



ITU ACM Student Chapter Course Program

Introduction to C

Week 1

Instructor

Ahmet Furkan Kavraz

Prepared by

Mehmet Yiğit Balık & Mihriban Nur Koçak & Emir Oğuz

Program

Bir bilgisayarı oluşturan iki yapı taşı vardır. Bunlar **donanım** ve **yazılımdır**. Bir bilgisayar sadece donanımsal olarak çalışmaz çünkü donanım, bir beyni yani yazılımı olmadan ne yapması gerektiğini bilemez. Bilgisayar donanımına ne yapması gerektiğini söyleyen komutlar dizisine program denir.

Programlama Dili

Programlama dili bilgisayarda çözülmesi gereken bir sorun için çözümün insanlar tarafından adım adım yazılmasını sağlayan ve dışına çıkılamayacak kuralları olan tanımlar kümesidir.

Neden C ?

- C, orta sınıf programlama dilleri arasında yer alan hızlı bir dildir.
- C ile işletim sistemi, derleyici ve aygıt sürücülerini dahil olmak üzere pek çok program yapılabilir.
- C taşınabilir bir dildir. Yazılan kod neredeyse hiçbir değişikliğe uğratılmadan başka işletim sistemlerinde çalıştırılabilir.
- C yapısal bir dildir. C kodları fonksiyon olarak adlandırılan alt programlardan oluşmuştur.

Programlama dillerinin en temelde 3 görevi vardır. Bunlar bilgi almak (**INPUT**), bilgiyi işlemek (**PROCESSING**) ve işlenmiş bilgiyi sunmaktır (**OUTPUT**).

C dilinde öğrenmeye OUTPUT ile başlamak daha rahat bir başlangıç yapılmasını sağlar.

printf() Fonksiyonu

printf() fonksiyonu, programlamada en temel işlem olan ekrana çıktı verme (**OUTPUT**) işlemini C programlama dili kullanılarak yapılmasını sağlar. İçerisine yazılan bir matematik işlemi veya bir yazı ekranda bu fonksiyon ile gösterilebilir. Yazının (**string**) " " işaretleri arasına yazılması gerekmektedir. Örnek olarak meşhur *Hello World* verilebilir.

```
#include <stdio.h>

int main() {
    printf("Hello World!\n");
    return 0;
}
```

Burada öncelikle yazı düzeni (syntax) ve terminal komutlarından bahsetmekte fayda var:

- C yazmak isteyen bir kişi kullanmak istediği ve bu eğitimde daha sonra anlatılacak olan kütüphaneleri kodun en üstüne **#include <kütüphane.h>** şeklinde yazmalıdır.
- Yine daha sonra anlatılacak olan **int main(){...kod}** kısmı her C kodunda bulunması gereken ana bölümdür.
- **return 0;** ise kodun en sonunda kullanılarak programın başarılı bir şekilde sona ermesini sağlar.
- İki süslü parantez arasına yazılan neredeyse her satırın sonuna noktalı virgöl konulmalıdır (;).
- Hello World kelimesinin yanına yazılan **\n** *enter tuşu* işlevi görür ve yeni bir satır koyar.

```
root@MSI: /mnt/c/Users/mybal/Desktop/C course/weeks/Introduction
root@MSI:/mnt/c/Users/mybal/Desktop/C course/weeks/Introduction# gcc -std=c99 -Wall -Werror week00.c -o week00
root@MSI:/mnt/c/Users/mybal/Desktop/C course/weeks/Introduction# ./week00
Hello World!
root@MSI:/mnt/c/Users/mybal/Desktop/C course/weeks/Introduction#
```

Yazılan C kodunu derlemek için ise terminale

`gcc -std=c99 -Wall -Werror week00.c -o week00` komutunu yazılmalıdır.

- **gcc** bir C kodu derleyicisidir.
- **-std=c99** yazdığımız C kodunun 99 standartlarına göre derlenmesini sağlar.
- **-Wall** derleyicinin tüm uyarı mesajlarını aktifleştirir.
- **-Werror** bütün uyarıların hata olarak gözükmelerini sağlar.

Not: *-Wall -Werror daha iyi bir performans için her zaman kullanılmalıdır.*

- **week00.c** kodunun yer aldığı C dosyasıdır.
- **-o** uygulamanın oluşmasını sağlar.
- **week00** ise uygulamaya verilen isimdir.

