GİRİŞ ÇIKIŞ İŞLEMLERİ

Modüler Programlama

C programlama dili, yapısal olmasının yanında aynı zamanda modüler bir programlama dilidir. Yapısal programlamanın çerçevesi *Yapısal Programlama* altında anlatılmıştır. Yapısal programlamada yazılan fonksiyonlar gruplar halinde başka kod dosyalarına konulur ve bun bileşenler gerektikçe projemize `#include` ön işlemci yönergeleriyle (preprocessor directive) dahil edilir.

Modüllere ayırmanın soyutlama (abstraction), değişim yönetimi (change management) ve yeniden kullanım (reusing) gibi üstünlükleri vardır. Modüllere ayrılan bir kodun bakımı da daha kolaydır. Ancak daha fazla fonksiyonun çağrılması, işlemciyi yorar. Çünkü her fonksiyon çağrısında işlemcinin mevcut durumu ve kaydediciler belleğe itilir ve fonksiyondan geri dönüldüğünde eski duruma dönmek içi bu veriler tekrar geri çekilir. İşlemciye yüklenen bu çağırma yükü (calling overhead) günümüz bilgisayarlarında oldukça ihmal edilebilir seviyededir. Ancak milisaniyeler bazında yapılması gereken bir iş varsa böyle zaman kritik işler gerçekleştirmek için bu durum göz önüne alınmalıdır.

C programlama dilinde en çok kullanacağımız modüllerden biri de standart giriş çıkış işlemlerinin (input output-IO) tanımlı olduğu `stdio.h` başlık (header) dosyasıdır. Bu modülü kodumuza dahil etmek (include) için kodu yazdığımız metin editörünüzde kaynak kodun başına `#include` ön işlemci yönergesini (preprocessor directive) aşağıdaki şekilde ekleriz.

```
#include <stdio.h>
```

Böylece konsola (konsol kelimesi UNIX/Linux işletim sitemindeki terminal ya da Windows işletim sistemindeki komut satırını ifade eder) bir şey yazmak için `printf` veya klavyeden bir şey okumak ve bir değişkene atamak için ise `scanf` fonksiyonlarını kullanabilir hale geliriz.

C programlama dilinde her işletim siteminde çalışan standart birçok başlık dosyası bulunmaktadır. Ancak bunların dışında çalıştığı işletim sistemine özel olan başlık dosyaları da bulunmaktadır;

Başlık Dosyası	Açıklama
`stdio.h`	Standart giriş-çıkış komutları
`math.h`	Matematiksel fonksiyonlar
`stdlib.h`	Dönüşüm, sıralama, arama vb. komutları
`string.h`	Alfa sayısal ve bazı bellek yönetim komutları
`curses.h`	curses, metin terminali ekranında imleç (cursor) denetimini destekleyen eski bir Unix/Linux kütüphanesidir.
`conio.h`	DOS/Windows komut satırında imleç (cursor) denetimini destekleyen kütüphanedir

Tablo 15. En çok kullanılan başlık dosyaları

Konsola Biçimlendirilmiş Veri Yazma

C programlama dilinde değişkenlerimizde olan veriyi biçimlendirilmiş (formatted) bir şekilde imlecin bulunduğu noktadan itibaren konsola yazabiliriz (print). Bunu sağlayan fonksiyon `stdio.h` başlık dosyasındaki `printf` fonksiyonudur.

Bir fonksiyonun hangi parametreleri aldığı ve neyi geri döndürdüğünü prototipinden (prototype) anlarız. Aşağıda `printf` fonksiyonunun prototipi verilmiştir;

```
void printf (char *format, ... );
```

Bu fonksiyon bir değişmez metni (string literal) ilk olarak parametre olarak alır. Metinlere ilişkin işlemler ileride *DİZGİLER* başlığında anlatılacaktır. Değişmez metinler C dilinde çift tırnak (") karakterleri arasında yazılır. İlk parametre olan bu biçimlendirme metni aynı zamanda kendisinden sonra gelecek parametrelerin de hangi biçimde konsola yazılacağını belirler. Her bir parametrenin

konsola nasıl yazılacağını biçim belirleyicisi (format specifier) belirler[15]. Biçim belirleyicileri iki karakter olup ilk karakteri yüzde (`%`) karakteridir.

```
/* Bu program, printf örneğidir. */
#include <stdio.h>
int main() {
   int yas=50;
   float kilo=100.0;
   printf("Merhaba!");  /* Bu printf fonksiyonunu sadece değişmez metin
                           içeren tek argümanla çağırmıştır. İmlecin
                           bulunduğu yerden itibaren
                           "Merhaba!" yazar ve imleci metnin sonuna taşır.*/

   printf("Yaşınız:%d",yas); /* Bu printf fonksiyonunu içinde
                                biçimlendirme karakteri olan
                                değişmez metin sonrasında yas değişkeniyle
                                birlikte iki argümanla çağırmıştır. İmlecin
                                son konumundan başlayarak
                                "Yaşınız:50" yazar ve imleci metnin
                                sonuna taşır. */
   printf("Kilonuz:%f",kilo); /* Bu printf fonksiyonunu içinde biçimlendirme
                                 karakteri olan değişmez metin ve izleyen
                                 kilo değişkeniyle birlikte iki
                                 argümanla çağırmıştır. İmlecin son
                                 konumundan başlayarak
                                 "Kilonuz:100.000000" yazar ve imleci
                                 metnin sonuna taşır.*/
   return 0;
}
/*Program Çıktısı:
Merhaba!Yaşınız:50Kilonuz:100.000000

...Program finished with exit code 0
*/
```

