama gibi ileri düzey C programlama teknikleri kullanılmış, veri bütünlüğü ise kapsamlı kontrollerle sağlanmıştır. Geliştirilen bu uygulama, küçük ölçekli bir kütüphane sisteminin temel işleyişini simüle etmektedir.

Struct Yapıları

```c
typedef struct Yazar {
    int yazarID;
    char ad[MAX_AD_SOYAD];
    char soyad[MAX_AD_SOYAD];
    struct Yazar* sonraki;
} Yazar;
```

`Yazar` yapısı, her bir yazarın sistemde benzersiz bir `yazarID` ile temsil edilmesini sağlar. Ad ve soyad bilgilerini içeren bu yapı, `sonraki` işaretçisi sayesinde tek yönlü bağlı liste oluşturur. Bu sayede yazarlar arasında sıralı ve dinamik bir veri yapısı elde edilmiştir.

```c
typedef struct Ogrenci {
    char ogrenciNo[OGRENCI_NO_UZUNLUK];
    char ad[MAX_AD_SOYAD];
    char soyad[MAX_AD_SOYAD];
    int puan;
    struct Ogrenci* onceki;
    struct Ogrenci* sonraki;
} Ogrenci;
```

`Ogrenci` yapısı, her öğrencinin sekiz haneli öğrenci numarası, adı, soyadı ve sistemdeki kütüphane puanı gibi bilgilerini içerir. `onceki` ve `sonraki` işaretçileri ile çift yönlü bağlı liste yapısı kurulmuş, böylece öğrencilerin hem ileriye hem de geriye doğru gezinerek erişimi sağlanmıştır.

```c
typedef struct Kitap {
```

```
    char ad[MAX_KITAP_AD];
    char isbn[ISBN_UZUNLUK];
    int adet;
    KitapKopya* kopyalar;
    struct Kitap* sonraki;
} Kitap;
```

`Kitap` yapısı, kitap adı, 13 haneli ISBN numarası ve kütüphanedeki toplam kopya sayısını içerir. Ayrıca her kitap için bağlı `KitapKopya` listesi bulunur. Bu yapı, aynı kitaba ait birden fazla fiziksel kopyanın durumlarını (`Rafta` ya da `ÖğrenciID`) izlemek için gereklidir.

```
typedef struct KitapKopya {
    char etiketNumarasi[ETIKET_NO_UZUNLUK];
    char durum[DURUM_UZUNLUK];
    struct KitapKopya* sonraki;
} KitapKopya;
```

`KitapKopya` yapısı, her bir kitap örneğini temsil eder. `etiketNumarasi` ile her kopya benzersiz şekilde tanımlanır. `durum` alanı, kopyanın rafta olup olmadığını veya hangi öğrenci tarafından ödünç alındığını belirtir. `sonraki` işaretçisiyle kitap kopyaları tek yönlü liste halinde saklanır.

```
typedef struct KitapYazarIliski {
    char isbn[ISBN_UZUNLUK];
    int yazarID;
} KitapYazarIliski;
```

`KitapYazarIliski` yapısı, bir kitabın bir veya birden fazla yazarla olan bağlantısını kurmak için kullanılır. Her ilişki, kitabın `isbn` numarası ile bir `yazarID` eşleştirilerek tanımlanır. Bu yapıların tümü, `KitapYazarVeriYoneticisi` adlı bir üst yapıda dizi olarak tutulur.

```
typedef struct OduncIslem {
    char ogrenciID[OGRENCI_NO_UZUNLUK];
    char kitapEtiketNo[ETIKET_NO_UZUNLUK];
    int islemTuru;
    char tarih[TARIH_UZUNLUK];
} OduncIslem;
```

`OduncIslem` yapısı, kitap ödünç alma ve teslim işlemlerini temsil eder. `ogrenciID`, `kitapEtiketNo`, `islemTuru` (0 = ödünç alma, 1 = teslim) ve işlem tarihi gibi bilgileri içerir. Bu işlemler `OduncVeriYoneticisi` yapısı altında dinamik dizi olarak saklanır.

```c
typedef struct {
    int gun;
    int ay;
    int yil;
} Tarih;
```

`Tarih` yapısı, ödünç işlemlerinin zaman takibini kolaylaştırmak için gün, ay ve yıl alanlarından oluşur. Ceza puanı hesaplamalarında teslim süresinin kontrolü bu yapı üzerinden gerçekleştirilir.

Fonksiyon Pointer Yapısı

Her bir veri tipi için iki tür fonksiyon işaretçisi tanımlanmıştır:

1. **İşlem Fonksiyonu İşaretçileri (Action Function Pointers) :** Bu işaretçiler, belirli bir liste elemanı üzerinde yapılacak işlemi temsil eder.

2. **Koşul Fonksiyonu İşaretçileri (Condition Function Pointers) :** Bu işaretçiler ise her eleman için bir koşulun sağlanıp sağlanmadığını kontrol eder. Koşul doğruysa (*return 1*), işlem gerçekleştirilir; değilse atlanır.

