



# ITU ACM Student Chapter Course Program

## Introduction to C

### Week 2

#### Instructor

Mehmet Yiğit Balık

#### Prepared by

Mehmet Yiğit Balık & Mihriban Nur Koçak & Emir Oğuz

## Koşullu İfadeler (if / else / else if)

Koşullu ifadeler, belirtilen bir durum altında çalışması istenilen kodların çalışmasını sağlar. Örneğin tam sayı veri tipine ait bir *test* değişkeni belirlenmiş olsun ve bu değişkene 5 değeri atansın. Daha sonra *test* değişkeni eğer 10'dan küçükse ekrana 10'dan küçük yazdırılsın.

Bu işlemi gerçekleştirebilmek için **if** bloğu kullanılması gerekir. **if** bloğunun yazım düzeni (syntax) şu şekildedir:

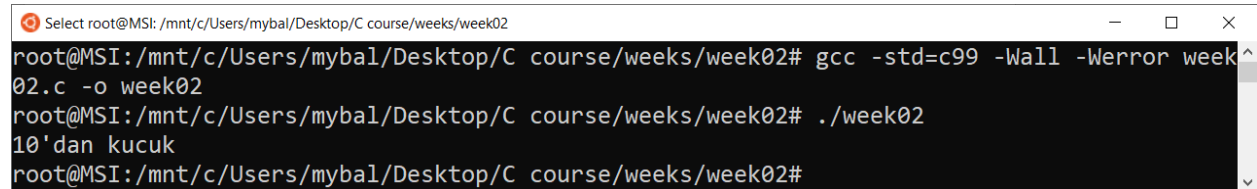
```
if (koşul buraya yazılır) {
```

Belirtilen koşul altında çalışması istenen kod buraya yazılır;

```
}
```

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

int main(){
    int test = 5;
    if(test < 10){
        printf("10'dan kucuk\n");
    }
    return EXIT_SUCCESS;
}
```



```
Select root@MSI: /mnt/c/Users/mybal/Desktop/C course/weeks/week02
root@MSI:/mnt/c/Users/mybal/Desktop/C course/weeks/week02# gcc -std=c99 -Wall -Werror week02.c -o week02
root@MSI:/mnt/c/Users/mybal/Desktop/C course/weeks/week02# ./week02
10'dan kucuk
root@MSI:/mnt/c/Users/mybal/Desktop/C course/weeks/week02#
```

5 değeri 10 değerinden küçük olduğu için 10'dan kucuk çıktısı ekrana yazdırılmış oldu. Şimdi *test* değişkeninin değeri 15 yapılsın. Bu durumda **if** bloğunun içindeki kodlar çalışmayacaktır.

**if** bloğuna ek olarak bir **else** bloğu eklenebilir. **else** bloğu kendisinin üstünde yazılmış olan **if** koşulu sağlanmadığı **her durumda** çalışacaktır. Her durumda çalışacağından dolayı **else** bloğunda parantezli koşul ifadesi kullanılmaz. **else** bloğu yazım düzeni (syntax) şu şekildedir.

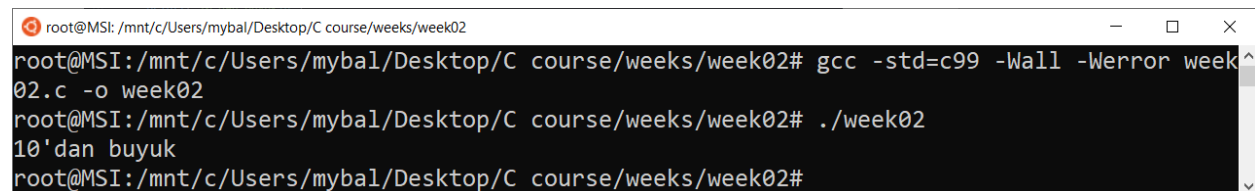
**else {**

if koşulu sağlanmadığı her durumda çalışması istenen kod buraya yazılır;

**}**

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

int main(){
    int test = 15;
    if(test < 10){
        printf("10'dan kucuk\n");
    }
    else{
        printf("10'dan buyuk\n");
    }
    return EXIT_SUCCESS;
}
```



```
root@MSI: /mnt/c/Users/mybal/Desktop/C course/weeks/week02
root@MSI:/mnt/c/Users/mybal/Desktop/C course/weeks/week02# gcc -std=c99 -Wall -Werror week02.c -o week02
root@MSI:/mnt/c/Users/mybal/Desktop/C course/weeks/week02# ./week02
10'dan buyuk
root@MSI:/mnt/c/Users/mybal/Desktop/C course/weeks/week02#
```

