**DOSYA İŞLEMLERİ**



"**FILE**" tipi, bir dosya değişkenidir, ve "stdio.h" in içinde tanımlanmıştır. Kullanacağımız dosyaya erişmek için bir 'dosya pointeri' tanımlamaktadır. Bu pointer sayesinde, dosyayı açtıktan sonra, ondan okumak ya da yazmak gibi işlemler yapabiliriz.



**Temel dosya işlem fonksiyonları aşağıda yer almaktadır:**

**fopen( )** Bir dosya açar.



**fclose( )** Bir dosyayı kapatır.

**fputc( )** Dosyaya bir karakter yazar.



**fgetc( )** Dosyadan bir karakter okur.



**fseek( )** Bir dosyadaki belirli bir byte'ı bulur.

**fprintf( )** Yapılandırılmış verileri dosyaya yazar.



**fscanf( )** Yapılandırılmış verileri dosyadan okur.



**feof( )** Dosya sonu geldiğinde doğru bir değer verir.

**ferror( )** Bir hata durumunda doğru bir değer verir.

**rewind( )** Dosya aktif konumunu başa alır.

**remove( )** Dosyayı siler.



**fflush( )** Tampon belleği siler.



**Dosya Açma**

FILE \*fopen (char \*dosya-adı, char \*mod);



dosya-adı: açılacak dosya adı



mod : dosya açılış şekli



**MOD Anlamı**

r Okuma için bir metin dosyası açar.



w Yazma için bir metin dosyası oluşturur.



a Bir metin dosyasına ekleme yapar.



rb Okuma için bir dosyayı ikili sistemde açar.



wb Yazma için ikili sistemde bir dosya oluşturur.



ab İkili sistemde bir dosyaya ekleme yapar.



r+ Okuma ve yazma için bir metin dosyası açar.



w+ Okuma ve yazma için bir metin dosyası oluşturur.



a+ Okuma ve yazma için bir metin dosyası oluşturur veya ekleme yapar.



r+b Okuma ve yazma için bir ikili sistem dosyası açar.



w+b Okuma ve yazma için bir ikili sistem dosyası oluşturur.



a+b Okuma ve yazma için bir ikili sistem dosyasına ekleme yapar.



Eğer fopen() fonksiyonu normal olarak çalışmazsa NULL bir işaretçi geri verir.



**Dosya Kapatma**



Bir dosyayı kapatmak için aşağıda prototipi verilen fclose() fonksiyonu kullanılır :

int fclose (FILE \*fp);



fp: fopen() ile kullanılan dosya işaretçisi

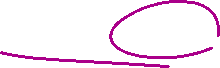
fclose() fonksiyonu fp ile bağlantılı dosyayı kapatır ve dosyanın akış ile olan bağlantısını keser.



**fgetc() ve fputc() Fonksiyonları**



Bir dosya açıldıktan sonra, açılış moduna bağlı olarak, aşağıda genel yapısı verilen 2 fonksiyonu kullanarak dosyaya bir karakter yazabilir veya dosyadan bir karakter okuyabilirsiniz:



int fgetc (FILE \*fp);

int fputc (int id, FILE \*fp);

fgetc() fonksiyonu, fp ile tanımlanan dosyadaki bir sonraki byte'ı unsigned char olarak okur ve int bir değer olarak geri verir.

fputc() fonksiyonu id değişken değeri olan byte'ı fp ile gösterilen dosyaya unsigned char olarak yazar.

fgetc() yerine getc() ifadesini, fputc() fonksiyonu yerine ise putc() ifadesini kullanabilirsiniz. Sonuçta, fonksiyonlar tarafından yapılan işlem tamamen birbirinin aynıdır.



#include <stdio.h>

#include <conio.h>

int main ()

{

FILE \*fp;

int id1;



fp = fopen ("dosya1.txt", "w");

if (fp == NULL) {

printf("Dosya açma hatası!");

getch();



return 0;

}

for (id1=0; id1<255; id1++) fputc (id1, fp);



fclose(fp);



fp = fopen ("dosya1.txt", "r");



if (fp == NULL) {

printf("Dosya açma hatası!");

getch();



return 0;

}



for (id1=0; id1<255; id1++) printf("%c", fgetc(fp));



fclose(fp);

getch();

return 0;

}

**feof() ve ferror() Fonksiyonları**



fgetc() fonksiyonu, aşağıda belirtilen 2 farklı durumda, EOF değerini geri verir:



1. Bir hata meydana geldiğinde



2. Dosya sonuna gelindiğinde



fgetc() fonksiyonu EOF değerini geri verdiğinde, yukarıdaki durumlardan hangisine göre bu değeri geri verdiğini bilemezsiniz.