Aşağıda fonksiyon pointer'ların projede kullanım şekilleri verilmiştir:

```c
typedef void (*YazarIslemFonksiyonu)(Yazar*);

//
// Yazar işlemlerini sırayla gerçekleştirebilmek için oluşturulmuş yapı.
// İleride daha ayrıntılı şekilde açıklanacak.
//

typedef int  (*YazarKosulFonksiyonu)(Yazar*, void*);
//
// Yazarın gerçekleştirilmek istenen şartlara uygun olup olmadığının kontrolünü
// sağlayabilmek için oluşturulmuş kontrol yapısı.
//



//
// Yukarıda açıklanan türden öğrenci ve kitaplar için de ayrı ayrı fonksiyon
// pointerlar oluşturulmuştur :
//
```

```c
typedef void (*OgrenciIslemFonksiyonu)(Ogrenci*);
typedef int  (*OgrenciKosulFonksiyonu)(Ogrenci*, void*);

typedef void (*KitapIslemFonksiyonu)(Kitap*);
typedef int  (*KitapKosulFonksiyonu)(Kitap*, void*);
```

Ana (main) Fonksiyonu

```c
// --- main ---
int main() {
    Yazar* yazarListesi = NULL;
    Ogrenci* ogrenciListesi = NULL;
    Kitap* kitapListesi = NULL;
    int sonYazarID = 0;

//
// Bu satırlar, programın başlangıcında bellekte yazar, öğrenci ve kitap
// kayıtlarının tutulacağı linked list'leri tanımalr ve başlangıçta NULL olarak
// ayarlanır.
//

//
// sonYazarID, her yeni yazar eklemede kullanılacak otomatik ID oluşturma işlevi
// için bir sayaç görevindedir.
//



    KitapYazarVeriYoneticisi kitapYazarVerileri = {NULL, 0};
    OduncVeriYoneticisi oduncVerileri = {NULL, 0};

//
// Bu yapıların amacı, kitap-yazar ilişkileri ile ödünç alma işlemlerini bir
// dizi olarak tutmak ve bu işlemler üzerinde toplu kontrol sağlayabilmektir.
//



    printf("Kutuphane Otomasyonu Baslatiliyor...\n");
    tumVerileriYukle(&yazarListesi, &ogrenciListesi, &kitapListesi, &sonYazarID,
      &kitapYazarVerileri, &oduncVerileri);
    printf("\nVeriler yuklendi. Ana menuye geciliyor.\n");
```

```
//
// Bu fonksiyon, daha önceden .csv formatında saklanan tüm verileri program
// başlarken belleğe yükler.
//




    anaMenuGoster(&yazarListesi, &ogrenciListesi, &kitapListesi, &sonYazarID,
      &kitapYazarVerileri, &oduncVerileri);

//
// Kullanıcının kullanacağı menü gösterimi
//

    bellekTemizle(&yazarListesi, &ogrenciListesi, &kitapListesi,
      &kitapYazarVerileri, &oduncVerileri);
    printf("Program basariyla sonlandirildi.\n");

//
// Program bitirilmeden önce yapılan bellek temizliği.
//

    return 0;
}
```

Fonksiyonların Tanımları ve Tanımlarının Açıklanması

Input Alma

```
void guvenliInputAlma(char* buffer, int size) {
    if (fgets(buffer, size, stdin) != NULL) {
        size_t len = strlen(buffer);
        if (len > 0 && buffer[len - 1] == '\n') {
            buffer[len - 1] = '\0';
        } else {
            int ch;
            if (len == size - 1 && buffer[len - 1] != '\n') {
                 while ((ch = getchar()) != '\n' && ch != EOF);
            }
        }
    } else {
        buffer[0] = '\0';
    }
```

```
}
```

Fonksiyon, input alımı sırasında sonraki girişleri etkileyebilecek satır sonu karakterini otomatik temizleyerek oluşabilecek taşmaları önlemek için özel olarak yazılmıştır.

Özellikle *scanf()* kullanımı sırasında oluşabilecek hataları önlemek için kullanılan, C programlamada yaygın bir tekniktir.

Tarih ile Alakalı Fonksiyonlar

```
Tarih stringToTarih (const char* tarihStr) {
    Tarih t = {0,0,0};
    if (tarihStr && strlen(tarihStr) == 10) {
        if (sscanf(tarihStr, "%d.%d.%d", &t.gun, &t.ay, &t.yil) != 3) {
            fprintf(stderr, "HATA: Tarih string formatı geçersiz: %s\n",
tarihStr);
            t.gun = 0; t.ay = 0; t.yil = 0;
        }
    }
    return t;
}
```

Alınan string şeklindeki “gg.aa.yyyy” tarih bilgisini anlamlandırılabilir tarih formatına dönüştürür.

```
int TarihlerArasiGunSayar (Tarih t1_basit, Tarih t2_basit) {
    struct tm tm1 = {0}, tm2 = {0};
    tm1.tm_year = t1_basit.yil - 1900;
    tm1.tm_mon  = t1_basit.ay - 1;
    tm1.tm_mday = t1_basit.gun;
    tm1.tm_hour = 12; tm1.tm_isdst = -1;

    tm2.tm_year = t2_basit.yil - 1900;
    tm2.tm_mon  = t2_basit.ay - 1;
    tm2.tm_mday = t2_basit.gun;
    tm2.tm_hour = 12; tm2.tm_isdst = -1;

    time_t time1_t = mktime(&tm1);
    time_t time2_t = mktime(&tm2);
    if (time1_t == (time_t)(-1) || time2_t == (time_t)(-1))
      return -1;

    return (int)(fabs(difftime(time2_t, time1_t)) / (60.0 * 60.0 * 24.0));
}
```

İki tarih arasındaki gün farkını hesaplar. *Struct tm* kullanarak iki *Tarih* yapısını kullanılan *<time.h>* daki *time_t* tipinde zamana dönüştürür.

difftime() ile bu 2 tarih arasındaki zaman farkı saniye cinsinden bulunur. Matematiksel hesaplamalarla gün cinsinden (yaklaşık olarak) tarih farkına çevirilir.

```
void bugununTarihiniAl (char* tarihBuffer) {
    time_t simdi = time(NULL);
    struct tm *yerelZaman = localtime(&simdi);
    if (yerelZaman != NULL)
      strftime(tarihBuffer, TARIH_UZUNLUK, "%d.%m.%Y", yerelZaman);
    else {
      strcpy(tarihBuffer, "00.00.0000");
      fprintf(stderr, "HATA: Yerel zaman alinamadi.\n");
}
```

Güncel zamanı *<time.h>* yardımı ile string olarak aldığı parametreye yazar.