Yukarıda görüldüğü üzere **if** bloğu ilk olarak 15 değerine sahip *test* değişkeni 10'dan küçük mü diye kontrol etti ve *test* değeri 10'dan büyük olduğu için kendi bünyesindeki kodları çalıştırmadı. Bunun üzerine **else** bloğu **if** kodları çalışmadığı için kendi kodlarını çalıştırdı. Eğer **if** kodları çalışsaydı **else** kodları çalışmayacaktı.

**if** ve **else** blokları sadece bir koşulun olması ve olmaması ihtimali üzerine çalışır. **else if** ile farklı bir koşul belirleyerek bu ihtimaller arttırılabilir. **else if** bloğunun yazım düzeni **if** bloğu ile aynıdır.

```
#include <stdio.h>
#include <stdlib.h>
int main(){
    int test = 15;
    if(test < 10){
        printf("10'dan küçük\n");
    }
    else if(test < 20){
        printf("20'den küçük\n");
    }
    else {
        printf("20'den büyük\n");
    }
    return EXIT_SUCCESS;
}
```

```
root@MSI: /mnt/c/Users/mybal/Desktop/C course/weeks/week02
root@MSI:/mnt/c/Users/mybal/Desktop/C course/weeks/week02# gcc -std=c99 -Wall -Werror week02.c -o week02
root@MSI:/mnt/c/Users/mybal/Desktop/C course/weeks/week02# ./week02
20'den küçük
root@MSI:/mnt/c/Users/mybal/Desktop/C course/weeks/week02#
```

Örnekte de görüldüğü gibi **if** , **else if**, ve **else** bloklarının bir arada kullanılması durumunda, önce belirlenen *test* değeri için **if** bloğunun koşulu kontrol edilir, bu koşul sağlanmadığı takdirde **else if** bloğunun koşulu kontrol edilir ve koşul sağlanırsa bu bloktaki kod çalıştırılır. Sağlanmaması durumunda ise **else** bloğunun kodu çalıştırılacaktır.

Tüm bu koşul yapılarında koşulları belirlerken kullanılan belirli operatörler vardır:

- **==** operatörü herhangi iki değerin eşitlik durumunu kontrol eder.
- **!=** operatörü herhangi iki değerin eşit olmama durumunu kontrol eder.
- **>=** operatörü soldaki değerin sağdaki değerden büyük veya eşit olması durumunu kontrol eder.
- **<=** operatörü soldaki değerin sağdaki değerden küçük veya eşit olması durumunu kontrol eder.
- **>** operatörü soldaki değerin sağdaki değerden büyük olması durumunu kontrol eder.

- < operatörü soldaki değerin sağdaki değerden küçük olması durumunu kontrol eder.

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

int main(){
    int test = 15;
    if(test != 16){
        printf("16'ya esit degildir\n");
    }
    return EXIT_SUCCESS;
}
```

```
root@MSI: /mnt/c/Users/mybal/Desktop/C course/weeks/week02
root@MSI:/mnt/c/Users/mybal/Desktop/C course/weeks/week02# gcc -std=c99 -Wall -Werror week02.c -o week02
root@MSI:/mnt/c/Users/mybal/Desktop/C course/weeks/week02# ./week02
16'ya esit degildir
root@MSI:/mnt/c/Users/mybal/Desktop/C course/weeks/week02#
```

Yukarıda gösterilen operatörler aslında **boolean** adı verilen (temelde 1 ve 0'dan oluşan) değerleri döndürür. İki tane boolean değeri vardır. Bunlar **true** ve **false** mantıksal değerleridir. **true** sayısal olarak 1 sayısına, **false** ise 0 sayısına karşılık gelir. Örneğin  $15 == 15$  koşulu, 15 değeri 15'e eşit olduğu için 1 yani **true** değerini döndürür.  $15 \geq 16$  koşulu ise 15 değeri 16'dan büyük veya eşit olmadığı için 0 yani **false** değerini döndürür.

**true** ve **false** mantıksal değerlerini kod yazarken kullanabilmek için **<stdbool.h>** kütüphanesini kodun başına eklemek gerekir.

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h> // true false
int main(){
    printf("%d\n", true);
}
```

```

printf("%d\n", false);
printf("%d\n", true == 1);
printf("%d\n", false == 0);
printf("%d\n", 15 == 15); //true
printf("%d\n", 15 > 16); // false
return EXIT_SUCCESS;
}