#include <stdio.h>

#include <stdlib.h>

#include <conio.h>

int main ()

{

FILE \*fp1, \*fp2;



char cd1;



if ((fp1=fopen("deneme.exe", "rb")) == NULL) {



printf("Kaynak dosya açılamadı!\n");

getch();



return 0;

}



if ((fp2=fopen("deneme\_kopya.exe", "wb")) == NULL) {

printf("Hedef dosya açılamadı!\n");



getch();



return 0;

}

while (!feof(fp1)) {



cd1 = fgetc(fp1);



if (ferror(fp1)) {

printf("Kaynak dosyadan okuma hatası!\n");

getch();



return 0;

}

if (!feof(fp1)) fputc (cd1, fp2);

if (ferror(fp2)) {

printf("Hedef dosyaya yazma hatası!\n");

getch();



return 0;

}

}

fclose (fp1);

fclose (fp2);

}

**fputs(), fgets(), fprintf() ve fscanf() Fonksiyonları**



int fputs (char \*cdizi, FILE \*fp);

char \*fgets (char \*cdizi, int id1, FILE \*fp);

int fprintf (FILE \*fp, char \*kontrol dizisi,...);

int fscanf (FILE \*fp, char \*kontrol dizisi,...);

fputs() fonksiyonu, cdizi ifadesi ile gösterilen karakter dizisini fp ile gösterilen dosyaya yazar. Normal olarak çalışırsa 0 olmayan bir değer, aksi takdirde EOF değerini geri verir. cdizi karakter dizisini sona erdiren NULL değeri dosyaya yazmaz.

fgets() fonksiyonu, fp ile gösterilen dosyadan okuduğu karakterleri cdizi ifadesi ile gösterilen karakter dizisine atar. Normal olarak çalışırsa cdizi karakter dizisini, aksi takdirde NULL bir işaretçi geri verir. id1 değişkeni ise, dosyadan okunacak karakter sayısını belirler. fgets() fonksiyonu, id1-1 kadar karakter okuduğu veya bir yeni satır karakteri ile karşılaştığı zaman, okuma işlemine son verir.

fprintf() fonksiyonu tıpkı printf() fonksiyonu gibi çalışır. Tek fark fprintf() fonksiyonunun ekran yerine dosyalarla çalışmasıdır. Normal olarak çalıştığında dosyaya yazdığı byte sayısını, hata durumunda ise EOF değerini geri verir.

fscanf() fonksiyonu ise tıpkı scanf() fonksiyonu gibi çalışır. Tek fark fscanf() fonksiyonunun ekran yerine dosyalarla çalışmasıdır. Normal olarak çalıştığında, okuduğu ve değişkenlere yüklediği alan sayısını geri verir. Geri verdiği değer değişkenlere yüklenmeyen alanları içermez. Eğer fscanf() fonksiyonu dosya sonunu okursa EOF değerini, hiç bir alan yüklemezse 0 değerini geri verir.

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <conio.h>

int main ()

{

FILE \*fp1;

char cdizi[80];



if ((fp1=fopen("deneme.txt", "w")) == NULL) {



printf("Dosya açılamadı!\n");



getch();

return 0;



}

do {



printf("Bir karakter dizisi giriniz: ");

gets(cdizi);

strcat(cdizi, "\n");



if (\*cdizi!='\n') fputs(cdizi, fp1);

} while (\*cdizi!='\n');



fclose (fp1);

if ((fp1=fopen("deneme.txt", "r")) == NULL) {

printf("Dosya açılamadı!\n");

getch();

return 0;

}

do {

if (fgets(cdizi, 79, fp1)) printf(cdizi);



} while (!feof(fp1));



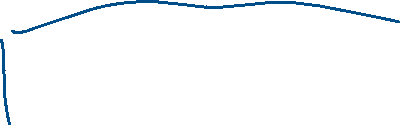
fclose (fp1);

getch();

}

#include <stdio.h>

#include <stdlib.h>



#include <string.h>

#include <conio.h>

int main ()