Koşul Fonksiyonları

Koşul fonksiyonları, her bir liste elemanı üzerinde belirli bir kontrol gerçekleştiren ve sonucu *true (1)* veya *false (0)* olarak döndüren özel fonksiyonlardır.

Bu yapılar, fonksiyon işaretçileri (function pointer) ile birlikte kullanılarak, liste üzerinde yalnızca belirli kriterlere uyan elemanlara işlem yapılmasını sağlar. Örneğin; yalnızca belirli bir yazara ait kayıtları listelemek, puanı belirli bir değerin altında olan öğrencileri bulmak ya da belirli bir ISBN numarasına sahip kitapları filtrelemek gibi işlemler koşul fonksiyonlarıyla yapılabilir.

```
typedef int  (*YazarKosulFonksiyonu)(Yazar*, void*);
```

Bu yapı, *Yazar* tipinde bir nesne ve isteğe bağlı ek veri *(void*)* alır. Fonksiyon, verilen yazarın belirtilen koşulu sağlayıp sağlamadığına göre *1 (doğru)* veya *0 (yanlış)* döndürür.

Bu sayede aynı fonkisyon farklı filtreleme işlemleri için (örneğin yanlızca cezalı öğrencileri listeleme) için kullanılabilir. Kodun okunabirliğini de arttırır ve listeyi dolaşırken gereksiz veriler ile uğraşılmaz (yanlızca alınan parametreler).

Aşağıda kodda koşul fonksiyonlarının nasıl kullanıldığı verilmilştir:

```c
int herZamanDogruKosuluYazar (Yazar* yazar, void* veri) {
    (void)yazar;
    (void)veri;
    return 1;
}


int herZamanDogruKosuluKitap (Kitap* kitap, void* veri) {
    (void)kitap;
    (void)veri;
    return 1;
}


int herZamanDogruKosuluOgrenci (Ogrenci* ogrenci, void* veri) {
    (void)ogrenci;
    (void)veri;
    return 1;
}

int cezaliOgrenciKosulu (Ogrenci* ogrenci, void* veri) {
    (void)veri;
    if (ogrenci) return ogrenci->puan <= 0;

    return 0;
}
```

Yazar İşlemleri için Fonksiyonlar

Yazar* yazarBulByID() Fonksiyonu

Bu fonksiyon ile linked list'te yer alan yazarlardan fonksiyona parametre olarak gönderilmiş ID'ye sahip olan yazar bulunur.

void yazarEkle() Fonksiyonu

Bu fonksiyon, kütüphane otomasyon sistemine yeni bir yazar eklemek için kullanılır. Her yazar için sistem tarafından otomatik olarak bir yazar ID'si oluşturulur ve bu yeni yazar, linked list'e ID değerine göre sıralı biçimde yerleştirilir. Böylece yazarlar her zaman artan ID sırasına göre tutulur.

void yazarSil() Fonksiyonu

Bu fonksiyon, kütüphane otomasyon sisteminde bir yazarı linked list'ten silmek ve kitap-yazar ilişkilerini de buna göre düzenlemek için kullanılır.

void yazarGuncelle() Fonksiyonu

Bu fonksiyon, sistemde kayıtlı olan bir yazarın ad ve soyad bilgilerini güncellemek için kullanılır.

void yazarlariKaydetAnlik() Fonksiyonu

Bu fonksiyon, sistemdeki yazar kayıtlarını anlık olarak dosyaya yazmak için kullanılır.

void yazarlariListele() Fonksiyonu

```
void yazarlariListele (Yazar* yazarListesi) {
    yazarListesiniGezVeIsle(
      yazarListesi,
      yazarBilgisiniYazdir,
      herZamanDogruKosuluYazar,
      NULL,
      "\n--- TUM YAZARLAR LISTESI ---\n",
      "Listelenecek yazar yok."
      );
}
```

Bu fonksiyon, sistemde kayıtlı olan tüm yazarların bilgilerini ekrana sıralı şekilde yazdırmak için kullanılır. Listeleme işlemi doğrudan fonksiyon içerisinde yapılmaz; bunun yerine, bu iş için özel olarak yazılmış esnek ve yeniden kullanılabilir bir gezici yardımcı fonksiyon olan *yazarListesiniGezVeIsle* kullanılır.

Bu sayede:

- Tüm yazarları listelemek
- Sadece Orhan Pamuk'u listelemek
- Sadece 100 puanın altındaki öğrencileri yazdırmak
- Sadece rafta olmayan kitapları yazdırmak

Yanlızca parametre değişimi ile hepsi aynı fonksiyonla yapılabilir.

void yazarListesiniGezVeIsle() Fonksiyonu

```
void yazarListesiniGezVeIsle (Yazar* listeHead, YazarIslemFonksiyonu islemYap,
YazarKosulFonksiyonu kosuluSaglarMi,
                              void* kosulVerisi, const char* baslikMesaji, const
char* bulunamadiMesaji) {

    if (baslikMesaji)
        printf("%s", baslikMesaji);
    if (!listeHead) {
         printf("%s\n", bulunamadiMesaji ? bulunamadiMesaji : "Liste bos veya
listelenecek oge bulunamadi.");
    }

    Yazar* temp = listeHead;
    int enAzBirOgeIsledim = 0;

    while (temp != NULL) {
        if (kosuluSaglarMi == NULL || kosuluSaglarMi(temp, kosulVerisi)) {
            if (islemYap != NULL) {
                islemYap(temp);
                enAzBirOgeIsledim = 1;
            }
        }
        temp = temp->next;
    }

//
// Liste gezilir ve tek tek her yazar için şu işlemler yapılır:
// kosuluSaglarMi tanımlı DEĞİLSE : NULL (tüm ögeler işlenir.)
// kosuluSaglarMİ tanımlıysa : Yanlızca koşulu sağlayan ögeler işlenir.
// Koşul sağlandıysa işlem yapabilmek için islemYap() çağırılır.
// Döngünün en az 1 kez tamamlandığını göstermek için flag
// enAzBirOgeIsledim = 1 yapılır.

    if (!enAzBirOgeIsledim && listeHead) {
      //
      // Liste boş değil ama koşula uyan olmadıysa;
      //
        printf("%s\n", bulunamadiMesaji ? bulunamadiMesaji : "Belirtilen kosula
uygun oge bulunamadi.");
    }
    if (baslikMesaji)
        printf("----------------------------------------\n");
```

```
}
```

