C de Dosya İşlemleri

C de Dosya İşlemleri

Dosya, veri depolamak için kullanılan bilgisayar depolama aygıtlarındaki (disk,cd,dvd,usb bellek) bilgiler topluluğudur.

Dosyalara neden ihtiyaç duyulur?

Bir program sonlandırıldığında, tüm veriler kaybolur. Veileri bir dosyada saklamak, program sonlandırılsa bile verilerinizi koruyacaktır.

Çok sayıda veri girmeniz gerekiyorsa, hepsini girmek çok zaman alacaktır. Ancak, tüm verileri içeren bir dosyanız varsa, C'deki birkaç komutu kullanarak dosyanın içeriğine kolayca erişebilirsiniz.

Verilerinizi herhangi bir değişiklik yapmadan bir bilgisayardan diğerine kolayca taşıyabilirsiniz.

C de Dosya İşlemleri

Dosya Türleri

Dosyalarla uğraşırken bilmeniz gereken iki tür dosya vardır:

- Metin dosyaları
- Binary (ikili) dosyalar

C de Dosya İşlemleri

Dosya Türleri

1. Metin dosyaları

Metin dosyaları normal **.txt** dosyalarıdır. Not Defteri gibi herhangi bir basit metin düzenleyiciyi kullanarak kolayca metin dosyaları oluşturabilirsiniz.

Bu dosyaları açtığınızda, dosyanın içindeki tüm içeriği düz metin olarak göreceksiniz. İçeriği kolayca düzenleyebilir veya silebilirsiniz.

Bakımları için minimum çaba gerekir, kolayca okunurlar ve çok az bir

C de Dosya İşlemleri

Dosya Türleri

2. İkili (binary) dosyalar

İkili dosyalar çoğunlukla bilgisayarınızdaki **.bin** dosyalarıdır.

Verileri düz metin olarak depolamak yerine ikili biçimde (0'lar ve 1'ler) şeklinde depolarlar.

Daha yüksek miktarda veri tutabilirler, kolayca okunamazlar ve metin dosyalarından daha iyi güvenlik sağlarlar.

C de Dosya İşlemleri

Dosya İşlemleri

C'de, dosyalar üzerinde metin veya ikili olmak üzere dört ana işlem gerçekleştirebilirsiniz:

a. Yeni bir dosya oluşturma

b. Mevcut bir dosyayı açma

c. Bir dosyayı kapatma

d. Dosyadan bilgi okuma ve dosyaya bilgi yazma

C de Dosya İşlemleri

Dosyalarla çalışma

Dosyalarla çalışırken, FILE türünde bir işaretçi bildirmeniz gerekir.

Bu bildirim, dosya ve program arasındaki iletişim için gereklidir.

```
FILE *ptr;
```

C de Dosya İşlemleri

Dosya açma - oluşturma ve düzenleme için

Bir dosyanın açılması fopen(), stdio.h kütüphanesinde tanımlanan fonksiyon kullanılarak gerçekleştirilir .

Standart G/Ç'de bir dosya açma sözdizimi şöyledir:

ptr = fopen("fileopen","mode");

C de Dosya İşlemleri

Örneğin,

fopen("C:\\cprogram\\deneme.txt","w");

fopen("C:\\cprogram\\deneme.bin","rb");

* dosya konumda yoksa yeni bir dosya oluşturur deneme.txt ve onu 'w' modu yani yazmak için açar. Yazma (w) modu, dosyanın içeriğini oluşturmanıza ve düzenlemenize (üzerine yazmanıza) olanak tanır.

* İkili dosyanın konumunda varsa 'rb' ikili modunda okumak için açar . Dosya yoksa, fopen()NULL döndürür. Okuma modu sadece dosyayı okumanıza izin verir, dosyaya yazamazsınız.

C de Dosya İşlemleri

mod	Modun Anlamı	Dosyanın Yokluğu Sırasında
r	Okumak için açık.	Dosya yoksa, fopen()NULL döndürür.
rb	İkili modda okumak için açın.	Dosya yoksa, fopen()NULL döndürür.
w	Yazmaya açık.	Dosya varsa, içeriğinin üzerine yazılır. Dosya yoksa, oluşturulacaktır.Dosya yoksa, oluşturulacaktır.
wb	İkili modda yazmak için açın.	Dosya varsa, içeriğinin üzerine yazılır. Dosya yoksa, oluşturulacaktır.
a	Eklemek için açın.	Dosya yoksa, oluşturulacaktır.Veriler dosyanın sonuna eklenir.
ab	İkili modda eklemek için açın.	Dosya yoksa, oluşturulacaktır.Veriler dosyanın sonuna eklenir.
r+	Hem okumaya hem de yazmaya açık.	Dosya yoksa, fopen()NULL döndürür.
rb+	İkili modda hem okuma hem de yazma için açın.	Dosya yoksa, fopen()NULL döndürür.
w+	Hem okumaya hem de yazmaya açık.	Dosya varsa, içeriğinin üzerine yazılır.Dosya yoksa, oluşturulacaktır.
wb+	İkili modda hem okuma hem de yazma için açın.	Dosya varsa, içeriğinin üzerine yazılır. Dosya yoksa, oluşturulacaktır.
a+	Hem okumaya hem de eklemeye açık.	Dosya yoksa, oluşturulacaktır.
ab+	İkili modda hem okuma hem de ekleme için açın.	Dosya yoksa, oluşturulacaktır.

