LIBRARY MANAGEMENT SYSTEM

Ahmet Deniz Daştan

GITHUB : saddeg21 (Ahmet Deniz Daştan) (github.com)

İçindekiler

PROJE TANIMI

PROJE KONUSU

Proje, yapısal programlama yaklaşımını kullanarak bir kütüphane yönetim sistemi kurmayı hedeflemektedir. Bu sistemde dinamik bellek kullanımı, struct yapılarının kullanımı, C dilinde reading/writing işlemlerinin yapılması, C ile tasarlanmış bir konsol uygulaması için GUI (Graphical User Interface) tasarlanması ile ilgili konularda kodların yazılması ile oluşturulmuştur.

PROJENİN TEMEL FONKSİYONLARI

ÖĞRENCİ İŞLEMLERİ:

1- **Öğrenci Ekle, Sil, Güncelle:** Öğrenciye ait tüm bilgilerin kullanıcıdan alınıp, dosyalar ve linkli listeler üzerinde ***Ekle/Sil/Güncelle*** işlemlerinin ayrı ayrı gerçekleştirilmesi gerekmektedir.
2- **Öğrenci Bilgisi Görüntüleme:** ID bilgisi veya Ad-Soyad bilgisi alınan öğrencinin kişisel bilgileri (Ad, Soyad, ID, Puan vb) ve tüm kitap hareketleri listelenmelidir.
3- **Kitap Teslim Etmemiş Öğrencileri Listele:**
4- **Cezalı Öğrencileri Listele**
5- **Tüm Öğrencileri Listele**
6- **Kitap Ödünç Al/Teslim Et**

KİTAP İŞLEMLERİ:

7- **Kitap Ekle, Sil, Güncelle:** Kitaplara ait tüm bilgilerin kullanıcıdan alınıp, dosyalar ve linkli listeler üzerinde ***Ekle/Sil/Güncelle*** işlemlerinin ayrı ayrı gerçekleştirilmesi gerekmektedir.
8- **Kitap Bilgisi Görüntüleme:** Kullanıcıdan alınan Kitap Adı bilgisine göre her bir kitabın ve bu kitabın örnek kopyalarına ***ait tüm bilgilerin listelenmesi*** işlemleri yapılmalıdır.
9- **Raftaki Kitapları Listele:**
10- **Zamanında Teslim Edilmeyen Kitapları Listele:**
11- **Kitap-Yazar Eşleştir :** İlgili dosya ve struct dizisi üzerinde güncelleme yapılmalıdır
12- **Kitabın Yazarını Güncelle:** İlgili dosya ve struct dizisi üzerinde güncelleme yapılmalıdır

YAZAR İŞLEMLERİ:

13- **Yazar Ekle, Sil, Güncelle:** Yazarlara ait tüm bilgilerin kullanıcıdan alınıp, dosyalar ve linkli listeler üzerinde ***Ekle/Sil/Güncelle*** işlemlerinin ayrı ayrı gerçekleştirilmesi gerekmektedir.
14- **Yazar Bilgisi Görüntüleme:** Kullanıcıdan alınan Yazar Adı bilgisine göre her bir yazarın bilgilerinin ve bu yazara ait kitaplara ***ait tüm bilgilerin listelenmesi*** işlemleri yapılmalıdır.

PROJEDE KULLANILMASI GEREKEN STRUCT YAPILARI

```
typedef struct Ogrenci { char ogrID[9]; char ad[30]; char soyad[30]; int puan; struct Ogrenci *next; struct Ogrenci *prev;} Ogrenci;
typedef struct Yazar { int yazarID; char yazarAd[30]; char yazarSoyad[30]; struct Yazar *next;}Yazar;
typedef struct KitapOrnek { char EtiketNo[20]; char Durum[8]; struct KitapOrnek *next;} KitapOrnek;
typedef struct Kitap {char kitapAdi[30]; char ISBN[14]; int adet; struct Kitap *next; struct KitapOrnek *head;} Kitap;
typedef struct Tarih{ unsigned int gun:5; unsigned int ay:4; unsigned int yil:12;} Tarih;
typedef struct KitapOdunc{ char EtiketNo[20]; char ogrID[9]; int islemTipi:2; struct Tarih islemTarihi;} KitapOdunc;
typedef struct KitapYazar{ char ISBN[14]; int YazarID;} KitapYazar;
```