Bu fonksiyon, yazarlar listesini gezmek, belirli bir koşulu sağlayan öğeler üzerinde belirli bir işlemi gerçekleştirmek ve gerektiğinde kullanıcıya uygun mesajlar vermek amacıyla yazılmış **genel amaçlı (filtreleme, yazarı silme vb.) bir işlem fonksiyonudur.**

listeHead	İşlenecek yazar bağlı listesinin başlangıcıdır.
islemYap	Uygulanacak işlem fonksiyonudur (örneğin: yazdır, sil).
kosuluSaglarMi	Hangi öğelerin işleneceğini belirleyen koşul fonksiyonudur.
kosulVerisi	Koşul fonksiyonuna dışarıdan gönderilecek ek veridir (örneğin aranacak ID).
baslikMesaji	İşlem başlamadan önce ekrana yazdırılan başlık mesajıdır.
bulunamadiMesaji	İşlenecek öğe yoksa ekrana yazdırılan uyarı mesajıdır.

void yazarListesiniYazdir() Fonksiyonu

Basit bir verilen yazarı printleyen kod.

void yazarlariYukle() Fonksiyonu

Bu fonksiyon, `"yazarlar.csv"` dosyasından kayıtlı yazar bilgilerini belleğe yükler ve linked list hâlinde yapılandırır. Böylece program başlatıldığında veya veri yeniden yüklenmek istendiğinde, sistemde daha önceden kayıtlı olan yazarlar eksiksiz şekilde yeniden oluşturulur.

Öğrenci İşlemleri için Fonksiyonlar

Ogrenci* ogrenciBulByID() Fonksiyonu

Bu fonksiyon, sistemde kayıtlı olan öğrenciler arasından parametre olarak verilen öğrenci numarasına sahip olan öğrenciyi linked list yapısı üzerinde arayıp bulmak için kullanılır.

void ogrenciEkle() Fonksiyonu

Bu fonksiyon, alınan bilgilerle sistemde linked list yapısına yeni bir öğrenci eklemek için kullanılır.

void ogrenciSil() Fonksiyonu

Bu fonksiyon, parametre olarak verilen öğrenci numarasına sahip öğrenciyi linked list yapısından silmek amacıyla kullanılır.

void ogrenciGuncelle() Fonksiyonu

Bu fonksiyon, linked list yapısında var olan bir öğrencinin bilgilerini öğrenci numarası aracılığıyla bularak isim, soy isim ve puanında değişiklik yapmak için kullanılır.

void ogrencileriKaydetAnlik() Fonkisyonu

Bu fonksiyon, sistemde bağlı liste hâlinde tutulmakta olan tüm öğrenci kayıtlarını anlık olarak diske (CSV dosyasına) yazmak için kullanılır.

void ogrencileriListele() Fonksiyonu

```
void ogrencileriListele (Ogrenci* ogrList) {
    ogrListniGezVeIsle(ogrList,
                       ogrenciBilgisiniYazdir,
                       herZamanDogruKosuluOgrenci,
                       NULL,
                       "\n--- TUM OGRENCI LISTESI ---\n",
                       "Listelenecek ogrenci yok.",
                       "Ogrenci No  Ad Soyad\t\t
                        Puan\n----------- -------------------- ----\n");
}
```

yazarlariListele() fonksiyonunun mantığıyla benzer çalışır. Listeleme işlemi doğrudan bu fonksiyonun içerisinde yapılmaz. Onun yerine yeniden kullanılabilir ve belirli parametrelerle ayrıştırma yapılabilen *ogrListiniGezVeIsle()* fonksiyonu yardımıyla listeleme gerçekleştirilir.

void ogrListiniGezVeIsle() Fonksiyonu

```
void ogrListniGezVeIsle (Ogrenci* listeHead, OgrenciIslemFonksiyonu islemYap,
OgrenciKosulFonksiyonu kosuluSaglarMi, void* kosulVerisi, const char*
baslikMesaji, const char* bulunamadiMesaji, const char* tabloBasligi) {
    if (baslikMesaji)
        printf("%s", baslikMesaji);

    if (!listeHead) {
```

```
        printf("%s\n", bulunamadiMesaji ? bulunamadiMesaji : "Liste bos veya
listelenecek oge bulunamadi.");
    }

    Ogrenci* temp = listeHead;
    int enAzBirOgeIsledim = 0;
    int tabloBasligiYazdirildiMi = 0;

    while (temp != NULL) {
        if (kosuluSaglarMi == NULL || kosuluSaglarMi(temp, kosulVerisi)) {
            if (tabloBasligi && !tabloBasligiYazdirildiMi) {
                printf("%s", tabloBasligi);
                tabloBasligiYazdirildiMi = 1;
            }
            if (islemYap != NULL) {
                islemYap(temp);
                enAzBirOgeIsledim = 1;
            }
        }
        temp = temp->next;
    }

    if (!enAzBirOgeIsledim && listeHead) {
        printf("%s\n", bulunamadiMesaji ? bulunamadiMesaji : "Belirtilen kosula
uygun oge bulunamadi.");
    }
    if (baslikMesaji)
        printf("-------------------------------------------------\n");
}
```