Bu komut yazıldıktan sonra kod derlenmiş olur. Uygulamayı çalıştırmak için ise:

`./week00` komutu yazılmalıdır.

- **./** sağ tarafına yazılan uygulamayı çalıştırır.

Veri Tipleri

C dilinde 4 adet **temel** veri tipi vardır:

int (Integer)

Tam sayıları temsil eder. Hafızada 4 byte yer kaplar. Integer için formatlama %d şeklinde yapılır, **d** harfi *decimal* kelimesinin ilk harfine karşılık gelir.

Not: % işareti formatlama olduğunu belirtir.

float (Float)

Maksimum 7 rakamdan oluşan reel sayıları temsil eder. Hafızada 4 byte yer kaplar. Float için formatlama %f şeklinde yapılır, **f** harfi *float* kelimesinin ilk harfine karşılık gelir.

double (Long Float)

Maksimum 15 rakamdan oluşan reel sayıları temsil eder. Hafızada 8 byte yer kaplar. Double için formatlama %lf şeklinde yapılır, **lf** *long float* kelimelerinin ilk harflerine karşılık gelir.

Not: Araya konan virgülden sonra çıktısı alınmak istenen sayı veya o sayıya karşılık gelen değişken -ileride işlenecektir- yazılır

Not: Ondalıklı sayılar için virgülden önce veya sonra kaç rakamın yazdırılacağı kullanıcı tarafından belirlenebilir. Bu işlem için

- %.2lf virgülden sonra sadece iki rakam yazdırılmak istendiğini belirtir
- %3.lf virgülden önce sadece üç rakam yazdırılmak istendiğini belirtir
- eğer yazdırılmak istenen kadar basamak yoksa, olmayan basamaklar yerine 0 yazdırılır

char (Character)

Bir harfi temsil eder. Hafızada 1 byte yer kaplar. Char için formatlama **%c** şeklinde yapılır, **c** *character* kelimesinin ilk harfine karşılık gelir. Char çıktısı alınmak istendiğinde diğer veri tipleri gibi virgülden sonra direkt yazılmaz. " (iki tane **tek tırnak**) işaretleri arasına yazılır.

Kod içerisinde, kodu yazan kişinin kendisine ve başkalarına not bırakmak veya bir şeyi açıklamak için yorum yazması gerekebilir bunun için **//** (iki eğik çizgi) kullanılmaktadır. Bir satırda **//** koyduktan sonra yazılan şeyler derleyici tarafından derlenmemektedir ve kod olarak algılanmamaktadır.

Aynı zamanda eğer bir paragrafın derleyici tarafından yorum olarak algılanması isteniyorsa, bu paragrafın başına ve sonuna sırasıyla **/*** ve ***/** işaretleri konulur. Bu sayede çok satırlı yorum elde edilmiş olur.

/* Çok
satırlı
yorum ***/**

```
#include <stdio.h>

int main(){
    printf("%d\n",3 + 4); // tam sayılar için formatlama
    printf("%f\n",3.14); // ondalikli sayılar için formatlama
    printf("%lf\n",3.14); // uzun ondalikli sayılar için formatlama
    printf("%c\n",'a'); // harfler için formatlama
    return 0;
}
```

```
root@MSI:/mnt/c/Users/mybal/Desktop/C course/weeks# gcc -std=c99 -Wall -Werror week01.c -o week01
root@MSI:/mnt/c/Users/mybal/Desktop/C course/weeks# ./week01
7
3.140000
3.140000
a
root@MSI:/mnt/c/Users/mybal/Desktop/C course/weeks#
```

Eğer bir veri kendi formatından başka bir formatla bastırılmaya çalışılırsa

-Wall -Werror sayesinde terminal hata verecektir.

Buraya kadar öğrenilenler artık değişkenler üretilerek yapılabilir.

Değişkenler ve Sabitler

Değişkenler, **belirtilen** bir veri tipi elemanın değerinin tutulmasını sağlar. Değişkene bir değer atamak matematikteki x değişkenini bir sayıya eşitlemek gibidir.

Belirtmekten kasıt, bir değişken oluşturulduğunda o değişkenin hangi veri tipi elemanını tutacağını değişken isminden önce belirtmektir.

