Metin Dosyaları

Metin Dosyaları

Dosya Açma ve Kapama

Dosya Okuma ve Yazma

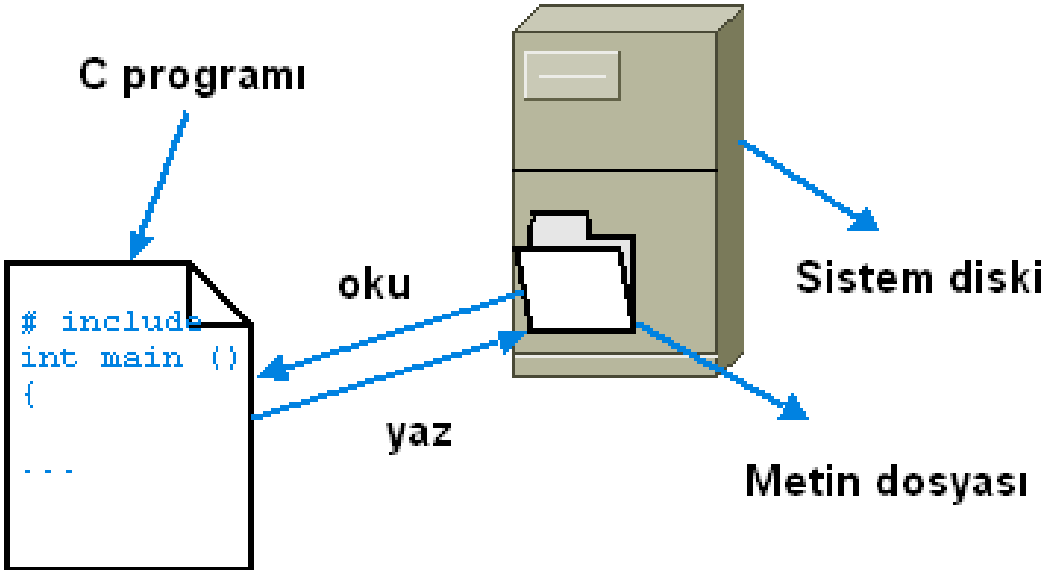
Rastgele Erişim

Standart Girdi/Çıktı

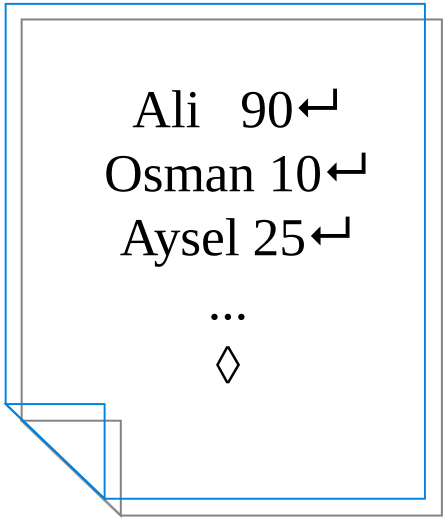
Hata Kontrolü

Metin Dosyaları

Metin dosyaları, verileri bir kere hazırlayıp, ikincil saklama biriminde (disk, CD v.b.) tutmaya ve daha sonra defalarca kullanmaya olanak tanır.



Metin Dosyaları



Her bir öğrenci için hazırlanan ve dosya içinde tek bir satırda tutulan bilgilerden her birine **kayıt** adı verilir. Örneğin,

"Ali 90" bir kayıttır.