Bu fonksiyon, bağlı liste hâlinde tutulan öğrenci verilerini gezmek, belirli koşulları sağlayan öğelere işlem uygulamak ve bu işlemler sırasında kullanıcıya okunabilir biçimde bilgi sunmak amacıyla oluşturulmuştur. Fonksiyon, koşul ve işlem soyutlamasını destekleyen esnek yapısıyla farklı listeleme senaryolarına kolayca uyarlanabilir.

void cezaliOgrencileriListele() Fonksiyonu

```
void cezaliOgrencileriListele (Ogrenci* ogrList) {
    ogrListniGezVeIsle(ogrList, ogrenciBilgisiniYazdir, cezaliOgrenciKosulu,
NULL,
                       "\n--- CEZALI OGRENCILER (Puan <= 0) ---\n", "Cezali
ogrenci bulunmamaktadir.",
                       "Ogrenci No  Ad Soyad              Puan\n----------- -----
---------------- ----\n");
```

```
}
```

Yine *ogrListiniGezVeIsle()* fonksiyonu kullanılarak, ve istenen parametrelerle çağırılarak modüler bir şekilde istenen özellikteki öğrencilerin listelenmesi sağlanmıştır.

void ogrenciBilgisiniYazdir() Fonksiyonu

Basit bir şekilde görüntülenmek istenen öğrencinin bilgilerini printleyen fonksiyon.

void ogrencileriYukle() Fonksiyonu

Bu fonksiyon, “Ogrenciler.csv” dosyasından kayıtlı öğrencilerin verilerini okuyarak bir linked list yapısına bu verileri geçirir.Böylece veriler belleğe alınmış olur ve veriler üzerinde dinamik bir şekilde işlemler gerçekleştirilebilir.

Kitap İşlemleri için Fonksiyonlar

Kitap* kitapBulByIBSN() Fonksiyonu

Bu fonksiyon, linked list içerisinde gezinerek parametre olarak verilmiş ISBN’e sahip kitabı bulur.

KitapKopya* kitapKopyasiBulByEtiket() Fonksiyonu

Bu fonksiyon, bir kitaba ait kopyaların bağlı listesi içinde, verilen etiket numarasına sahip olan kopyayı arar. Eğer aranan etiket numarası ile eşleşen bir KitapKopya düğümü bulunursa, o düğümün adresi döndürülür. Aksi takdirde *NULL* döndürülerek bulunamadığı belirtilir.

void kitapEkle() Fonksiyonu

```
void kitapEkle(Kitap** kitapListHead, const char* ad, const char* isbn, int adet)
{
    int i;
    if (kitapBulByISBN(*kitapListHead, isbn) != NULL) {
        printf("HATA: %s ISBN numarali kitap zaten mevcut.\n", isbn);
        return;
    }
//
// Aynı ISBN numarasına sahip kitap listede bulunuyor mu diye kontrol
//
    if (strlen(isbn) != 13) {
        printf("HATA: ISBN 13 haneli olmalidir.\n");
```

```
        return;
    }
    Kitap* yeniKitap = (Kitap*)malloc(sizeof(Kitap));
    if (!yeniKitap) { printf("HATA: Kitap icin bellek ayrilamadi!\n");
        return; }

    strncpy(yeniKitap->ad, ad, MAX_KITAP_AD-1);
    yeniKitap->ad[MAX_KITAP_AD-1] = '\0';

    strncpy(yeniKitap->isbn, isbn, ISBN_UZUNLUK-1);
    yeniKitap->isbn[ISBN_UZUNLUK-1] = '\0';

    yeniKitap->adet = 0;
    yeniKitap->kopyalar = NULL;
    yeniKitap->next = NULL;

    for (i = 0; i < adet; i++) {
        KitapKopya* yeniKopya = (KitapKopya*)malloc(sizeof(KitapKopya));
        if (!yeniKopya) {
            printf("HATA: Kitap kopyasi icin bellek ayrilamadi! %d. kopya
eklenemedi.\n", i + 1);
        } else {
            sprintf(yeniKopya->etiketNumarasi, "%s_%d", isbn, yeniKitap->adet +
1);
            strncpy(yeniKopya->durum, "RAFTA", DURUM_UZUNLUK-1);
            yeniKopya->durum[DURUM_UZUNLUK-1]='\0';

            yeniKopya->next = yeniKitap->kopyalar;
            yeniKitap->kopyalar = yeniKopya;
            yeniKitap->adet++;
        }
    }

    if (*kitapListHead == NULL || strcmp((*kitapListHead)->ad, yeniKitap->ad) > 0
||
        (strcmp((*kitapListHead)->ad, yeniKitap->ad) == 0 &&
strcmp((*kitapListHead)->isbn, yeniKitap->isbn) > 0)) {
        yeniKitap->next = *kitapListHead;
        *kitapListHead = yeniKitap;
    } else {
        Kitap* temp = *kitapListHead;
        int devam = 1;
        while (temp->next != NULL && devam) {
```

```
            if (strcmp(temp->next->ad, yeniKitap->ad) < 0 ||
                (strcmp(temp->next->ad, yeniKitap->ad) == 0 && strcmp(temp->next-
>isbn, yeniKitap->isbn) < 0)) {
                temp = temp->next;
            } else {
                devam = 0;
            }
        }
        yeniKitap->next = temp->next;
        temp->next = yeniKitap;
    }
    printf("'%s' (ISBN: %s) %d kopya ile eklendi.\n", ad, isbn, yeniKitap->adet);
    kitaplariKaydetAnlik(*kitapListHead);
}
```

Bu fonksiyon, kütüphane sistemine yeni bir kitap eklemek amacıyla kullanılır. Verilen kitap adı (*ad*), ISBN numarası (*isbn*) ve kopya sayısı (*adet*) bilgileri ile bir kitap oluşturulur. Ardından bu kitap, alfabetik sıraya göre bağlı listeye eklenir. Ayrıca her kitap için belirtilen sayıda kopya (*KitapKopya*) oluşturulur ve ilk durumu *"RAFTA"* olarak ayarlanır.

