ITU ACM Student Chapter Course Program

Introduction to C

Week 8

Instructor

Mehmet Yiğit Balık

Prepared by

Mehmet Yiğit Balık & Mihriban Nur Koçak & Emir Oğuz

Dosyalar İle Çalışmak

Şu ana kadar gösterilenler ile yazılan programlarda programın sona ermesi ile birlikte yaratılan bütün veriler kayboluyordu. Oluşturulan verilerin kaydedilmesini ve program sona erse dahi bu verilerin saklanmasını sağlamak için C programlama dilinde **dosyalar** ile çalışılır.

Eğer dosyaya yazılacak çok fazla veri varsa hepsinin yazılması çok zaman alabilmektedir. Fakat içerisinde çok fazla veri yazılı olan dosyanın içeriğine C'deki birkaç fonksiyonu kullanarak rahatlıkla ulaşılabilir.

Kaydedilen bu veriler kolaylıkla başka dijital ortamlara taşınabilir.

C programlama dilinde dosyalarla çalışırken şimdilik üzerinde durulması gereken 2 tip dosya vardır:

- **Text Dosyası**
  - Bilgisayarda uzantısı **.txt** olarak tutulur.
  - İçerisine yazılan veriler insanların kolaylıkla okuyabileceği(human-readable) düz yazı olarak saklanır. Herhangi bir editörde açılarak üzerinde değişiklikler yapılabilir.
  - Binary dosyalarına göre daha güvensizdir
  - Binary dosyalarına göre daha çok yer kaplarlar.

- **Binary Dosyası**
  - Bilgisayarda uzantısı **.bin** olarak tutulur.
  - İçerisine yazılan veriler insan tarafından okunamaz(machine-readable). **0** ve **1** rakamları kullanılarak veriler kaydedilir.
  - Çok yer kaplayacak veriler bu şekilde depolandığında daha az yer kaplar.
  - Text dosyalarına göre daha güvenlidir.

C Dosya İşlemleri (File Operations)

C'de hem Binary dosyasına hem de Text dosyasına uygulanabilecek işlemler:

- Yeni bir dosya yaratma
- Var olan bir dosyayı açma
- Açılan dosyayı kapatma
- Dosyadan veri okuma
- Dosyaya veri yazma

FILE Veri Yapısı

C kodunda bir dosya direkt olarak tutulmaz, bu dosyanı adresi tutulur (**pointer** ile) ve yapılan değişiklikler bu adres üzerinden yapılır. Dosya adresini tutmak için FILE veri yapısına ait bir pointer oluşturulmalıdır.

```
FILE *fptr;
```

Oluşturulan bu **FILE** veri yapısına ait pointer'a bir dosyanın adresini atamak için **fopen** fonksiyonu kullanılır. Yazı düzeni şu şekildedir:

```
fptr = fopen("dosyaAdı","mod");
```

Örnek:

```
fptr= fopen("numbers.txt","r");

fptr= fopen("numbers.bin","wb");
```

Yukarıda görüldüğü üzere **fopen** iki adet parametre alır. Bunlardan ilki işlem uygulanacak olan dosyanın ismidir (uzantısı ile birlikte). İkincisi ise dosyanın hangi modda açılacağıdır. Yukarıdaki örnekte Text dosyası için mod **r** olarak yazılmıştır, bu okuma (**r**ead) modu anlamına gelmektedir. Binary dosyası içinse **wb** modu yazılmıştır, bu bin (binary) yazma (**w**rite **b**in) modudur.

Dosya Açma Modları:

Dosyalar hangi modda açılıyorsa, dosya üzerinde ancak o modun izin verdiği işlemler gerçekleştirilebilir.