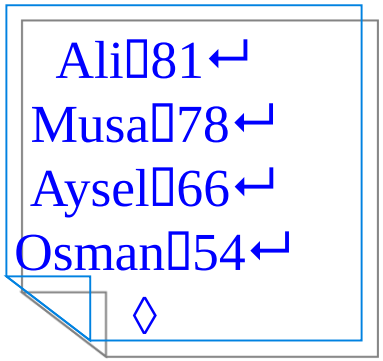
Dosyada,

↵ kayıtları ayırmakta,

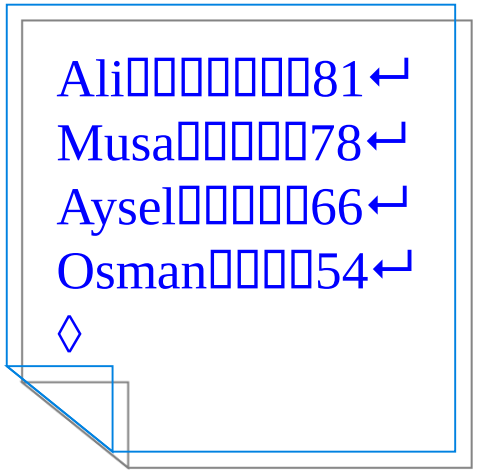
◊ dosya sonunu göstermektedir.

Metin Dosyaları

Değişken uzunlukta kayıtları içeren dosya:



Sabit uzunlukta kayıtları içeren dosya:



Dosya Açma ve Kapama

Dosya Göstergesi fiziksel dosyaya erişim için programın içinden tanımlanmış olan bir ***iç dosya adıdır.***

```
FILE *iç_dosya_adı;
```

Dış dosya adı ise, işletim sistemi üzerinde dosyaya verilen ismi temsil eder

Örnek: Disk üzerinde daha önceden hazırlamış olduğumuz bir dosyaya programımız içinden erişebilmek için aşağıdaki tanımlamayı yapalım.

```
FILE *ogrenciDosyası;
```

Dosya Açma ve Kapama

Dosya Açma

`FILE` * *iç_dosya_adı*;
iç_dosya_adı = **`fopen`**(*dış_dosya_adı*, *dosya_açma_modu*);

`fopen()` fonksiyonu dış_dosya_adı'nı taşıyan dosyanın açılmasını sağlar ve bu dosyanın adresinin programa döndürülmesini sağlar.

dosya_açma_modu : dosyanın hangi amaçla açılacağını belirlemek için kullanılan bir parametredir.

Dosya Açma ve Kapama

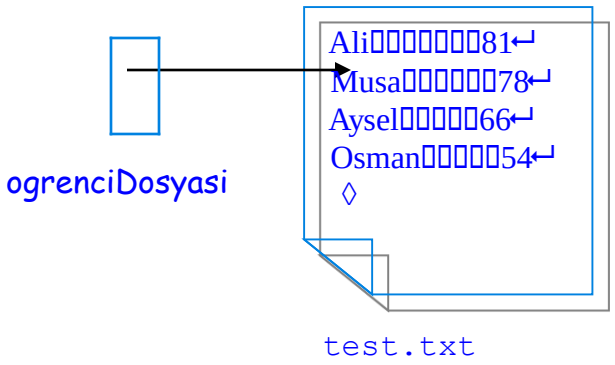
mod	Anlamı
`r`	Okuma (read)
`w`	Yazma (write) ve dosya yaratma
`a`	Sonuna ekleme (append)
`r+`	Okuma ve yazma
`w+`	Okuma, yazma ve dosya yaratma (önceki veriler silinir)
`a+`	Okuma, yazma ve dosya yaratma (önceki verilerin sonuna devam edilir)

Dosya Açma modları

Dosya Açma ve Kapama

Örnek:

```
FILE *ogrenciDosyasi;
ogrenciDosyasi = fopen("test.txt", "w");
```



Dosya Açma ve Kapama

Dosya Kapama

fclose() fonksiyonu daha önce **fopen()** fonksiyonu ile açılmış olan bir dosyanın kapatılmasını sağlar.

fclose(*iç_dosya_adı*);

Örnek:

```
FILE *ogrenciDosyasi;
ogrenciDosyasi=fopen("c:\\test.txt","w");

...

fclose(ogrenciDosyasi);
```

Dosya Okuma ve Yazma

Okuma İşlemleri

Metin dosyasi okuma amaçlı ‘r’ modunda açılmalıdır.