void kitapSil() Fonksiyonu

```
void kitapSil(Kitap** kitapListHead, const char* isbn) {
    Kitap* kontrolKitap = kitapBulByISBN(*kitapListHead, isbn);
    if (!kontrolKitap) {
        printf("%s ISBN'li kitap bulunamadi.\n", isbn);
        return;
    }

    KitapKopya* k = kontrolKitap->kopyalar;
    int oduncAlinmisVar = 0;
    int devamKopyaKontrol = 1;

    while(k != NULL && devamKopyaKontrol){
        if(strcmp(k->durum, "RAFTA") != 0) {
            oduncAlinmisVar = 1;
            devamKopyaKontrol = 0;
        }
        if (devamKopyaKontrol) k = k->next;
    }

    if(oduncAlinmisVar){
```

```
        printf("HATA: %s ISBN'li kitabin odunc alinmis kopyalari var.
Silinemez.\n", isbn);
        return;
    }

    Kitap* silinecek = NULL, *onceki = NULL;
    int silindi = 0;

    if (strcmp((*kitapListHead)->isbn, isbn) == 0) {
        silinecek = *kitapListHead;
        *kitapListHead = (*kitapListHead)->next;
        silindi = 1;
    } else {
        onceki = *kitapListHead;
        int devamSil = 1;
        while (onceki->next != NULL && devamSil) {
            if (strcmp(onceki->next->isbn, isbn) != 0) {
                 onceki = onceki->next;
            } else {
                devamSil = 0;
            }
        }
        if (!devamSil && onceki->next !=NULL) {
            silinecek = onceki->next;
            onceki->next = silinecek->next;
            silindi = 1;
        }
    }

    if (silindi && silinecek) {
        printf("'%s' (ISBN: %s) adli kitap ve tum kopyalari silindi.\n",
silinecek->ad, silinecek->isbn);
        KitapKopya* kopya = silinecek->kopyalar, *tempKopya;
        while (kopya != NULL) {
            tempKopya = kopya; kopya = kopya->next;
            free(tempKopya);
        }
        free(silinecek);
        kitaplariKaydetAnlik(*kitapListHead);
    } else if (!silindi) {
        printf("%s ISBN'li kitap silinirken bulunamadi.\n", isbn);
    }
}
```

Bu fonksiyon, sistemde kayıtlı olan kitaplardan belirli bir ISBN numarasına sahip olanı liste üzerinden silmek amacıyla yazılmıştır.

Ancak kitap, ancak **tüm kopyaları raftaysa (yani ödünçte olan kopyası yoksa)** silinebilir.

Kitap silindiğinde, ona ait tüm kopyalar da bellekten temizlenir ve ardından güncel kitap listesi *.csv* dosyasına kaydedilir.

void kitapGuncelle() Fonksiyonu

Bu fonksiyon, sistemdeki bir kitabın ISBN numarasına göre adını güncellemek için kullanılır.

void kitaplariKaydetAnlik() Fonksiyonu

Bu fonksiyon, sistemdeki tüm kitapları ve onların kopyalarını *.csv* formatında bir dosyaya anlık olarak yazmak için kullanılır.

void kitapYazarEslestir() Fonksiyonu

```
void kitapYazarEslestir(KitapYazarVeriYoneticisi* kitapYazarDB, Kitap* kitapList,
Yazar* yazarListesi, const char* isbn, int yazarID) {
    int i, zatenVar = 0;

    if (!kitapBulByISBN(kitapList, isbn)) {
        printf("HATA: %s ISBN'li kitap bulunamadi.\n", isbn);
        return;
    }
    if (!yazarBulByID(yazarListesi, yazarID)) {
        printf("HATA: %d ID'li yazar bulunamadi.\n", yazarID);
        return;
    }

//
// Kitap ve yazar var mı diye kontrol
//

    for(i = 0; i < kitapYazarDB->sayi && !zatenVar; i++){
        if(strcmp(kitapYazarDB->iliskiler[i].isbn, isbn) == 0 && kitapYazarDB-
>iliskiler[i].yazarID == yazarID){
            zatenVar = 1;
        }
    }
```

```
//
// Zaten kitap ve yazar eşleştirilmiş mi diye ekstra kontrol
//

    if(zatenVar) {
        printf("Bu kitap-yazar eslesmesi zaten mevcut.\n");
        return;
    }

    KitapYazarIliski* yeniIliskiler = realloc(kitapYazarDB->iliskiler,
(kitapYazarDB->sayi + 1) * sizeof(KitapYazarIliski));
    if (!yeniIliskiler) { printf("HATA: Kitap-Yazar iliskisi icin bellek
ayrilamadi!\n");
        return;
    }
    kitapYazarDB->iliskiler = yeniIliskiler;
    strncpy(kitapYazarDB->iliskiler[kitapYazarDB->sayi].isbn, isbn, ISBN_UZUNLUK-
1);

    kitapYazarDB->iliskiler[kitapYazarDB->sayi].isbn[ISBN_UZUNLUK-1] = '\0';
    kitapYazarDB->iliskiler[kitapYazarDB->sayi].yazarID = yazarID;

    (kitapYazarDB->sayi)++;
    printf("%s ISBN'li kitap ile %d ID'li yazar eslestirildi.\n", isbn, yazarID);
    kitapYazarIliskileriniKaydetAnlik(kitapYazarDB);
}
```

Bu fonksiyon, bir kitap ile bir yazarı eşleştirmek (ilişkilendirmek) amacıyla kullanılır. ISBN numarası ile tanımlanan kitap, *yazarID* ile tanımlanan yazarla eşleştirilir.