Bir önceki örnek bu sefer değişkenler kullanılarak yapılabilir.

```
#include <stdio.h>

int main(){
    int tam_sayi_1 = 3;
    int tam_sayi_2 = 4;
    float ondalikli = 3.14;
    double uzun_ondalikli = 3.14;
    char benim_harfim = 'a';
    printf("1. tam sayi: %d\n", tam_sayi_1);
    printf("2. tam sayi: %d\n", tam_sayi_2);
    printf("Tam sayi toplami: %d\n", tam_sayi_1 + tam_sayi_2);
    printf("Ondalikli sayi: %d\n", ondalikli);
    printf("Uzun ondalikli sayi: %d\n", uzun_ondalikli);
    printf("Harf: %c\n", benim_harfim);
    return 0;
}
```

```
root@MSI: /mnt/c/Users/mybal/Desktop/C course/weeks
a
root@MSI:/mnt/c/Users/mybal/Desktop/C course/weeks# gcc -std=c99 -Wall -Werror week01.c -o
week01
root@MSI:/mnt/c/Users/mybal/Desktop/C course/weeks# ./week01
1. tam sayi: 3
2. tam sayi: 4
Tam sayi toplami: 7
Ondalikli sayi: 3.140000
Uzun ondalikli sayi: 3.140000
Harf: a
root@MSI:/mnt/c/Users/mybal/Desktop/C course/weeks#
```

Sabitler, değişkenler gibi olup eşitlendikten sonra bir daha değiştirilemezler. Sabit oluşturmak için veri tipini belirtmeden önce sabit olduğunun belirtilmesi gerekir. Bunun için **const** anahtar kelimesi kullanılır.

```
#include <stdio.h>

int main(){
    int degisken_1 = 3;
    int degisken_2 = 4;
    int degisken_3 = 2;
    degisken_3 = degisken_1 + degisken_2;
    printf("degisken_3 degeri: %d\n",degisken_3);
    return 0;
}
```

```
root@MSI: /mnt/c/Users/mybal/Desktop/C course/weeks
root@MSI:/mnt/c/Users/mybal/Desktop/C course/weeks# gcc -std=c99 -Wall -Werror week01.c -o
week01
root@MSI:/mnt/c/Users/mybal/Desktop/C course/weeks# ./week01
degisken_3 degeri: 7
root@MSI:/mnt/c/Users/mybal/Desktop/C course/weeks#
```



```
#include <stdio.h>

int main(){
    const int sabit_1 = 3;
    const int sabit_2 = 4;
    const int sabit_3 = 2;
    sabit_3 = sabit_1 + sabit_2;
    printf("sabit_3 degeri: %d\n",sabit_3);
    return 0;
}
```

```
root@MSI: /mnt/c/Users/mybal/Desktop/C course/weeks
root@MSI:/mnt/c/Users/mybal/Desktop/C course/weeks# gcc -std=c99 -Wall -Werror week01.c -o
week01
week01.c: In function 'main':
week01.c:9:11: error: assignment of read-only variable 'sabit_3'
   9 |     sabit_3 = sabit_1 + sabit_2;
     |           ^
root@MSI:/mnt/c/Users/mybal/Desktop/C course/weeks#
```

Fotoğraflarda görüldüğü üzere iki değişken toplama işlemi ile üçüncü bir değişkene atanabilir. Fakat aynı işlem sabitler ile denendiğinde derleyici hata vermektedir çünkü yukarıda bahsedildiği gibi sabitlere bir değer atandıktan sonra o değer değiştirilemez.

scanf() Fonksiyonu

Programlama dillerinin diğer bir temel özelliği de kullanıcılardan girdi (INPUT) alınmasıdır. C'de bu **scanf** fonksiyonu ile yapılmaktadır.

scanf fonksiyonu ile bir girdi alınırken, her şeyde olduğu gibi bu girdiler de bir veri tipine sahiptir. Bunun için **scanf** fonksiyonunda da **printf** fonksiyonundaki gibi formatlama yapılması gerekmektedir.