Görüldüğü üzere biçimlendirme metni içindeki `%d` ile ifade edilmiş biçim belirleyicisi izleyen argüman olan `yas` değişkeni ondalık (decimal) olarak konsola yazdırılmıştır. Burada dikkat edilmesi gereken `yas` argümanının, `%d` yani ondalık olarak biçimlendirilecek bir veri olmasıdır. Benzer şekilde en sonraki talimatta biçimlendirme metni içindeki `%f` ile ifade edilmiş biçim belirleyicisi izleyen argüman olan `kilo` değişkeni kayan noktalı sayı (float) olarak konsola yazdırılmıştır.

Biçim Karakterleri	Biçimlendirme metni sonrasındaki parametrelerin biçimi
%d	Konsola ondalık sayı olarak yazılır
%c	Konsola karakter olarak yazılır.
%f	Konsola kayan noktalı sayı olarak yazılır.
%x	Konsola onaltılık (hexadecimal) sayı olarak yazılır.
%s	Konsola metin olarak yazılır.
%%	Konsola % karakteri yazılır.

Tablo 16. Biçim Belirleyiciler

Eğer konsola `%` karakteri yazmak istersek biçimlendirme olarak iki `%` karakteri kullanmalıyız. Buna ilişkin örnek aşağıda verilmiştir.

```
/* Bu program, printf ile konsola % karakteri basma örneğidir. */
#include <stdio.h>
int main() {
   int doluluk=80;
   printf("Doluluk Oranı: %%%d",doluluk);
   return 0;
```

[15] https://www.tutorialspoint.com/cprogramming/c_format_specifiers.htm

```
}
/*Program Çıktısı:
Doluluk Oranı: %80

...Program finished with exit code 0
*/
```

Bu başlık altında şu ana kadar tem bir parametreyi biçimlendirilmiş olarak konsola yazmayı öğrendik. Birden çok parametreye örnek olarak aşağıdaki örnek verilebilir.

```
/* Bu program, çok argümanlı printf örneğidir. */
#include <stdio.h>

int main() {
   char yas=45;
   float kilo=81.0;
   int doluluk=80;
   printf("Yaş: %d, Kilo: %f, Doluluk Oranı: %%%d",yas,kilo,doluluk);
   return 0;
}
/* Program Çıktısı:
Yaş: 45, Kilo: 81.000000, Doluluk Oranı: %80

...Program finished with exit code 0
*/
```

Format metnini imleci hareket ettirecek şekilde biçimlendirebiliriz. Yukarıdaki örneğin çıktısı göz önüne alındığında, biçimlendirme metni içinde imlecin satır başına geçmesini sağlamak için `'\n'` şeklinde kaçış tuşu dizisi (escape sequence) kullanılabilir. Bu diziler aşağıda verilmiştir;

Kaçış tuşu dizisi	Açıklama
\n	Konsolda imleç yeni satırın (new line) başına hareket eder.
\t	İmlece, metin yazarken sekme (tab) tuşuna basında olan hareketi yapar. Yani imleç, konsolda 8 in katları olan sütuna hareket eder.
\r	Konsolda imleç bulunduğu satırın başına (carriage return) hareket eder.
\a	Konsol, uyarı (alert) sesi çıkarır.
\\	Konsola ters bölü karakterini yazar.
\"	Konsola çift tırnak karakterini yazar.
\'	Konsola tek tırnak karakterini yazar.

Tablo 17. Kaçış Tuşu Dizileri

Aşağıda kaçış tuşu dizilerine ilişkin örnek program verilmiştir.

```
/* Bu program, escape sequence printf örneğidir. */
#include <stdio.h>
int main() {
   printf("ILHAN\n"); // ILHAN yazılarak konsolda satır başı yapılır.
   printf("OZKAN\r"); // OZKAN yazılarak aynı satırın başına dönülür.

   printf("ABC\tDEF\n");  /* ABC yazılır ve 8. sütuna ilerlenir ve
                            sonra DEF yazılır, sonrasında satır başı
                            yapılır. TAB karakteri 8 ve katları olan
                            sütunlara imleci ilerletir.*/
   printf("\"İyi\"-\"Kötü\""); // Çift tırnak içeren "İyi"-"Kötü" yazılır.
   return 0;
}
/* Program Çıktısı:
ILHAN
ABCAN   DEF
"İyi"-"Kötü"
```

```
...Program finished with exit code 0
*/
```