{

FILE \*fp1;

char cdizi[40];

double dd1;



int id1;

char cd1;

if ((fp1=fopen("deneme.txt", "w")) == NULL) {

printf("Dosya açılamadı!\n");

getch();



return 0;

}



fprintf (fp1,"%lf %d %s %c", 654.123, 9852, "Bilgisayar", 'A');



fclose (fp1);

if ((fp1=fopen ("deneme.txt", "r")) == NULL) {

printf("Dosya açılamadı!\n");



getch();



return 0;

}

fscanf (fp1,"%lf%d%s %c", &dd1, &id1, cdizi, &cd1);

printf("%lf %d %s %c", dd1, id1, cdizi, cd1);



fclose (fp1);

getch();

}

**ÖDEVLER:**

1. 0 ile 1000 arasında rasgele ürettiğinin 100 adet tam sayıyı **sayilar.txt** dosyasına yazınız.



1. Oluşturduğunuz **sayilar.txt** dosyasındaki sayıları okuyunuz. Okuduğunuz sayılardan **asal sayı** olanları **sayilar2.txt** dosyasına kaydediniz.



**fread() ve fwrite() Fonksiyonları**

Verileri dosyaya yazma ve dosyadan okuma işlemlerinde aşağıda genel yapıları verilen fread() ve fwrite() fonksiyonlarını da kullanabilirsiniz:

size\_t fread (void \*tampon-bellek, size\_t boyut, size\_t id1, FILE \*fp);

size\_t fwrite (void \*tampon-bellek, size\_t boyut, size\_t id1, FILE \*fp);

Daha önce kullandığımız fprintf() ve fscanf() fonksiyonları dosyadan veri okuma ve dosyaya veri yazma işlemlerinde veriler üzerinde değişim yaparak çalıştığından, fread() ve fwrite() fonksiyonlarını kullanmak daha pratiktir. fprintf() fonksiyonunu kullanarak bir dosyaya sayı yazarken, sayının dosyanın ASCII metnine çevrilmesi gerekir. Yine fscanf() fonksiyonu ile bir dosyadan bir sayı okurken, sayının fscanf() fonksiyonunun dahili format yapısına çevrilmesi gerekir.

fread() fonksiyonu fp ile gösterilen dosyadan okunan boyut yapısında id1 kadar değeri tampon-bellek ile gösterilen belleğe atar. Burada, boyut ifadesi okunan verinin byte olarak değerini, id1 ifadesi ise kaç adet veri okunduğunu belirler. fread() fonksiyonu okunan veri sayısını geri verir. Bu değer 0 ise, herhangi bir veri okunmamış demektir. Bu durumda, ya bir hata olmuştur ya da dosya sonu gelmiştir.

fwrite() fonksiyonu, fread() fonksiyonunun yaptığı işlemin tam tersini yapar. fwrite() fonksiyonu, tampon-bellek ile gösterilen bellekte bulunan boyut yapısında id1 kadar değeri fp ile gösterilen dosyaya yazar. Burada, boyut ifadesi yazılan byte olarak değerini, id1 ifadesi ise kaç adet veri yazıldığını belirler. fwrite() fonksiyonu yazılan veri sayısını geri verir. Sadece bir hata meydana geldiğinde bu değer id1 değerinden az olur.

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <conio.h>

int main ()

{

FILE \*fp;

int idizi[5];

int id1;

for (id1=0; id1<5; id1++) {

printf("int bir değer giriniz: ");

scanf("%d", &idizi[id1]);

}

if ((fp=fopen ("deneme.txt", "wb")) == NULL) {

printf("Dosya açılamadı!\n");

getch();

return 0;

}

if (fwrite(idizi, sizeof (idizi), 1, fp) != 1) {

printf("Yazma hatası!\n");

getch();

return 0;

}

fclose(fp);

if ((fp=fopen("deneme.txt", "rb")) == NULL) {

printf("Dosya açılamadı!\n");

getch();

return 0;

}

printf("\nDeğerleri silinmiş dizi elemanları:\n");

for (id1=0; id1<5; id1++) {

idizi[id1] = 0;

printf("%d ", idizi[id1]);

}

printf("\n");

if (fread(idizi, sizeof idizi, 1, fp) != 1) {

printf("Okuma hatası!\n");

getch();

return 0;