```

```

root@MSI: /mnt/c/Users/mybal/Desktop/C course/weeks/week02
root@MSI:/mnt/c/Users/mybal/Desktop/C course/weeks/week02# gcc -std=c99 -Wall -Werror week02.c -o week02
root@MSI:/mnt/c/Users/mybal/Desktop/C course/weeks/week02# ./week02
1
0
1
1
1
1
0
root@MSI:/mnt/c/Users/mybal/Desktop/C course/weeks/week02#

```

Tek bir **if** veya **else if** bloğu içinde birden fazla koşulu kontrol edebilmek için ise belirli mantıksal operatörler kullanılır:

- **&&** : ve mantıksal operatörüdür
- **||** : veya mantıksal operatörüdür
- **!** : değil mantıksal operatörüdür

1. İfade	2. İfade	1. İfade AND 2. İfade	1. İfade	2. İfade	1. İfade OR 2. İfade
True	True	True	True	True	True
True	False	False	True	False	True
False	True	False	False	True	True
False	False	False	False	False	False

**Not:**

- $15 == 15 \ \&\& \ 15 \leq 16$  : **true** ve **true** : **true**
- $15 == 15 \ || \ 15 > 16$  : **true** veya **false** : **true**
- $!(15 == 15)$  : **true** değil : **false**

Şimdi tüm bu öğrenilenleri kapsayacak bir örnek çözülebilir.

**Not:** Üniversite notlandırma sistemi

- 100 - 91 : AA
- 90 - 81 : BA
- 80 - 71 : BB
- 70 - 61 : CB
- 60 - 51 : CC
- 50 - 41 : DC
- 40 - 31 : DD
- 30 ve daha az : FF

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h> // true false
int main() {
    int sayisal_not;
    printf("***Universite ogrenci islerine hos geldiniz!***\n");
    printf("Harf notunuzu ogrenmek icin lutfen sayisal notunuzu giriniz: ");
    scanf("%d",&sayisal_not);
    //eger not 100'den kucuk esit ve 90'dan buyukse
    if(sayisal_not <= 100 && sayisal_not > 90){
        printf("Harf notunuz: AA\n");
    }
    //eger not 90'dan kucuk esit ve 80'den buyukse
    else if(sayisal_not <= 90 && sayisal_not > 80){
        printf("Harf notunuz: BA\n");
    }
    else if(sayisal_not <= 80 && sayisal_not > 70){
        printf("Harf notunuz: BB\n");
    }
    else if(sayisal_not <= 70 && sayisal_not > 60){
        printf("Harf notunuz: CB\n");
    }
    else if(sayisal_not <= 60 && sayisal_not > 50){
        printf("Harf notunuz: CC\n");
    }
    else if(sayisal_not <= 50 && sayisal_not > 40){
        printf("Harf notunuz: DC\n");
    }
}
```

```

}
else if(sayisal_not <= 40 && sayisal_not > 30){
    printf("Harf notunuz: DD\n");
}
//eger yukardaki kosullar saglanmiyorsa
else{
    printf("Harf notunuz: FF\n");
}
return EXIT_SUCCESS;
}

```