- **r**
  - Okuma (**read**) modudur. Yazma işlemleri gerçekleştirilemez.
  - Eğer açılması istenilen dosya yoksa **fopen() NULL** döndürür.
- **rb**
  - Bin (**binary**) okuma modudur.
  - Eğer açılması istenilen dosya **fopen() NULL** döndürür.
- **w**
  - Yazma (**write**) modudur.
  - Eğer dosya halihazırda var ise içeriği silinir ve yeni veriler üzerine yazılır.
  - Eğer dosya halihazırda yoksa yeni bir dosya yaratılır.
- **wb**
  - Bin (**binary**) yazma modudur.
  - Eğer dosya halihazırda var ise içeriği silinir ve yeni veriler üzerine yazılır.
  - Eğer dosya halihazırda yoksa yeni bir dosya yaratılır.
- **a**
  - Ekleme (**append**) modudur.
  - Yeni veriler var olan verilerin sonuna eklenir.
  - Eğer dosya halihazırda yoksa yeni bir dosya yaratılır ve istenilen veriler bu dosyaya yazılır.
- **ab**
  - Bin (**binary**) ekleme modudur.
  - Yeni veriler var olan verilerin sonuna eklenir.
  - Eğer dosya halihazırda yoksa yeni bir dosya yaratılır ve istenilen veriler bu dosyaya yazılır.

Not: *Eğer yukarıdaki modlar sağ tarafına* **+** *işareti konularak kullanılırsa:*

- *Yazma moduna ve ekleme moduna okuma özelliği eklenir (**w+, wb+** & **a+, ab+**).*

- *Okuma moduna yazma özelliği eklenir (**r+, rb+**).*

Açılan her dosya tıpkı dinamik hafızada yer ayırma yöntemiyle oluşturulmuş pointer'lar gibi program sona ermeden kapatılmalıdır. Dosyalar **fclose** fonksiyonu kullanılarak kapatılır.

```
fclose(fptr);
```

Buradaki **fptr** daha önceden **fopen** ile açılan dosyanın adresinin atandığı **FILE** veri yapısına ait pointer'dır.

Text Dosyasında INPUT/OUTPUT

Bir dosyanın nasıl açılacağı ve nasıl kapatılacağı öğrenildiğine göre sırada bu açma ve kapama işlemlerinin arasında, dosya üzerinden yapılacak işlemlerde kullanılacak fonksiyonlar var. Farklı işlevlere sahip bu fonksiyonlar kullanılarak dosya üzerinden birçok input/output işlemi yapılabilir.

INPUT Fonksiyonları:

- **fscanf(FILE *fptr, "%format", &degisken);**
  - Bu fonksiyon kullanıldığında yapılan formatlama sayısı kadar dosyadan veri okunabilir.
  - Eğer formatlama yapılmasına rağmen bir değer okunamaz ise -1 değeri döndürür.
- **fgets(char *degisken, size, FILE *fptr);**
  - Bu fonksiyon formatlama olmaksızın dosyadan bir satırı **string** şeklinde okur ve ilk parametre olarak verilen **string** içine kaydeder. İkinci parametre

okunacak maksimum boyutu ifade eder, kodu yazan kişi tarafından belirlenir**.**

OUTPUT Fonksiyonları:

- **fprintf(FILE *fptr, "%format", degisken);**
  - Bu fonksiyon kullanıldığında yapılan formatlama sayısı kadar dosyaya veri yazılabilir.
- **fputs(const char *degisken, FILE *fptr);**
  - Bu fonksiyon kullanılarak bir **string** dosyaya yeni bir satır şeklinde yazdırılır.

Aşağıdaki örnekte önceden hazırlanmış bir sınıf yoklama listesi üzerinden işlemler yapılacak. Bu text dosyası sınıftaki her öğrenciye ait isim, soyisim, cinsiyet ve numara bilgilerini içeriyor. Kodun amacı yedi öğrencinin bilgilerinin yer aldığı bu listeyi, kız öğrenciler ve erkek öğrencilerin bilgilerini ayırarak ayrı text dosyalarına yazdırmak.