PROJE PROGRAMLAMA KURALLARI

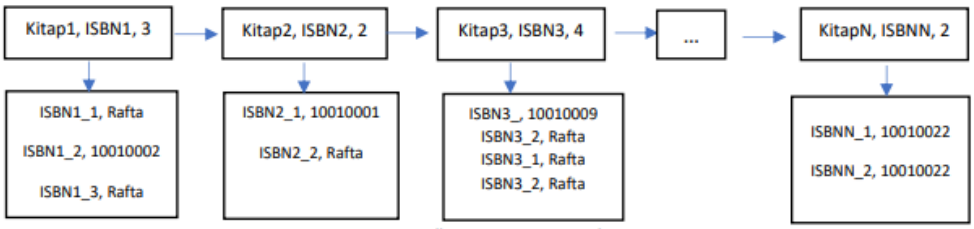
- Yukarıdaki maddelerde verilen bir menü ve her bir işlem için gerekli fonksiyonları tasarlayınız. Benzer mantıkla çalışan fonksiyonlarda tekrardan kaçınmak için fonksiyon pointer kullanınız
- Yapılması gereken tüm ekle, sil ve güncelleme işlemler için fonksiyon tanımlamanız gerekmektedir.
- Ekle, Sil, Güncelle gibi veri üzerinde değişiklik yapılan işlemleri ilgili veri yapısında yaptıktan sonra dosyaya mutlaka kaydediniz.
- Programlarınızda static ve global değişken kullanımı yasaktır.
- Her türlü bellek tahsisi, dinamik bellek yönetimi fonksiyonları ile yapılmalıdır.
- Yukarıda detayları verilen her bir dosyadaki nesne için uygun structure tanımlamalarını yapınız.
- Oluşturulacak dosyalar CSV dosyası formatında olmalıdır.

PROJEDE İSTENEN İŞLEVSEL GEREKSİNİMLER

- **Yazar Ekleme/Silme/Düzenleme/Listeleme İşlemleri:** Her bir yazarın adı, soyadı, her yeni yazar kaydında otomatik artarak (1, 2, .., N) verilecek yazarID bilgisi TEK YÖNLÜ LINKED LISTEDE yazarID değerine göre sıralı bir şekilde saklanmalı yazarlar.csv isimde dosyada tutulmalıdır. Yazarlar için kullanıcıdan alınıp dosyada saklanan bu bilgiler istenirse silinebilmeli veya düzeltilebilmelidir ve son hali aynı dosyada güncel olarak bulunmalıdır.

- **Öğrenci Ekleme/Silme/Düzenleme/Listeleme İşlemleri:** Her bir öğrencinin adı, soyadı, başlangıçta 100 olan kütüphane puanı ve her bir öğrenci için benzersiz olan 8 haneli öğrenci numarası olmalıdır. Öğrencilere ait bilgiler dosyada ve uygulama içerisinde tanımlanacak struct yapısında ÇİFT YÖNLÜ LINKED LIST ve Ogrenciler.csv isimli dosyada saklanmalıdır. Saklanan bu bilgiler istenirse silinebilmeli veya düzeltilebilmelidir ve son hali aynı dosyada güncel olarak bulunmalıdır

- **Kitap Ekleme/Silme/Düzenleme/Listeleme İşlemleri:** Her bir kitabın adı, 13 Haneli ISBN numarası ve adet bilgisi olmalıdır. Bir kitaptan birden fazla sayıda örnek kütüphanede olabileceği için, kayıt oluşturma sırasında her bir örnek kitaba ayrıca benzersiz olan etiket numarası ISBN numarasına ISBN_1, ISBN_2, ISBN_N şeklinde otomatik eklenecek sayı ile verilmelidir. Kitaplar önce isimlerine sonra da ISBN numaralarına göre TEK YÖNLÜ LINKED LIST (struct Kitap) ile aşağıdaki gibi saklanmalıdır. Her kitabın örnek sayısı kadar kitabın bilgisi için ayıca TEK YÖNLÜ LINKED LIST (struct KitapOrnek) oluşturulmalı ve bu listelerde kitabın ETİKET numarası ile kitabın ödünç alınma durumu saklanmalıdır. Bu bağlamda, kitap ödünç alınmış ise hangi öğrenci tarafından ödünç alındı ise o öğrencinin ID numarası yer almalıdır. Eğer kitap ödünç alınmamışsa, bu alanda RAFTA bilgisi yazmalıdır. Her bir KİTAP ve bunların örnekleri arasındaki ilişki Şekil1'de gösterildiği gibi TEK YÖNLÜ LINKED LIST'ler kullanılarak modellenmeli, ayrıca bu veriler uyumlu bir şekilde CSV türündeki dosyalarda saklanmalıdır. CSV dosyası; her bir örneğe ait verinin virgül (,) işareti ile ayrıldığı metin dosyasıdır.