}

fclose(fp);

printf("Dosyadan okunan dizi değerleri:\n");

for (id1=0; id1<5; id1++) printf("%d ", idizi[id1]);

getch();

}

**Rastgele Erişim**

Şimdiye kadar incelediğimiz bütün örneklerde, dosyadan yaptığımız okuma işlemlerini dosyanın başından sonuna doğru bir sıra dahilinde yaptık. Bunun yanında, aşağıda prototipi verilen fseek() fonksiyonunu kullanarak bir dosyanın herhangi bir yerindeki bir bilgiyi okuyabilirsiniz:

int fseek (FILE \*fp, long ara, int yer);

Burada, fp ifadesi işlem yapılan dosyayı, ara ifadesi yer ifadesinin gösterdiği değerin tanımladığı dosya konumundan,işlem yapılmak istenen yerin byte olarak uzaklığını verir. yer ifadesi dosyada arama işleminin başlayacağı yeri gösterir. yer ifadesinin yerine aşağıdaki makrolardan birini kullanmanız gerekir:

SEEK\_SET (0) Aramayı dosya başından başlatır.

SEEK\_CUR (1) Aramayı aktif konumdan başlatır.

SEEK\_END (2) Aramayı dosya sonundan başlatır.

Ayrıca, aşağıda prototipi verilen ftell() fonksiyonunu kullanarak bir dosyanın aktif konumunu öğrenebilirsiniz.

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <conio.h>

int main ()

{

long int lid1, lid2;

int id1;

FILE \*fp1;

if ((fp1=fopen("deneme.txt", "rb")) == NULL) {

printf("Dosya açılamadı!\n");

getch();

return 0;

}

/\* Dosya sonuna ulaşıp dosya boyutunu kaydeder. \*/

fseek (fp1, 0L, SEEK\_END);

lid2 = ftell(fp1);

for ( ; ; ) {

printf("Byte sıra no.sunu giriniz: ");

scanf("%ld", &lid1);

if (lid1>=lid2) break;

if (fseek(fp1, lid1, SEEK\_SET)) {

printf("Arama hatası!");

getch();

return 0;

}

id1 = getc(fp1);

printf("%ld konumundaki değer ASCII %d : %c\n", lid1, id1, id1);

}

fclose(fp1);

getch();

**remove(), rewind() ve fflush() Fonksiyonları**

Aşağıda prototipi verilen remove() fonksiyonunu bir dosyayı silmek için kullanabilirsiniz:

int remove (char \*dosya-adı);

remove() fonksiyonu dosya-adı ifadesi ile gösterilen dosyayı siler. Başarılı bir şekilde sona erdiğinde 0 değerini, aksi takdirde 0 olmayan bir değer geri verir.

Bir dosyanın aktif konumunu dosya başına almak için aşağıda prototipi verilen rewind() fonksiyonunu kullanabilirsiniz:

void rewind (FILE \*fp);

fp ifadesi ile gösterilen dosyanın aktif konum göstergesini dosyanın başına alır. Bu fonksiyon herhangi bir değer geri vermez. Çünkü, başarılı bir şekilde açılan dosyanın aktif konumu başa alınabilir.

C dilinde, bir dosya ile ilgili tampon belleği boşaltmak için aşağıda prototipi verilen fflush() fonksiyonunu kullanabilirsiniz:

int fflush (FILE \*fp);

fflush() fonksiyonu, fp ifadesi ile gösterilen dosyanın tampon belleğini boşaltır. Fonksiyon başarılı olduğu zaman 0 değerini, aksi takdirde EOF değerini geri verir. Eğer fflush() fonksiyonunu argüman olmadan kullanırsanız, mevcut bütün disk tampon bellekleri silinir.

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <conio.h>

#include <ctype.h>

int main ()

{

printf("Dosya silinecek! Devam edecek misiniz (E/H)?");

if (toupper(getche())=='E') remove("deneme.txt");

getch();

}

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <conio.h>

#include <ctype.h>

int main ()

{

FILE \*fp1;

if ((fp1=fopen("deneme.txt", "r")) == NULL) {

printf("Dosya açılamadı!\n");

getch();

return 0;

}

while (!feof(fp1)) putchar(getc(fp1));

rewind(fp1);

printf("\n\n");

while (!feof(fp1)) putchar(getc(fp1));

fclose(fp1);

getch();

}