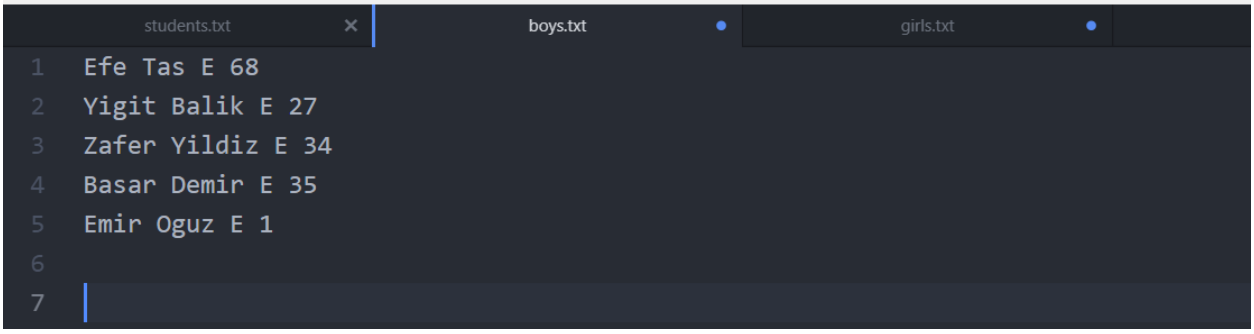
students.txt

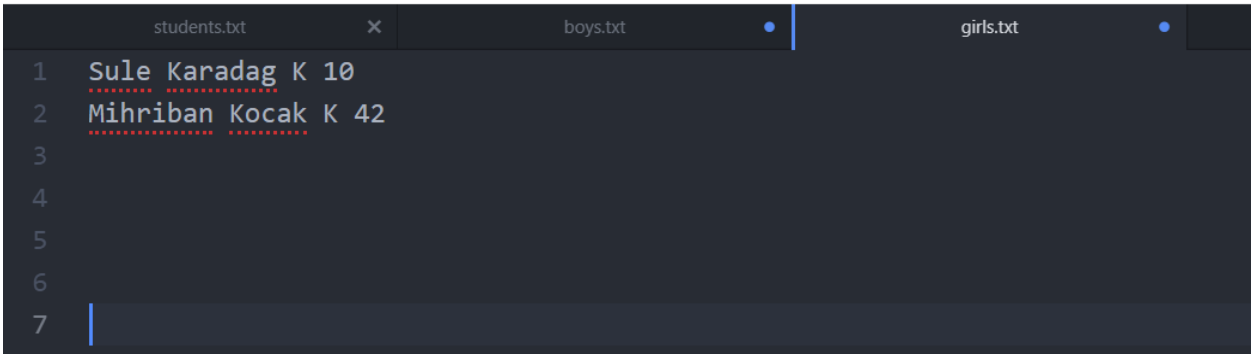
```
1  Efe Tas E 68
2  Yigit Balik E 27
3  Sule Karadag K 10
4  Zafer Yildiz E 34
5  Mihriban Kocak K 42
6  Basar Demir E 35
7  Emir Oguz E 01
```

```c
#include <stdio.h>

#include <stdlib.h>
```

```c
typedef struct
{
    char name[20];
    char surName[20];
    char gender;
    int number;
} STUDENT;


int main()
{
    char *file_name = "students.txt";
    FILE *fptr = fopen(file_name, "r");
    FILE *girlsfp = fopen("girls.txt", "w");
    FILE *boysfp = fopen("boys.txt", "w");
    STUDENT student;


    while (fscanf(fptr, "%s %s %c %d", student.name, student.surName,
&student.gender, &student.number) != -1)
    {
        if (student.gender == 'E')
        {
            fprintf(boysfp, "%s %s %c %d\n", student.name,
student.surName, student.gender, student.number);
        }
        else if (student.gender == 'K')
        {
```

```
            fprintf(girlsfp, "%s %s %c %d\n", student.name,
student.surName, student.gender, student.number);

        }

    }



    fclose(girlsfp);

    fclose(boysfp);

    fclose(fptr);

    return EXIT_SUCCESS;

}
```





Bir başka örnek olarak ise bir text dosyasının içinde verilen sayıları çift, asal ve asal olmayan tek sayılar olmak üzere ayırarak farklı text dosyalarına yazdırma örneği yapılabilir.