Şekil 1 Kitap ve Örnekleri Arasındaki İlişki

- **Kitap – Yazar İlişkilendirme İşlemleri:** Bir kitabın birden çok yazar tarafından yazılma, bir yazarın da çok sayıda kitap yazabilme durumu dikkate alınarak, KitapISBN - YazarID eşleştirmesi yapılmalıdır. Bu eleştirme bilgileri KitapYazar.csv dosyasında Şekil-2'de gösterildiği gibi saklanmalıdır. Bu dosya üzerinde yapılacak okuma işlemlerinde kaydedilecek veriler için bir struct tasarlanmalıdır. Bu struct yoluyla dosyadaki satır sayısı (N) kadar yer açıldıktan sonra, N elemanlı dinamik STRUCT DİZİSİNDE veriler program açık kaldığı sürece saklanmalıdır. Şekil-2'de bulunan örneğe göre; 1234567891011 kodlu kitap 1, 2 ve 3 ID'ye sahip yazarlar tarafından yazılmış olup, 1234567891012 kodlu kitabı ise sadece 1 ID'ye sahip yazar yazmıştır. Örnekten de anlaşıldığı gibi bir yazar çok sayıda kitap yazabilirken, bir kitap çok sayıda yazar tarafından yazılabilir.
  NOT1: Madde-1 ile girişi yapılan yazar bilgisi sistemden silindiğinde aşağıdaki dosyada güncelleme yapılmalıdır. İlgili yazar artık olmayacağı için bu alanda -1 yazmalıdır. (Örnek aşağıdadır)
  NOT2: Kitap – Yazar eşleştirme işlemlerinde kayıtlı olmayan bir Yazar veya kitap için işlem yapılmamalı ve uyarı vermelidir. Veri tutarlılığı adına gerekli tüm kontroller yapılmalıdır.

```
1234567891011,1
1234567891011,2
1234567891011,3
1234567891012,1
1234567891013,4
1234567891014,4
1234567891014,1
```

```
1234567891011,-1
1234567891011,2
1234567891011,3
1234567891012,-1
1234567891013,4
1234567891014,4
1234567891014,-1
```

Şekil 2 Kitap Yazar Eşleştirmesi (Sol: Yazar silinmeden önce, Sağ: Yazar silindikten sonra)

- **Öğrenci Kitap Ödünç Alma İşlemleri:** Bir öğrenci çok sayıda kitabı ödünç alabilirken, bir kitap farklı zamanlarda çok sayıda öğrenci tarafından da ödünç alınabilir. Bu nedenle bu bilgilerin tutarlı bir şekilde saklanması gerekmektedir. Her bir örnek kitabın ödünç alınma ve teslim edilme durumlarını saklayan bir CSV dosyası ve buna uyumlu olacak bir struct tasarlayınız. Bu veri yapısında ÖğrenciID, KitapEtiketNO, İşlem Türü ve Tarih Bilgisi Şekil-3 ile gösterildiği gibi saklanmalıdır. İşlem türü bilgisi; kitabın ÖDÜNÇ ALINMASI – 0 ya da kitabın TESLİM EDİLMESİ – 1 olarak kodlanmalıdır. Bununla birlikte kitap ödünç alındığında Şekil-1 ve Madde-3 ile izah edildiği gibi, bir kitabın Raf Durum Bilgisi, Rafta ya da ÖğrenciNo olarak ilgili dosyalarda ve veri yapısında güncellemelidir.
  NOT1: Kitap ödünç alma ve teslim işlemlerinde kayıtlı olmayan bir öğrenci veya kitap için işlem yapılmamalı ve uyarı vermelidir. Veri tutarlılığı adına gerekli kontroller yapılmalıdır. Ayrıca, bir öğrencinin Puan Bilgisi negatif durumda olursa kitap verilemez uyarısı ile işlem iptal edilmelidir.
  NOT2: Kitap teslimi kitabın ödünç alındığı tarihten 15 günden sonra yapılırsa ilgili öğrenciye -10 ceza puanı verilmeli ve bu bilgiler ilgili dosya ve veri yapısında güncellenmelidir.
  NOT3: Kitap ödünç alma işlemleri sırasında, bir kitaba ait tüm örnekler başka öğrenciler tarafından ödünç alınmışsa İŞLEM BAŞARISIZ uyarısı verilmelidir.

```
1234567891011_1,10011088,0,12.10.2022
1234567891018_2,10011048,0,12.10.2022
1234567891011_1,10011088,1,22.10.2022
1234567891012_1,10011048,1,27.10.2022
1234567891018_2,10011088,0,28.10.2022
```