Eşleştirme verisi, *KitapYazarVeriYoneticisi* adlı bir dinamik dizi yapısında tutulur. Bu yapı sayesinde, her kitap birden fazla yazarla, her yazar da birden fazla kitapla ilişkilendirilebilir.

KitapYazarIlıski struct'ı da *KitapYazarVeriYoneticisi* içerisinde bulunur ve kapsüllenmiş bir şekilde ilişkilendirilmiş kitabın ISBN'i ve yazarın ID'si tutulur.

void kitabinYazariniGuncelle() Fonksiyonu

Bu fonksiyon, bir kitabın daha önceden eşleştirilmiş yazar bilgisini güncellemek amacıyla kullanılır.

void kitapYazarIliskileriniKaydetAnlik() Fonksiyonu

Bu fonksiyon, kitap- yazar ilişkilerini tutan linked list yapısını ilgili *.csv* dosyasına geçirir.

void kitapOduncAl() Fonksiyonu

Bu fonksiyon, bir öğrencinin sistemden kitap ödünç almasını sağlar. Belirli bir ISBN numarasına sahip kitabın rafta bulunan bir kopyası, belirli bir öğrenciye ait kimlik numarası ile ilişkilendirilerek sistemde "ödünç" olarak kaydedilir.

void oduncKayitlariniKaydetAnlik() Fonksiyonu

Bu fonksiyon, gerçekleştirilmiş ödünç alma verilerini *KitapOdunc.csv* dosyasına yazar.

void kitapTeslimEt() Fonksiyonu

Bu fonksiyon, bir öğrencinin ödünç aldığı kitabı kütüphaneye **teslim etmesini** sağlayan işlemleri yürütür.

void kitaplariListele() Fonksiyonu

Bu fonksiyon, tüm kitapları ve onlara ait kopyaları listeleyen yardımcı bir işlemdir.

void raftakiKitaplariListele() Fonksiyonu

Bu fonksiyon, sadece rafta bulunan kitapları (yani henüz ödünç verilmemiş olanları) ekrana yazdırır. Kitap listesi içinde dolaşır ve her kitap için tüm kopyaları kontrol eder.

void kitapTeslimEtmemisOgrencileriListele() Fonksiyonu

Bu kitapTeslimEtmemisOgrencileriListele fonksiyonu, kütüphane sistemindeki kitapları teslim etmemiş öğrencileri listeler.

void zamanindaTeslimEdilmeyenKitaplariListele() Fonksiyonu

Bu fonksiyon, 15 günden fazla süredir ödünçte olan kitapları listelemek için kullanılır. Yani teslim süresi geçmiş kitapları ve bu kitapların kimde olduğunu ekrana yazdırır.

void kitapListiniGezVeIsle() Fonksiyonu

```
void kitapListniGezVeIsle (Kitap* listeHead, KitapIslemFonksiyonu islemYap,
KitapKosulFonksiyonu kosuluSaglarMi,
                           void* kosulVerisi, const char* baslikMesaji, const
char* bulunamadiMesaji) {
```

```
    if (baslikMesaji)
        printf("%s", baslikMesaji);

        if (!listeHead) {
         printf("%s\n", bulunamadiMesaji ? bulunamadiMesaji : "Liste bos veya
listelenecek oge bulunamadi.");
    }

    Kitap* temp = listeHead;
    int enAzBirOgeIsledim = 0;
    while (temp != NULL) {
        if (kosuluSaglarMi == NULL || kosuluSaglarMi(temp, kosulVerisi)) {
            if (islemYap != NULL) {
                islemYap(temp);
                enAzBirOgeIsledim = 1;
            }
        }
        temp = temp->next;
    }
     if (!enAzBirOgeIsledim && listeHead) {
         printf("%s\n", bulunamadiMesaji ? bulunamadiMesaji : "Belirtilen kosula
uygun oge bulunamadi.");
    }
    if (baslikMesaji) printf("-----------------------------------\n");
}
```

Bu *kitapListniGezVeIsle* fonksiyonu, bir kitaplar listesini dolaşır ve istenen koşulları sağlayan kitaplara belirli bir işlemi uygular.

void kitaplariYukle() Fonksiyonu

Bu kitaplariYukle fonksiyonu, bir CSV dosyasından kitap ve kitap kopyası bilgilerini okuyup linked list yapısına yükleyen bir işlemi gerçekleştirir.

Diğer fonksiyonlar rapor çok uzadığı için listelenmemiştir.

Örnekler

Örnekler verilen örnek dosyası kullanılarak hazırlanmıştır.

Öğrenci Görüntüleme

```
C:\git-code\code\c\benim-y-p>a
Kutuphane Otomasyonu Baslatiliyor...
Yazarlar.csv dosyasindan yazarlar yuklendi. Atanan Son Yazar ID: 16
Ogrenciler.csv dosyasindan ogrenciler yuklendi.
Kitaplar.csv dosyasindan kitaplar ve kopyalari yuklendi.
KitapYazar.csv dosyasindan kitap-yazar iliskileri yuklendi.
KitapOdunc.csv dosyasindan odunc kayitlari yuklendi.

Veriler yuklendi. Ana menuye geciliyor.

ANA MENU
1- Ogrenci Islemleri
2- Kitap Islemleri
3- Yazar Islemleri
4- Cikis
Seciminiz: 1

--- OGRENCI ISLEMLERI ---
1- Ogrenci Ekle
2- Ogrenci Sil
3- Ogrenci Guncelle
4- Ogrenci Bilgisi Goruntule (ID)
6- Tum Ogrencileri Listele
7- Kitap Teslim Etmemis Ogrencileri Listele
8- Cezali Ogrencileri Listele
9- Kitap Odunc Al
10- Kitap Teslim Et
99- Ana Menuye Don
Seciminiz: 4
Goruntulenecek Ogrenci No: 10202512
Ogrenci Bilgileri:
  No: 10202512
  Ad: Abdullah
  Soyad: Yilmaz
  Puan: 100
Kitap Hareketleri:
  Bu ogrenciye ait kitap hareketi bulunamadi.
```