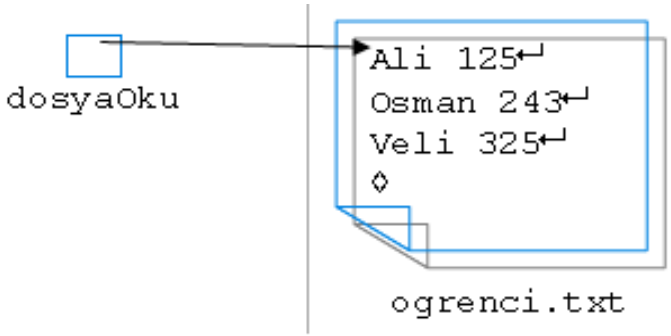
fgetc() fonksiyonu dosyadan göstergenin o an gösterdiği karakterin alınmasını sağlar.

fgetc(***iç_dosya_adı*****);**

Dosya Okuma ve Yazma

Örnek:

```
FILE *dosyaOku;
dosyaOku=fopen("input.txt","r");
char ch;
```




```
ch = fgetc(dosyaOku);
printf("Dosyadaki ilk karakter: %c\n", ch);
```

Çıktı:

```
Dosyadaki ilk karakter: A
```

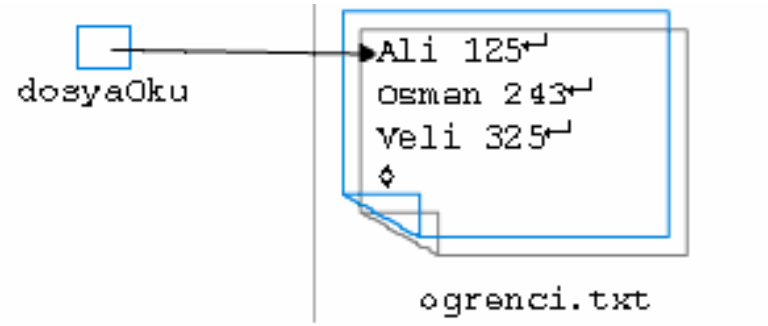
Dosya Okuma ve Yazma

`fgets()` fonksiyonu dosyadan bir dizgi okur ve bu dizginin göstergesini geri döndürür..

`fgets(`*dizgi, n, iç_dosya_adı*`);`

Örnek:

```
FILE *dosyaOku;
dosyaOku=fopen("input.txt","r");
char st1[30];
fgets(st1,4,dosyaOku);
printf("st1: %s\n", st1)
```



Çıktı:

```
st1: Ali
```

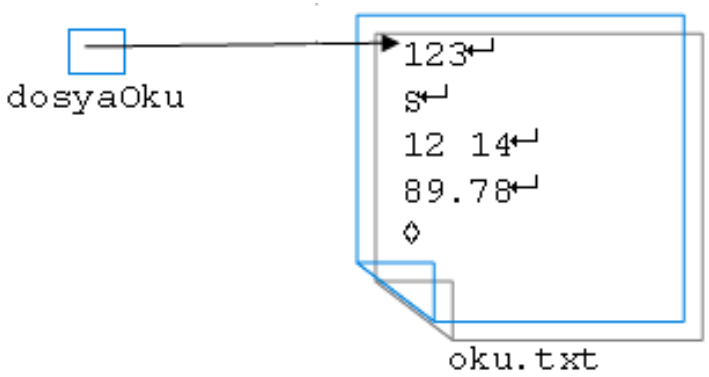
Dosya Okuma ve Yazma

fscanf() fonksiyonu **scanf()** fonksiyonuna benzer bir şekilde çalışır ve verileri dosyadan okur.

fscanf(*iç_dosya_adı, format, ...*);