Şekil 3 Kitap Ödünç Alma İşlemleri

STRUCT YAPILARI VE BELLEK YÖNETİMİ

STRUCT NEDİR?

C programlama dilinde struct, objeler yaratmamızı sağlayan blueprintlerdir. Struct'a göre object yaratmak için iki yöntem bulunur.

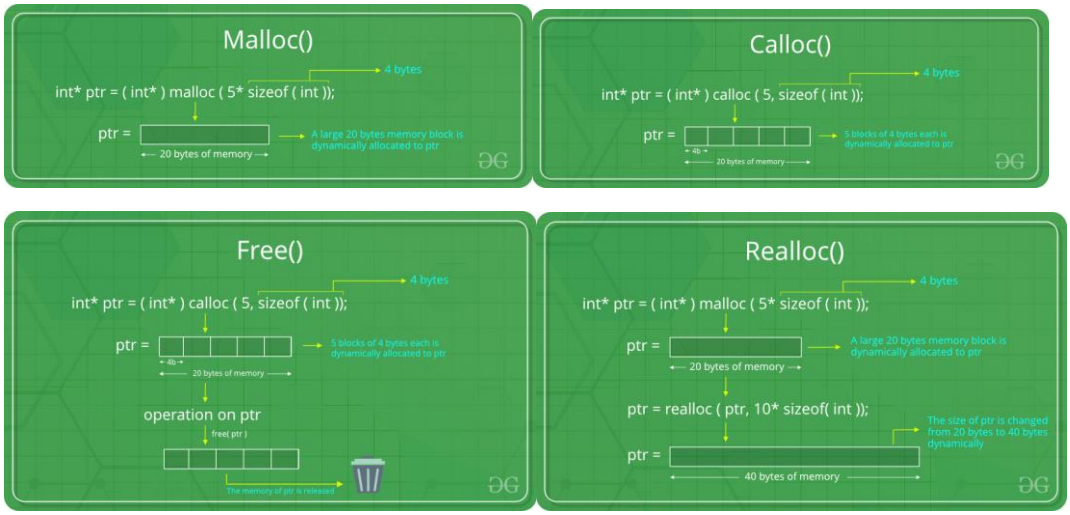
→ CLASSICAL DEFINITION
```
struct Person person1, person2, p[20];
```
→ DECLARATION OF OBJECT FROM STRUCT WITH POINTER
```
struct person *ptr;
```

BELLEK YÖNETİMİ

C, garbage collector içermeyen ve değişkenlerin memory allocationının manual olarak yapılmasını gerektiren bir dildir. Statically-typed olduğu için de allocationlarda type belirtip runtime boyunca o konumu boşaltmadığımız sürece ayırdığımız konumda aynı tip veriler kullanmak zorundayız.

Main memory, yani RAM bir matrixe benzer ve her bir veri hexadecimal sayı ile belirtilen adreslerde tutulur.

C dilinde bulunan ***stdlib*** standard kütüphanesi ile dinamik bellek yönetimi yapılabilir.



FONKSİYONLAR VE ÇALIŞMA YÖNTEMLERİ

ÖĞRENCİ FONKSİYONLARI

- **Ogrenci* createNodeOgrenci(char ogrID[9], char ad[30], char soyad[30], int puan) :**
  Yeni bir öğrenci node yaratır ve bunun adresini return eder.
- **void addOgrenci(char ogrID[9], char ad[30], char soyad[30], int puan,ListOgrenci* list):**
  Yaratılan öğrenci nodeunu linkli listeye ekler.
- **void displayOgrenciler(ListOgrenci* list):**
  Linkli listedeki öğrencileri düzgün formatta ekrana yazdırır.
- **void displayOgrenci(char ogrID[9],ListOgrenci* list):**
  İstenen ID'deki öğrenci eğer varsa öğrenciye dair bilgileri ekrana yazdırır.
- **void deleteNodeOgrenci(char ogrID[9],ListOgrenci* list):**
  İstenen ID'deki öğrenciyi linkli listeden varsa siler.
- **Ogrenci* findOgrenci(char ogrID[9],ListOgrenci* list):**
  İstenen ID'deki öğrenciyi linkli listede bulur ve adresini return eder.