C de Dosya İşlemleri

Dosya Kapatma

Dosya (hem metin hem de ikili) okuma/yazma işleminden sonra kapatılmalıdır.

Bir dosyanın kapatılması fclose()işlevi kullanılarak gerçekleştirilir .

```
fclose(fptr);
```

Burada, fptr kapatılacak dosyayla ilişkili bir dosya işaretçisi bulunur.

C de Dosya İşlemleri

Bir metin dosyasına okuma ve yazma

Bir metin dosyasını okumak ve yazmak için işlevleri fprintf() ve fscanf().

Dosya için bir işaretçi bekler.

Örnek 1: Bir metin dosyasına yazma

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
   int sayi;
   FILE *fptr;
   fptr = fopen("C:\\deneme.txt","w");
   if(fptr == NULL)
   {
      printf("HATA!");  exit(1);
//exit 1 başarısız bir sonlandırmayı ifade eder
//exit 0 başarıslı bir sonlandırmayı ifade eder
    }
```

```
printf("Bir Sayı Giriniz: ");
   scanf("%d",&sayi);
   fprintf(fptr,"%d",sayi);
   fclose(fptr);
   return 0;

}
```

Örnek 2: Bir metin dosyasından okuma

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
   int sayi;
   FILE *fptr;
   if ((fptr = fopen("C:\\program.txt","r"))
== NULL){
       printf("Dosya açılırken HATA!
oluştu");
       // Dosya işaretçisi NULL döndürürse
programdan çıkar.
       exit(1);
}
fscanf(fptr,"%d", &sayi);
   printf("Değer =%d", sayi);
   fclose(fptr);
     return 0;
}
```

C de Dosya İşlemleri

- **C Programlama Dilinde Yapılar (struct)**

Programlama dillerinde veri yapıları önemli bir yer tutar. İşlenecek verilerin:

hangi yapıda

bellekte ne şekilde

hangi sırayla tutulacağını

programı yazarken biz belirleriz.

Bu işlem doğru şekilde yapıldığında program yazma işlemi kolaylaşır ve program kodları da daha anlaşılır olur.

C de Dosya İşlemleri

- **C Programlama Dilinde Yapılar (struct)**

Bazı programlarda bir öğeye ait bir kaç farklı özellik için farklı tiplerde değişkenler kullanmamız gerekir. Örneğin bir öğrencinin bilgilerini bellekte tutmak istersek öğrencinin adı, doğum tarihi, sınıfı gibi bilgilerin tutulması için değişik tiplerde değişkenler kullanmamız gerekir. Bu tip durumlar için C'de kullanılan yapı 'struct' dır. Struct içerisinde istenilen tipte değişken tanımlanabilir. Bunu içindeki verileri bir arada tutan bir veri paketi olarak düşünebiliriz.

C de Dosya İşlemleri

- **C Programlama Dilinde Yapılar (struct)**

```
struct Personel
{
    char isim[50];
    int pinNo;
    float maas;
};
```

C de Dosya İşlemleri

- **C Programlama Dilinde Yapılar (struct)**

Örnek 3:

```
#include <stdio.h>
//Struct tanımı
struct POINT{
    int x;
    int y;
};
int main()
{
    struct POINT p; //Point tipinde bir struct oluşturuluyor.
    //p'nin içeriğinde bulunan x ve y değişkenlerine değerler veriliyor.
    p.x=234;
    p.y=987;
    //p'nin içeriği ekrana yazdırılıyor.
    printf("x=%d\ny=%d",p.x,p.y);
    return 0;
}
```

C de Dosya İşlemleri

Bir ikili dosyaya okuma ve yazma

Fonksiyonlar fread() ve fwrite() ikili dosyalar olması durumunda sırasıyla

diskteki bir dosyadan okumak ve bir dosyaya yazmak için kullanılır.