printf fonksiyonundan farklı olarak virgülden sonra girdinin atanacağı değişkenin sol tarafına **&** işareti konulmalıdır, eğer bu işaret konulmazsa derleyici hata verecektir. Bunun nedeni daha sonra anlatılacaktır.

```
#include <stdio.h>

int main() {
    //degiskenler ilk tanimlandiklarinda bir degere esitlenmek zorunda degil
    dirler
    int girdi_tam_sayi;
    float girdi_ondalikli;
    double girdi_uzun_ondalikli;
    char girdi_harf;
    printf("Harf giriniz: ");
    scanf(" %c",&girdi_harf);
    printf("Tam sayi giriniz: ");
    scanf("%d",&girdi_tam_sayi);
    printf("Ondalikli sayi giriniz: ");
    scanf("%f",&girdi_ondalikli);
    printf("Uzun ondalikli sayi giriniz: ");
    scanf("%lf",&girdi_uzun_ondalikli);
    printf("-----\n");

    printf("Gİrdirler\n");
    printf("Harf: %c\n",girdi_harf);
    printf("Tam sayi: %d",girdi_tam_sayi);
    //Bir printf icinde birden fazla degisken basilabilir
    printf("Ondalikli sayi: %f\nUzun ondalikli sayi: %lf\n",girdi_ondalikli,
girdi_uzun_ondalikli);
    //Bu tip printlemelerde siralama cok onemlidir
    //en soldaki formata virgulden sonra en soldaki degisken basilir
    return 0;
}
```

```
root@MSI:/mnt/c/Users/mybal/Desktop/C course/weeks# gcc -std=c99 -Wall -Werror week01.c -o week01
root@MSI:/mnt/c/Users/mybal/Desktop/C course/weeks# ./week01
Harf giriniz: a
Tam sayi giriniz: 4
Ondalikli sayi giriniz: 4
Uzun ondalikli sayi giriniz: 2
-----
Girdiler
Harf: a
Tam sayi: 4
Ondalikli sayi: 4.000000
Uzun ondalikli sayi: 2.000000
root@MSI:/mnt/c/Users/mybal/Desktop/C course/weeks#
```

Operatörler

A) Aritmetik Operatörler

Aritmetik operatörler matematiksel operasyonlar gerçekleştirirler. Örnek olarak toplama, çıkarma, çarpma, bölme ve mod alma işlemleri verilebilir.

```
#include <stdio.h>

int main(){
    int sayi_1 = 10;
    int sayi_2 = 3;
    double toplama_islemi = sayi_1 + sayi_2;
    printf("%lf\n",toplama_islemi);
    double cikarma_islemi = sayi_1 - sayi_2;
    printf("%lf\n",cikarma_islemi);
    double carpma_islemi = sayi_1 * sayi_2;
    printf("%lf\n",carpma_islemi);
    //Alttaki islem sonucunun 3.333 olmasi beklenir
    //fakt sonuc 3.0000 olarak gozukur buna Integer Division denir
    //yani iki sayi birbirine bolundugunde sonuc
    //ondalikli olmasi gerekse bile oyle olmaz
    double tam_sayi_bolme_islemi = sayi_1 / sayi_2; //sonuc 3.0000
    printf("%lf\n",tam_sayi_bolme_islemi);
    //Integer Division'dan kacinmak icin
    //islemdeki sayılardan birinin onune double isareti konulmalidir (onceli
    k durumuna dikkat edilmelidir)
    double ondalikli_sayi_bolme_islemi = sayi_1 / (double) sayi_2; //sonuc 3
    .333333
```

```

printf("%lf\n",ondalikli_sayi_bolme_islemi);
int mod_islemi = sayi_1 % sayi_2;
printf("%d\n",mod_islemi);
return 0;
}

```

```

root@MSI: /mnt/c/Users/mybal/Desktop/C course/weeks
root@MSI:/mnt/c/Users/mybal/Desktop/C course/weeks# gcc -std=c99 -Wall -Werror week01.c -o
week01
root@MSI:/mnt/c/Users/mybal/Desktop/C course/weeks# ./week01
13.000000
7.000000
30.000000
3.000000
3.333333
1

```

Önemli not: İki tam sayı birbirlerine direkt bölündükleri zaman eğer sonuç normalde ondalıklı olsa bile çıktı böyle olmaz. Dolayısıyla bir bölme yapılacağı zaman veri kaybı olmaması için bölen ve bölünen ikilisinden birisinin değişken tipi **double** olarak belirtilmelidir.

B) Arttırma – Azaltma Operatörleri (Increment – Decrement)

Pek çok programlama dilinde bulunan bu operatörler değişkenlerin kolayca 1 arttırılmasını veya azaltılmasını sağlar.