Biçim belirleyiciler (format specifier) ile daha fazla biçimlendirme yapılabilir. Bu durumda biçimlendirme için aşağıdaki desen kullanılır;

```
%[önek][genişlik][.ondalık]<biçim karakteri>
```

Burada;

1. Genişlik, konsola kaç karakter genişliğinde yazdırılacağını belirler.
2. Ondalık, konsola kayan noktalı sayı yazılacak ise noktadan sonra kaç rakam yazdırılacağını belirler.
3. Önek içerisinde aşağıdaki karakterler olabilir;
   a. `+` karakteri: Sayının işaretini konsola yazılmasını sağlar.
   b. `-` karakteri: Yazdırılacak parametrenin belirtilen genişliğin soluna yaslanarak yazdırılmasını sağlar.
   c. `0` karakteri: Yazdırılacak sayının soluna belirtilen genişliğe bağlı olarak `0` karakteri konulmasını sağlar.

Aşağıda 10 karakterlik genişlikte PI sayısının biçimlendirme örnekleri verilmiştir.

```
/* Bu program, detaylı biçimlendirme printf örneğidir. */
#include <stdio.h>
int main() {
   const float PI=3.141527;
   printf("1234567890\n");
   printf("%.2f\n",PI);
   printf("%3.f\n",PI);
   printf("%10.2f\n",PI);
   printf("%+10.3f\n",PI);
   printf("%+10.4f\n",PI);
   printf("%010.4f\n",PI);
   printf("%-10.4f\n",PI);
   printf("%-+10.2f\n",PI);
   printf("%-+10.2f\n",-PI);
   printf("%+10.2f\n",-PI);
   printf("%0+10.2f\n",-PI);
   return 0;
}
/*Program Çıktısı:
1234567890
3.14
  3
      3.14
    +3.142
   +3.1415
00003.1415
3.1415
+3.14
-3.14
     -3.14
-000003.14

...Program finished with exit code 0
*/
```

Klavyeden Biçimlendirilmiş Veri Okuma

C programlama dilinde değişkenlerimizde klavyeden biçimlendirilmiş (formatted) olarak girilen veriyi tarayıp (scan), veriyi tutan değişkenlere koyabiliriz. Bunu sağlayan fonksiyon `stdio.h` başlık dosyasındaki `scanf` fonksiyonudur. Bu `scanf` fonksiyonunun prototipi verilmiştir;

```
void scanf (char *format, ... );
```

Bu fonksiyon da bir metni ilk olarak parametre olarak alır. İlk parametre olan bu biçimlendirme metni aynı zamanda kendisinden sonra gelecek parametrelerin de hangi biçimde klavyeden okunacağını belirler. Her bir parametrenin klavyeden nasıl okunacağını `printf` fonksiyonundaki gibi biçim belirleyicisi (format specifier) belirler. Farklı olarak, okunan değerlerin konulacağı değişkenlerin adresleri parametre olarak verilir. Bir değişkenin adresinin & işleci ile elde edileceği *Tekli İşleçler* altında anlatılmıştı.

Aşağıda kapasite oranını okuyup % olarak ekrana yazan bir program örneği verilmiştir.

```
/* Bu program, detaylı biçimlendirme scanf örneğidir. */
#include <stdio.h>
int main() {
   float kapasiteOrani;
   printf("Kapasite Oranını (0.00-1.00) Giriniz:");
   scanf("%f",&kapasiteOrani);
   printf("Girilen Kapasite Oranı: %.2f\n",kapasiteOrani);
   printf("%% olarak Girilen Kapasite Oranı: %%%2.2f", 100*kapasiteOrani);
   return 0;
}
/*Program Çıktısı:
Kapasite Oranını (0.00-1.00) Giriniz:0.75
Girilen Kapasite Oranı: 0.75
% olarak Girilen Kapasite Oranı: %75.00

...Program finished with exit code 0
*/
```

Benzer şekilde yarıçapı tamsayı olarak klavyeden alınan bir çemberin ve çevresini hesaplayan program aşağıda verilmiştir.

```
/* Bu program, detaylı biçimlendirme scanf örneğidir. */
#include <stdio.h>
int main() {
   int yaricap;
   float cevre;
   printf("Çemberin yarıçapını tamsayı olarak giriniz:");
   scanf("%d",&yaricap);
   cevre=2*3.1415*yaricap;
   printf("Yarıçapı %d olan çemberin çevresi: %.2f",yaricap,cevre);
   return 0;
}
/*Program Çıktısı:
Çemberin yarıçapını tamsayı olarak giriniz:4
Yarıçapı 4 olan çemberin çevresi: 25.13

...Program finished with exit code 0
*/
```