YAZAR FONKSİYONLARI

- **Yazar* createNodeYazar(int yazarID,char yazarAd[30],char yazarSoyad[30]):**
  Yeni bir yazar node yaratır ve adresini return eder.
- **void addYazar(int yazarID,char yazarAd[30],char yazarSoyad[30],ListYazar* list):**
  Yaratılan öğrenci nodeunu linkli listeye ekler.
- **void displayYazarlar(ListYazar* list):**
  Yazarları linkli listede traverse edip ekrana yazdırır.
- **void deleteNodeYazar(int yazarID,ListYazar* list,int* yazarCounter):**
  YazarID'si alıp linkli listede varsa listeden siler.
- **Yazar* findYazar(int yazarID,ListYazar* list):**
  Verilen yazarID'ye sahip yazarı bulup adresini return eder.

KİTAP FONKSİYONLARI

- **Kitap* createNodeKitap(char kitapAdi[30], char ISBN[14],int adet):**
  Yeni bir kitap node yaratır ve adresini return eder.
- **KitapOrnek* createOrnekKitapNode(KitapOdunc* koArray,int* kitapOduncCounter,char ISBN[14],char Durum[8],int i):**
  Örnek kitap node'u yaratır.
- **void addOrnekKitap(KitapOdunc* koArray,int* kitapOduncCounter,char Durum[8],int i,Kitap* kitap):**
  Yaratılan örnek kitap nodeunu ait olduğu kitabın altındaki sub-list'e pushlar.
- **Kitap* addKitap(KitapOdunc* koArray,int* kitapOduncCounter,char kitapAdi[30], char ISBN[14], int adet,ListKitap* list) :**
  Yaratılan kitabı linkli listeye ekler.
- **void displayOrnekKitaplar(Kitap* kitap):**
  Örnek kitapları listeler.
- **void displayKitaplar(ListKitap* list):**
  Kitapları listeler.
- **void loopOrnekKitap(KitapOdunc* koArray,int* kitapOduncCounter,Kitap* kitap):**
  Sadece bir kitaba ait örnek kitapları döngü şeklinde dönüp işleme tabi tutar.
- **Kitap* findKitap(char ISBN[14],ListKitap* list):**
  Verilen ISBN numarasina göre kitabı bulur ve adresini return eder.
- **void deleteNodeKitap(ListKitap* list,char ISBN[14]):**
  Kitap nodeunu listeden siler.
-

KİTAP-YAZAR FONKSİYONLARI

- **KitapYazar* createKitapYazarNode(char ISBN[14],int yazarID):**
  Kitap-Yazar ilişkisini temsil eden bir node yaratır
- **void displayKitapYazar(KitapYazar* array,int* kitapYazarCounter);**
  Dinamik arrayde saklanan kitap-yazar verilerini ekrana yazdırır.
- **void deleteKitapYazarID(KitapYazar* array,int* kitapYazarCounter,int yazarID):**
  Kitap yazar ilişkisini ortadan kaldırır.

KİTAP-ÖDÜNÇ FONKSİYONLARI

- **KitapOdunc* createKitapOduncNode(char EtiketNo[20],char ogrID[9],int islemTipi,unsigned int gun, unsigned int ay,unsigned int yil):**
  Kitap-Ödünç nodeu yaratır ve adresini return eder.

READ-WRITE FONKSİYONLARI

- **void readOgrenci(ListOgrenci* list);**
- **void readYazarlar(ListYazar* list,int*);**
- **void readKitaplar(KitapOdunc* koArray,int* kitapOduncCounter,ListKitap* list);**
- **void readOduncKitaplar(int* kitapOduncCounter,KitapOdunc* koArray,ListKitap* List);**
- **void readKitapYazar(KitapYazar*,int*);**

- **void updateKitapYazar(KitapYazar* kyArray,int* kitapYazarCounter);**
- **void updateOgrenciler(ListOgrenci* listOgrenciler);**
- **void updateYazarlar(ListYazar* list,int* yazarCounter);**
- **void updateKitaplar(ListKitap* list);**
- **void updateKitapOdunc(KitapOdunc* koArray,int* kitapOduncCounter);**