Örnek:

```
int x, y,k;
double z,t;
char ch;
char st1[30],st2[30];
FILE *dosyaOku;
dosyaOku=fopen("oku.txt","r");
fscanf(dosyaOku,"%d\n",&x);
printf("%d\n", x);
```



Çıktı:

```
123
```

Dosya Okuma ve Yazma

Yazma İşlemleri

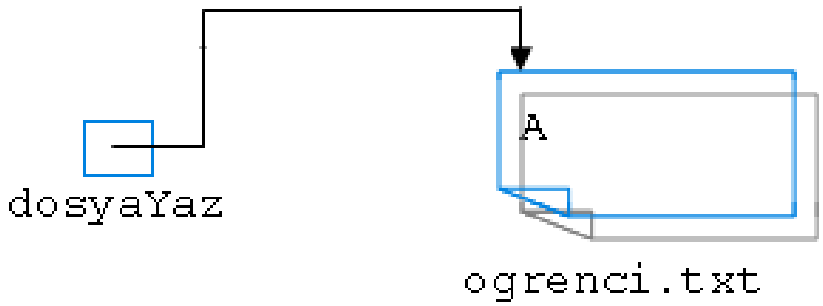
Metin dosyasi okuma amaçlı ‘w’ modunda açılmalıdır.

fputc() fonksiyonu tek bir karakterin dosyaya yazılmasını sağlar.

fputc(*karakter_tanımı, iç_dosya_adı*);

Örnek:

```
char ch='A';
FILE *dosyaYaz;
dosyaYaz=fopen("ogrenci.txt","w");
fputc(ch, dosyaYaz);
```



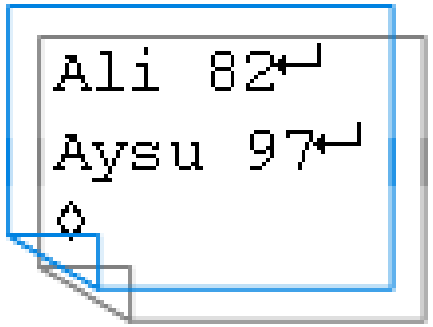
Dosya Okuma ve Yazma

`fputs()` fonksiyonu dosyaya bir dizginin yazılmasını sağlar.

`fputs(`*dizgi*`, `*iç_dosya_adı*`);`

Örnek:

```
FILE *dosyaYaz;
dosyaYaz=fopen("ogrenci.txt","w");
fputs("Ali 82\n",dosyaYaz);
fputs("Aysu 97\n",dosyaYaz);
fclose(dosyaYaz);
```

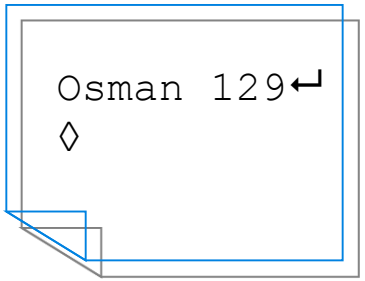


ogrenci.txt

Dosya Okuma ve Yazma

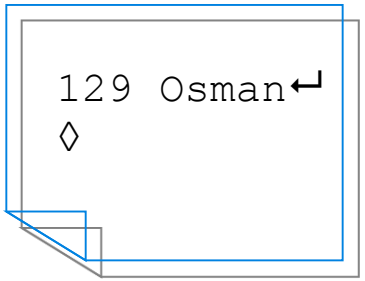
`fprintf()` fonksiyonu dosyaya formatlı bir biçimde yazma işlemi yapmak amacıyla kullanılır.