```

root@MSI: /mnt/c/Users/mybal/Desktop/C course/weeks/week02
root@MSI:/mnt/c/Users/mybal/Desktop/C course/weeks/week02# gcc -std=c99 -Wall -Werror week02.c -o week02
root@MSI:/mnt/c/Users/mybal/Desktop/C course/weeks/week02# ./week02
***Universite ogrenci islerine hos geldiniz!***
Harf notunuzu ogrenmek icin lutfen sayisal notunuzu giriniz: 90
Harf notunuz: BA
root@MSI:/mnt/c/Users/mybal/Desktop/C course/weeks/week02# ./week02
***Universite ogrenci islerine hos geldiniz!***
Harf notunuzu ogrenmek icin lutfen sayisal notunuzu giriniz: 30
Harf notunuz: FF
root@MSI:/mnt/c/Users/mybal/Desktop/C course/weeks/week02#

```

## Döngüler (for / while)

Döngüler bir nevi kısayollardır. Bir işlemin kolayca tekrarlanmasını sağlarlar. Döngüler yardımıyla 1'den 100'e kadar sayıları saydırmak için küçük bir işlem yeterli olur. Örneğin şu ana kadar öğrenilenlerle 100 defa "döngüler" yazdırabilmek için 100 satır "printf()" kodu kullanılması gerekecekti. Fakat döngüler sayesinde birkaç satır kod ile istenilen sonuç alınabilir.

Sıklıkla kullanılan iki çeşit döngü vardır:

- **for**
- **while**



## for döngüsü:

```
for (değişken = başlangıç_değeri ; değişken'e bağlı koşul ; değişkenin güncellenmesi){  
    işlemler;  
}
```

**for** parantezinin içinde üç farklı kısım bulunur:

- İlk kısımda döngünün içinde kullanılacak değişkene bir başlangıç değeri atanır.
- İkinci kısımda değişkene bağlı bir koşul belirtilir. Eğer koşul sağlanıyorsa döngü devam eder, aksi halde döngü sona erer.
- Değişkenin her bir döngünün sonunda ne şekilde güncelleneceği belirtilir.

**Not:** Eğer her adımda değişken güncellenmezse döngü sonsuza kadar devam eder ve kodun geri kalanının çalışmasını engeller.

```
#include <stdio.h>  
#include <stdlib.h>  
int main(){  
    for(int i = 1; i <= 100; i++){  
        printf("%d ", i);  
    }  
    return EXIT_SUCCESS;  
}
```

```
root@MSI: /mnt/c/Users/mybal/Desktop/C course/weeks/week02  
root@MSI:/mnt/c/Users/mybal/Desktop/C course/weeks/week02# gcc -std=c99 -Wall -Werror week02.c -o week02  
root@MSI:/mnt/c/Users/mybal/Desktop/C course/weeks/week02# ./week02  
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33  
34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63  
64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93  
94 95 96 97 98 99 100  
root@MSI:/mnt/c/Users/mybal/Desktop/C course/weeks/week02#
```

## while döngüsü:

**while**(değişken'e bağlı koşul){

işlemler;

}

**while** parantezinin içerisinde tek bir kısım bulunur:

- Bu kısımda döngüden önce tanımlanmış bir değişkene bağlı koşul belirtilir.

**Not:** Her döngüde, daha önceden tanımlanmış olan değişken güncellenmelidir, eğer güncellenmezse döngü sonsuza kadar devam eder ve kodun geri kalanının çalışmasını engeller.

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    int i = 1;
    while(i <= 100) {
        printf("%d ", i);
        i++;
    }
    printf("\n");
    return EXIT_SUCCESS;
}
```

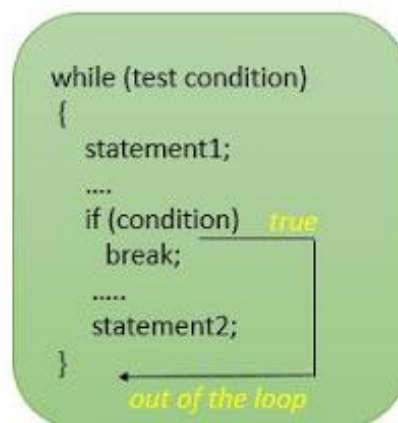
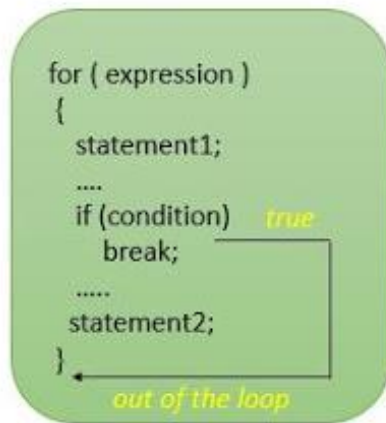
```
root@MSI: /mnt/c/Users/mybal/Desktop/C course/weeks/week02
root@MSI:/mnt/c/Users/mybal/Desktop/C course/weeks/week02# gcc -std=c99 -Wall -Werror week02.c -o week02
root@MSI:/mnt/c/Users/mybal/Desktop/C course/weeks/week02# ./week02
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33
34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63
64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93
94 95 96 97 98 99 100
root@MSI:/mnt/c/Users/mybal/Desktop/C course/weeks/week02#
```