- ++ 1 Arttırma operatörüdür.
- -- 1 Azaltma operatörüdür.

```

#include <stdio.h>

int main(){
    int sayi_1 = 10;
    int sayi_2 = 10;
    printf("sayi_1: %d\n",sayi_1);
    printf("sayi_2: %d\n",sayi_2);
    sayi_1++;
    sayi_2--;
    printf("sayi_1: %d\n",sayi_1);
    printf("sayi_2: %d\n",sayi_2);
    return 0;
}

```

```
root@MSI: /mnt/c/Users/mybal/Desktop/C course/weeks
root@MSI:/mnt/c/Users/mybal/Desktop/C course/weeks# gcc -std=c99 -Wall -Werror week01.c -o ^
week01
root@MSI:/mnt/c/Users/mybal/Desktop/C course/weeks# ./week01
sayi_1: 10
sayi_2: 10
sayi_1: 11
sayi_2: 9
root@MSI:/mnt/c/Users/mybal/Desktop/C course/weeks#
```

C) Atama Operatörleri

Dersin başından beri kullanılan = işareti temel atama operatörüdür.

- += operatörü soldaki değişkene sağdakini ekler. Yani $a += b$ demek $a+b$ toplamını a'ya ata demektir ($a = a + b$).

Not: Buradaki = işareti matematikteki ile aynı görevi görmemektedir.

- -= operatörü soldaki değişkenden sağdakini çıkarır. Yani $a -= b$ demek $a = a-b$ demektir.
- *= operatörü soldaki değişkeni sağdaki ile çarpar. Yani $a *= b$ demek $a = a*b$ demektir.
- /= operatörü soldaki değişkeni sağdakine böler. Yani $a /= b$ demek $a = a/b$ demektir.
- %= operatörü soldaki değişkeni sağdaki ile böler ve kalanı soldakine atar. Yani $a \% b$ demek $a = a \% b$ demektir.

```

#include <stdio.h>

int main(){
    int sayi_1 = 10;
    int sayi_2 = 10;
    int sayi_3 = 10;
    int sayi_4 = 10;
    int sayi_5 = 10;
    sayi_1 += 3;
    printf("%d\n",sayi_1);
    sayi_2 -= 3;
    printf("%d\n",sayi_2);
    sayi_3 /= 3; // DIKKAT Integer Division
    printf("%d\n",sayi_3);
    sayi_4 *= 3;
    printf("%d\n",sayi_4);
    sayi_5 %= 3;
    printf("%d\n",sayi_5);
    return 0;
}

```

```

root@MSI: /mnt/c/Users/mybal/Desktop/C course/weeks
root@MSI:/mnt/c/Users/mybal/Desktop/C course/weeks# gcc -std=c99 -Wall -Werror week01.c -o week01
root@MSI:/mnt/c/Users/mybal/Desktop/C course/weeks# ./week01
13
7
3
30
1
root@MSI:/mnt/c/Users/mybal/Desktop/C course/weeks#

```

Temel Kütüphaneler & Standart Fonksiyonlar

Kütüphaneler, hazır fonksiyonlar bulunduran kodlar topluluğudur. Bu hazır kodları kullanarak Amerika Kitasını baştan keşfetmeye gerek kalmaz.

- <stdio.h> Temel INPUT(scanf)/OUTPUT(sprintf) fonksiyonlarını içerir .
- <stdlib.h> EXIT_SUCCESS ve EXIT_FAILURE return kalıplarını içerir.
- <math.h> Yaygın olarak kullanılan matematiksel fonksiyonları içerir (karekök vs.)

Önemli not: Matematik kütüphanesi kullanıldığı zaman derleme komutunun sonuna -**lm** (link math) ibaresi konulmalıdır.

```
#include <stdio.h> //scanf ve printf
#include <stdlib.h> // EXIT_SUCCESS;
#include <math.h> //matematiksel islemler

int main(){
    int sayi_1 = 100;
    int sayi_2 = 2;
    printf("%lf\n",sqrt(sayi_1));
    printf("%lf\n",pow(sayi_2,3));
    return 0;
}
```

```
root@MSI: /mnt/c/Users/mybal/Desktop/C course/weeks
root@MSI:/mnt/c/Users/mybal/Desktop/C course/weeks# gcc -std=c99 -Wall -Werror week01.c -o^
week01 -lm
root@MSI:/mnt/c/Users/mybal/Desktop/C course/weeks# ./week01
10.000000
8.000000
```