Bir ikili dosyaya yazma

İkili bir dosyaya yazmak için fwrite() işlevi kullanmamız gerekir . Fonksiyonlar dört parametre alır:

- diske yazılacak verilerin adresi
- diske yazılacak verinin boyutu
- bu tür veri sayısı
- yazmak istediğiniz dosyanın işaretçisi.

fwrite(adresverisi, veribüyüklüğü, verisayısı, dosyanın işaretçisi);

Örnek 4: Bir metin dosyasından yazma

```
#include <stdio.h>
#include <stdlib.h>
struct ucsayi
{
   int n1, n2, n3;
};
int main()
{
   int n;
   struct ucsayi s;
   FILE *fptr;
   if ((fptr = fopen("C:\\program.bin","wb")) == NULL){
       printf("Error! opening file");
       // Program exits if the file pointer returns NULL.
       exit(1);
   }
for(n = 1; n < 5; ++n)
   {
      s.n1 = n;
      s.n2 = 5*n;
      s.n3 = 5*n + 1;
      fwrite(&num, sizeof(struct ucsayi), 1,
fptr);
   }
   fclose(fptr);
    return 0;
}
```

C de Dosya İşlemleri

Bir ikili dosyadan okuma

Fonksiyon fread() ve fwrite() fonksiyonları kullanılır. 4 parametresi vardır .

fread(**adresverisi, veribüyüklüğü, verisayısı, dosyanın işaretçisi**);

Örnek 5: Bir metin dosyasından okuma

```
#include <stdlib.h>
struct ucsayi
{
   int n1, n2, n3;
};
int main()
{
   int n;
   struct ucsayi s;
   FILE *fptr;
   if ((fptr = fopen("C:\\program.bin","rb")) == NULL){
       printf("Dosya açılırken hata oluştu");
exit(1);
   }
for(n = 1; n < 5; ++n)
   {
      fread(&s, sizeof(struct ucsayi), 1, fptr);
printf("n1: %d\tn2: %d\tn3: %d\n", num.n1, num.n2, num.n3);
   }
   fclose(fptr);
    return 0;
}
```

C de Dosya İşlemleri

fseek() kullanarak veri alma/arama

Bir dosyanın içinde çok sayıda kayıt varsa ve belirli bir konumdaki bir kayda erişmeniz gerekiyorsa, kaydı almak için önceki tüm kayıtları gözden geçirmeniz gerekir.

Bu, çok fazla bellek ve çalışma zamanı gerek olacaktır. Gerekli verilere ulaşmanın daha kolay bir yolu kullanılarak elde edilebilir: **fseek()**

fseek() Kusörü(imleç) dosyada verilen kayda arar.

fseek() kullanımı:

fseek(FILE * stream, long int offset, int whence);

C de Dosya İşlemleri

fseek() kullanarak veri alma/arama

fseek() kullanımı:

fseek(FILE * stream, long int offset, int whence);

- **stream:** Konumu değiştirilecek dosyanın pointer (işaretçisini) gösterir.
- **offset:** Origin parametresine göre kaydırılacak karakter sayısını gösterir. Negatif bir değer aldığında dosya konum göstergesi geriye doğru hareket eder.
- **origin:** offset in konumunu gösterir. Aşağıdaki değerlerden birini içerir:

C de Dosya İşlemleri

fseek() kullanarak veri alma/arama

- **origin:** offset in konumunu gösterir. Aşağıdaki değerlerden birini içerir:
- SEEK_SET: Ofseti dosyanın başından başlatır.
- SEEK_CUR: Dosyadaki kursörün geçerli (aktif) konumundan ofseti başlatır.
- SEEK_END: Ofseti dosyanın sonundan başlatır.

Örnek 6: fseek() kullanarak veri alma/arama

```
#include <stdio.h>
#include <stdlib.h>
struct threeNum
{
   int n1, n2, n3;
};
int main()
{
   int n;
   struct threeNum num;
   FILE *fptr;
   if ((fptr = fopen("C:\\program.bin","rb")) == NULL){
       printf("Error! opening file");
       // Dosya işaretçisi NULL döndürürse programdan
çıkar.       exit(1);
   }
// Kursörü dosyanın sonuna taşır.
   fseek(fptr, -sizeof(struct threeNum), SEEK_END);
   for(n = 1; n < 5; ++n)
   {
      fread(&num, sizeof(struct threeNum), 1, fptr);
      printf("n1: %d\tn2: %d\tn3: %d\n", num.n1, num.n2,
num.n3);
      fseek(fptr, -2*sizeof(struct threeNum),
SEEK_CUR);
   }
   fclose(fptr);
    return 0;
}
```