Burada istenilen işlemi gerçekleştirmek üzere iki farklı fonksiyon yazılmıştır. Bu iki fonksiyonun da return değeri bir **boolean'**dır. Yani bu iki fonksiyon da true veya false değerlerini döndürürler. Burada *isprime()* fonksiyonu parametre olarak aldığı sayı değeri asal ise **true** değilse **false** döndürür. Bir diğer fonksiyon olan *iseven()* fonksiyonu ise parametre olarak aldığı sayı değeri çift ise **true** değilse **false** döndürür. Text dosyasında yer alan her bir sayı bu iki fonksiyona sırayla gönderildiğinde eğer bir sayı asal ise **prime.txt** dosyasına, eğer asal değil ama çift ise **even.txt** dosyasına, eğer hem çift değil hem asal değilse **odd.txt** dosyasına yazdırılıyor.

```c
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>

bool iseven(const int num);
bool isprime(const int num);



int main(){
    char *file_name = "numbers.txt";

    FILE *evenfp = fopen("even.txt","w");
    FILE *oddfp = fopen("odd.txt","w");
    FILE *primefp = fopen("prime.txt","w");
    FILE *fp = fopen(file_name,"r");

    int num = 0;
    while(fscanf(fp,"%d",&num) != -1){
        if(isprime(num)){
```

```
            fprintf(primefp, "%d\n", num);
        }

        else if(iseven(num)){
            fprintf(evenfp, "%d\n", num);
        }

        else{
            fprintf(oddfp, "%d\n", num);
        }
    }
    fclose(primefp);
    fclose(evenfp);
    fclose(oddfp);
    fclose(fp);
    return EXIT_SUCCESS;
}



bool iseven(const int num){
    if(num % 2 == 0){
        return true;
    }
    return false;
}
bool isprime(const int num){
    for(int i = 2;i < abs(num);i++){
```

```
        if(num % i == 0){
            return false;
        }
    }
    return true;
}
```

numbers.txt | even.txt | odd.txt | prime.txt

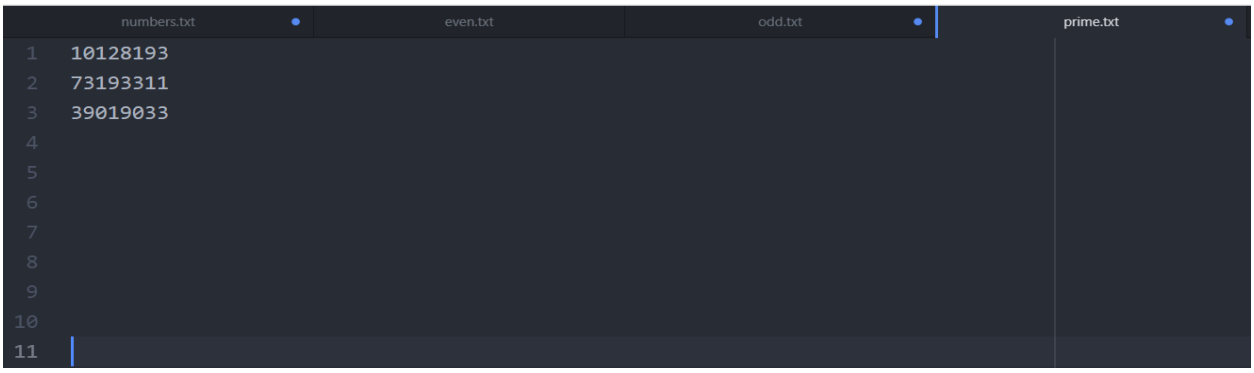
```
73771782  81296771  79982326  75332246  10128193
81643413  76259734  94432076  50063976  91748657
42311916  -1920042  90747362  53851612  43498487
73193311  96685173  39019033  8630045 59322952
```

numbers.txt | even.txt | odd.txt | prime.txt

```
73771782
79982326
75332246
76259734
94432076
50063976
42311916
-1920042
90747362
53851612
59322952
```

numbers.txt | even.txt | odd.txt | prime.txt

```
81296771
81643413
91748657
43498487
96685173
8630045
```



Binary Dosyasında INPUT/OUTPUT

INPUT Fonksiyonu:

- **fread(**degiskenAdresi,degiskeninBuyuklugu,degiskenAdedi,dosyaPointer**);**
  - İlk parametre Binary dosyasından alınacak verinin atanacağı değişkenin adresidir.
  - İkinci parametre bu verinin büyüklüğüdür.
  - Üçüncü parametre bu verilerin sayısıdır.
  - Dördüncü parametre veri alınacak olan bin dosyasının **pointer**'ıdır.

- **fwrite(**degiskenAdresi,degiskeninBuyuklugu,degiskenAdedi,dosyaPointer**);**
  - İlk parametre Binary dosyasına yazılacak verinin adresidir.
  - İkinci parametre bu verinin büyüklüğüdür.
  - Üçüncü parametre bu verilerin sayısıdır.
  - Dördüncü parametre verinin yazıldığı bin dosyasının **pointer**'ıdır.