Dosyadan Okuma Yapmaya İlişkin İpuçları

- %c harici tanımlayıcılarla yapılan okumalarda beyaz boşluk karakterleri (boşluk, satı sonu, tab, vb.) atlanır. Bu karakterler %c ile yapılan okumalarda atlanmaz. Örneğin:

```
/*fPtr il gosterilen dosyanın icerigi: “10  20

30                  40 “
10 ile 20 arasinda 2 bosluk var*/

Ör1:
for(i=0; i<3; i++){
        fscanf(fPtr, “%d”, &num);
        printf(“%d ”, num);
}
/*Output: “10 20 30 “*/

Ör2:
for(i=0; i<5; i++){
        fscanf(fPtr, “%c”, &ch); /*tirnak icerisinde bosluk yok*/
        printf(“%c”, ch);
}
/*Output: “10  2”*/
```

- %c ile yapılan okumalarda beyaz boşluk karakterlerinin atlanması isteniyorsa, format stringinde bu karakterlerin atlanması istenilen bölgeye bir boşluk bırakılır. Bu durumda o bölgedeki varsa bütün beyaz boşluk karakterleri atlanır. Mesela aynı dosya için:

```
Ör1:
for(i=0; i<5; i++){
        fscanf(fPtr, “ %c”, &ch);
        printf(“%c”, ch);
}
/*Output: “10203”*/

Ör2:
fscanf(fPtr, “%d %c%c%d%c %c %c”, n1, c1, c2, n2, c3, c4, c5);
printf(“%d\n%c\n%c\n%d\n%c\n%c\n%c”, n1, c1, c2, n2, c3, c4, c5);
/*Output:
10
2
0
30

4
0 */
```

Bu özellik konsoldan okunan verideki beyaz boşluk karakterlerinin çıkarılması için de kullanılabilir.

- fscanf fonksiyonu okuduğu nesne sayısını döndürür. Belirtilen tipte nesne okuyamıyorsa (dosya sonu ve hata durumu hariç) 0 döndürür. Dosya sonuna gelindikten **sonra** okuma yapılıyorsa veya okuma hatası oluşmuşsa EOF (end of file) döndürür:

```
int status;
…
status = fscanf(fPtr, "…", &var1, &var2, …);
/*status u kontrol ediyoruz, var1, var2… 'yi degil*/
if(status==EOF)
        printf("End of file is reached and one more read operation is performed\n");
```

- Dikkat edin; dosyadaki son değer okunurken EOF döndürülmez, daha sonraki okumalarda döndürülür. Masala yukarıdaki dosya için:

```
status=0;
while(status != EOF){
        status = fscanf(fPtr, "%d", &num);
        printf("%d ", num);
}
/*Output: "10 20 30 40 40"*/
```

Son değer okunduğunda fscanf başarılı olarak bir değer okuduğu için 1 döndürür. Dolayısıyla döngüden çıkılmaz. Ancak bir sonraki seferde fscanf dosya sonu olduğu için değer okuyamaz ve EOF döndürür. Değer okunmadığı için fscanf num'ın içeriğini değiştirmez ve eski değer tekrar basılır. status EOF'a eşit olduğu için döngüden çıkılır.
Problem, ancak fazladan bir okuma yapıldığında EOF'un tespit edilebilmesi. Bu sorun aşağıdaki algoritmalarla çözülebilir:

- Algorithm A:
  While EOF is not detected
    Read data
    If EOF is not detected
      Consume the last read data

- Algorithm B (works faster):
  Read data
  While EOF is not detected
    Consume the last read data
    Read data

Ör:

```
status = fscanf(fPtr, "%d", &num); /*read first number*/
while(status != EOF){
        printf("%d ", num); /*consume data*/
        status = fscanf(fPtr, "%d", &num) /*read next number*/
}
/*Output: "10 20 30 40"*/
```

Veri okuma ve EOF kontrolünü tek satırda aşağıdaki gibi birleştirmek de mümkün:

```
while((status = fscanf(fPtr, "%d", &num)) != EOF)
        printf("%d ", num);
/*Output: "10 20 30 40"*/
```

- fscanf(), dosya sonunda ve bir okuma hatası olduğunda EOF döndürür. EOF döndürme nedeninin anlaşılabilmesi için C standartlarında 2 fonksiyon mevcuttur:
  - feof(FILE* fPtr): Normalde 0, dosya sonuna ulaşıldığında 0' dan farklı bir değer döndürür
  - ferror(FILE* fPtr): Normalde 0, hata durumunda 0'dan farklı bir değer döndürür.

Ör:

```
while((status = fscanf(fPtr, "%d", &num)) != EOF)
        printf("%d ", num);
if(feof(fPtr))
        printf("\nEnd of file is reached.");
else
        printf("\nAn I/O error occured.");
```

feof() -yukarıdaki gibi- fscanf()'in niçin EOF döndürdüğünün anlaşılması içindir; dosyadan okumada döngü şartı olarak kullanılmamalı. Aksi taktirde:

- Bir I/O hatası oluşursa sonsuz döngü meydana gelir
- Bazı durumlarda istikrarsız sonuç üretebilir

- Sayısal değer okunurken, beyaz boşluk karakterleri dışında sayısal olmayan bir karaktere rastlanırsa okuma gerçekleşmez, 0 döndürülür, file pointer ilerlemez. Mesala:

```
/*fPtr ile gosterilen dosyanın icerigi: "10 20e 30"*/

while(status != EOF){
        status = fscanf(fPtr, "%d", &num)
        printf("%d ", num);
}

/*Kavramsal olarak cikis (sonsuz dongu nedeniyle pratikte farklilik gosterebilir):
"10 20 20 20 20 …"*/
```

Yukarıdaki kod 20 okunduktan sonra file pointer ilerleyemediği için sonsuz döngüye girer.

- Dosya pointerının dosyanın başını yada herhangi başka bir konumu göstermesini sağlayan C fonksiyonları var. Biz bu fonksiyonları pek incelemeyeceğiz. Aksi belirtilmemişse bu fonksiyonları araştırıp ödevlerinizde kullanabilirsiniz.