`fprintf(`*iç_dosya_adı*`, `*format*`, ....);`



Örnek:

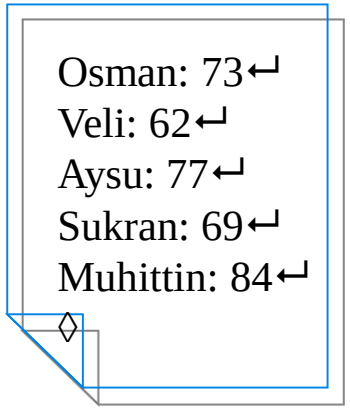
```
#include<stdio.h>
#include<math.h>
main(){
char okuDosyaAdi[] = "c:\\dosya_oku.txt";
char yazDosyaAdi[] = "c:\\dosya_yaz.txt";
char ad[30];
int no;
FILE *yfp, *ofp;
ofp = fopen(okuDosyaAdi,"r");
fscanf(ofp,"%s %d",&ad,&no);
yfp = fopen(yazDosyaAdi,"w");
fprintf(yfp,"%d %s\n",no, ad);
fclose(ofp);
fclose(yfp); }
```



Dosya Okuma ve Yazma

Örnek: Bir dosya içinden öğrencilerin dönem içindeki iki ara sınav ve bir final notunu okuyan ve dönem sonu notunu ara sınavların %25 ve final notunun %50 etkileyecek şekilde hesaplayarak diğer bir dosyaya yazdıran bir program yazınız.

```
#include <stdio.h>
int main(void)
{ int mt1, mt2, final, donemNotu;
  char ad[30];
  FILE *dosyaOku;
  FILE *dosyaYaz;
  dosyaOku=fopen("ogrenci.txt","r");
  dosyaYaz=fopen("notlar.txt","w");
  while (fscanf(dosyaOku,"%s %d %d %d\n",&ad, &mt1,
    &mt2, &final) != EOF)
  {  donemNotu = (mt1 *0.25)+(mt2*0.25)+(final*0.5);
     fprintf(dosyaYaz,"%s: %d\n", ad, donemNotu);
  }
fclose(dosyaOku);
fclose(dosyaYaz);
return(0);
 }
```



notlar.txt

Standart Girdi / Çıktı

stdout: genellikle bilgisayar ekranı olarak önceden tanımlanmış olan ve ***standart çıktı*** biriminin nereyi göstereceğini belirlemeye yarayan bir tanımlama alanıdır.

stdin: ***standart girdi*** birimini tanımlar ve genellikle klavye olarak önceden tanımlanmıştır.

Örnek: `fprintf()` fonksiyonunun çıktısının bir dosya içine değil de ekrana yansıtılması mümkündür.

```
fprintf(stdout,"Merhaba Dunya\n");
```

Örnek: Benzer bir işlem **`fscanf()`** fonksiyonu ile aşağıdaki gibi gerçekleştirilir.

```
fscanf(stdin,"%d",&sayi1);
```

Hata Kontrolü

```
FILE *dg;
```

tanımlamasına göre hata kontrolleri

Fonksiyon Tanımı	Açıklama
`clearerr(dg);`	Hata ve dosya-sonu durumunun temizlenmesini sağlar.
`feof(dg);`	Dosya-sonuna (EOF) ulaşılmış ise sıfırdan farklı bir değer, diğer durumlarda sıfır döndürür.
`ferror(dg);`	Hata durumu oluşmuş ise sıfırdan farklı bir değer, diğer durumlarda sıfır değerini döndürür.
`perror(s);`	Standart çıktı biriminde tek satırdan oluşan bir hata mesajının gösterilmesini sağlar.

Hata Kontrolü

Örnek:

```
FILE *ofp;
ofp=fopen("c:\\dosya_oku.txt","r");
if (ofp==NULL)
   perror ("Hata: Dosya Acilamadi");
else {
   fputc ('x',ofp);
   if (ferror (ofp)){
     printf ("Hata: c:/dosya_oku.txt dosyasina");
     printf (" yazma islemi gerceklesmedi\n");
   }
}
fclose (ofp);
```

C++ PROGRAMLAMADA DOSYALAMA İŞLEMİ

C++ ta dosyaya kaydetme ve dosyadan okuma işlemlerini ofstream ve ifstream sınıflarını kullanarak yapıyoruz. Bu sınıfları içinde barındıran fstream dosyasını öncelikle programımıza eklememiz gerekiyor.