Kitap Bilgisi Görüntüleme

```
ANA MENU
1- Ogrenci Islemleri
2- Kitap Islemleri
3- Yazar Islemleri
4- Cikis
Seciminiz: 2

--- KITAP ISLEMLERI ---
1- Kitap Ekle
2- Kitap Sil
3- Kitap Guncelle (Adini)
4- Kitap Bilgisi Goruntule (Adina Gore)
5- Raftaki Kitaplari Listele
6- Zamaninda Teslim Edilmeyen Kitaplari Listele
7- Kitap-Yazar Eslestir
8- Kitabin Yazarini Guncelle
9- Tum Kitaplari Listele
99- Ana Menuye Don
Seciminiz: 4
Goruntulenecek Kitap Adi (veya bir kismi): Her Sey
Kitap Adi: Her Seyin Teorisi, ISBN: 3219022972921, Toplam Kopya: 3
  Kopyalar:
    Etiket: 3219022972921_1, Durum: RAFTA
    Etiket: 3219022972921_2, Durum: RAFTA
    Etiket: 3219022972921_3, Durum: RAFTA
```

Yazar Bilgisi Görüntüleme

```
ANA MENU
1- Ogrenci Islemleri
2- Kitap Islemleri
3- Yazar Islemleri
4- Cikis
Seciminiz: 3

--- YAZAR ISLEMLERI ---
1- Yazar Ekle
2- Yazar Sil
3- Yazar Guncelle
4- Yazar Bilgisi Goruntule (Adina Gore)
5- Tum Yazarlari Listele
99- Ana Menuye Don
Seciminiz: 4
Goruntulenecek Yazar Adi (veya bir kismi): Carl
Bulunan Yazarlar ve Kitaplari:
Yazar ID: 16, Ad: Carl, Soyad: Sagan
  Kitaplari:
    - Soluk Mavi Nokta (ISBN: 9361229347876)
    - Mesaj (ISBN: 2598714271168)
```

Kitap Ödünç Alma (Başarılı)

```
ANA MENU
1- Ogrenci Islemleri
2- Kitap Islemleri
3- Yazar Islemleri
4- Cikis
Seciminiz: 1

--- OGRENCI ISLEMLERI ---
1- Ogrenci Ekle
2- Ogrenci Sil
3- Ogrenci Guncelle
4- Ogrenci Bilgisi Goruntule (ID)
6- Tum Ogrencileri Listele
7- Kitap Teslim Etmemis Ogrencileri Listele
8- Cezali Ogrencileri Listele
9- Kitap Odunc Al
10- Kitap Teslim Et
99- Ana Menuye Don
Seciminiz: 9
Odunc Alacak Ogrenci No: 17011010
Odunc Alinacak Kitabin ISBN'i: 4054110263807
'4054110263807_1' etiketli kitap Ross (17011010) tarafindan 27.05.2025 tarih
inde odunc alindi.
```

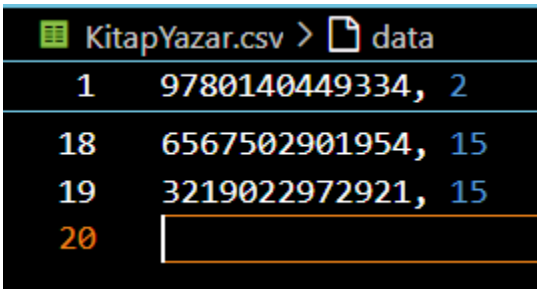
Kitap Ödünç Alma (Başarısız)

```
ANA MENU
1- Ogrenci Islemleri
2- Kitap Islemleri
3- Yazar Islemleri
4- Cikis
Seciminiz: 1

--- OGRENCI ISLEMLERI ---
1- Ogrenci Ekle
2- Ogrenci Sil
3- Ogrenci Guncelle
4- Ogrenci Bilgisi Goruntule (ID)
6- Tum Ogrencileri Listele
7- Kitap Teslim Etmemis Ogrencileri Listele
8- Cezali Ogrencileri Listele
9- Kitap Odunc Al
10- Kitap Teslim Et
99- Ana Menuye Don
Seciminiz: 9
Odunc Alacak Ogrenci No: 17011001
Odunc Alinacak Kitabin ISBN'i: 3214569871012
HATA: Ogrencinin puani (-10) yetersiz. Kitap odunc alamaz.
```

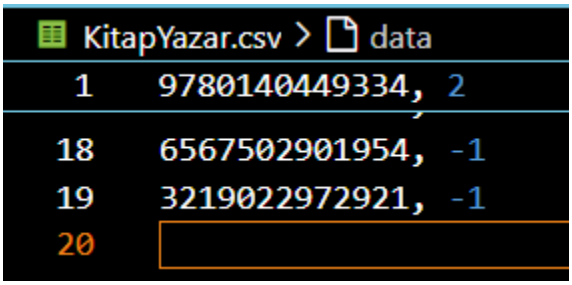
Yazarın Kitapla Olan İlişkisini Koparma

```
ANA MENU
1- Ogrenci Islemleri
2- Kitap Islemleri
3- Yazar Islemleri
4- Cikis
Seciminiz: 3

--- YAZAR ISLEMLERI ---
1- Yazar Ekle
2- Yazar Sil
3- Yazar Guncelle
4- Yazar Bilgisi Goruntule (Adina Gore)
5- Tum Yazarlari Listele
99- Ana Menuye Don
Seciminiz: 2
Silinecek Yazar ID: 15
'Stephen Hawking' (ID: 15) adli yazar silindi.
```

ÖNCESİ:



SONRASI:



video : https://youtu.be/InM8cRrEN_0