#include<fstream>

Dosyaya kayıt yapmak için ofstream sınıfından bir nesne oluşturuyoruz, <<operatörü ile verilerimizi dosyaya kaydediyoruz.

```
#include<fstream>
using namespace std;
int main () {
int x;
ofstream yaz("Veri.txt");
x=5;
yaz<<x;
yaz.close();
return 0;  }
```

Görüldüğü üzere x değişkenine 5 değeri atandı ve dosyaya yazdırıldı.

Peki aynı dosyaya tekrar bir şeyler kaydetmek istediğimizde ne yapacağız?

Bu kez dosyayı ekleme(append) modunda açmamız gerekiyor.

ofstream yaz("Veri.txt", ios::append); //append yerine app yazılabilir

Burada dikkat etmemiz gereken nokta dosyaya kaydedilen verilerin birbirine karışmamasıdır.
Mesela biraz önce Veri.txt dosyasına 5 yazdırdık. Ardından herhangi bir sayı daha yazdırdığımızda dosyada bu sayılar tek bir sayı gibi görünecektir (5 ve 2 yazdırdığımızda 52). Daha sonra programdan dosyadaki veriyi okumasını istediğinizde 52 olarak okuyacaktır. Bu karışıklığı engellemek için kaydedeceğimiz verilerin arasına bir boşluk koyuyoruz.

```
int main ()
{
  int y;
  ofstream yaz(“Veri.txt”, ios::app);    //ofstream sınıfından yaz isminde bir nesne oluşturduk
   y=2;
   yaz<<” “<<y;
  yaz.close();
  return 0;
}
```

Görüldüğü üzere y değişkenini yazdırmadan önce bir boşluk ekledik.

Dosyadan veri okuma işlemi ise ifstream sınıfı kullanılarak yapılır.

```
int main ()
{
  int x;
  ifstream oku("Veri.txt");    //ifstream sınıfından "oku" nesnesini oluşturduk.
  oku>>x;                      //dosyada bulunan ilk veriyi okuyup x değişkenine
atadık.
  oku.close();
  return 0;
}
```

Peki dosyada birden fazla veri varsa ne yapacağız?

Dosya işlemleri sıralı erişim tipinde işlemlerdir. Dosyada bir veriyi okumak istediğinizde onun önünde bulunan tüm verileri okuyup atlamanız gerekir. Örneğin dosyada bulunan 5. veriyi okumak istediğinizde önce ilk 4 veriyi okuyup atlamanız gerekiyor.

```
for(i=0;i<5;i++)
{
oku>>x;
}
```

şeklinde uygulanabilir.

Burada önemli olan nokta, dosya açıldıktan sonra yani ifstream sınıfından bir nesne oluşturulduktan sonra dosya kapatılana kadar dosyaya sıralı erişim yapıldığıdır.

Örnek :

```
#include<iostream>
#include<fstream>
using namespace std;
int main()
{
ifstream oku("Veri.txt",ios::app)
oku>>x;  //Dosyadaki ilk veriyi x değişkenine atadık.
cout<<x<<endl;
oku>>y;  //Dosyadaki ikinci veriyi y değişkenine atadık.
cout<<y<<endl;
oku.close();
}
```

Dosyada sırasıyla 5 ve 2 sayılarının bulunduğunu varsayarsak ekrana önce 5 sonra 2 sayısı yazılacaktır. Yani oku.close() komutu kullanılana kadar dosyadan okuma işlemi yaptığınızda program en son kaldığı yerden verileri okumaya devam edecektir.

Ödev: Basit bir personel takip programı yapınız. Burada karşılama ekranında personel giriş, çıkış gibi fonksiyon seçenekleri olacaktır.