## break ve continue

**break** bir döngünün istenilen koşula göre sonlanmasını sağlayan bir anahtar kelimedir. Örnek olarak 1'den 100'e kadar saydırma yapan bir döngüde değişkenin değerinin 50 olması durumunda döngünün durdurulması istendiğinde **break** kullanılabilir.

```
#include <stdio.h>
#include <stdlib.h>
int main(){
    int i = 1;
    while(i <= 100){
        printf("%d ",i);
        if(i == 50){
            printf("Dongu kirildi...\n");
            break;
        }
        i++;
    }
    printf("\n");
    return EXIT_SUCCESS;
}
```

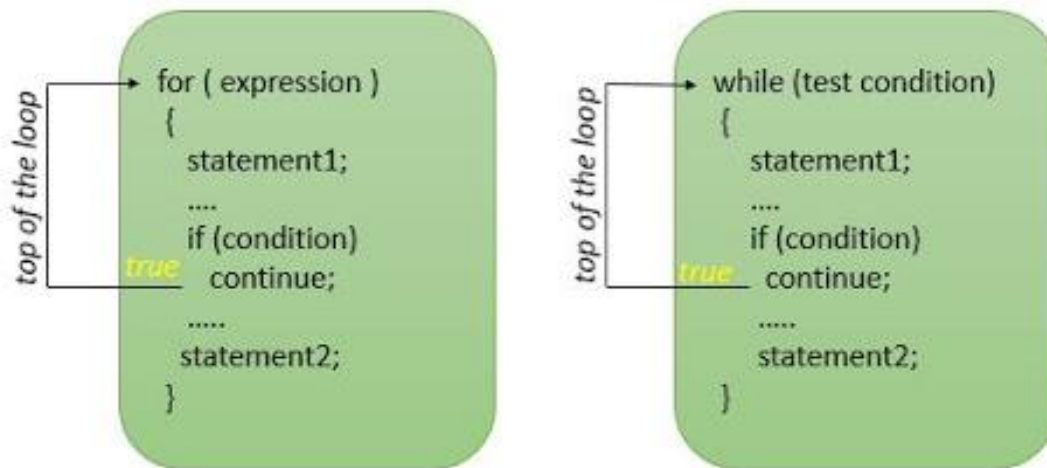
```
root@MSI: /mnt/c/Users/mybal/Desktop/C course/weeks/week02
root@MSI:/mnt/c/Users/mybal/Desktop/C course/weeks/week02# gcc -std=c99 -Wall -Werror week02.c -o week02
root@MSI:/mnt/c/Users/mybal/Desktop/C course/weeks/week02# ./week02
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33
34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 Dongu kirildi...
root@MSI:/mnt/c/Users/mybal/Desktop/C course/weeks/week02#
```



**continue** bir döngünün istenilen bir durumda es geçilmesini sağlayan anahtar kelimedir. **break** anahtar kelimesi istenilen durumda döngüyü sonlandırırken **continue** istenilen durumun es geçilmesini sağlar. Örnek olarak 1'den 100'e kadar tek sayıları yazdırma verilebilir.

```
#include <stdio.h>
#include <stdlib.h>
int main(){
    int i = 1;
    while(i <= 100){
        if(i % 2 == 0){
            i++;
            continue;
        }
        printf("%d ",i);
        i++;
    }
    printf("\n");
    return EXIT_SUCCESS;
}
```

```
root@MSI: /mnt/c/Users/mybal/Desktop/C course/weeks/week02
root@MSI: /mnt/c/Users/mybal/Desktop/C course/weeks/week02# gcc -std=c99 -Wall -Werror week02.c -o week02
root@MSI: /mnt/c/Users/mybal/Desktop/C course/weeks/week02# ./week02
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49 51 53 55 57 59 61 63
65 67 69 71 73 75 77 79 81 83 85 87 89 91 93 95 97 99
root@MSI: /mnt/c/Users/mybal/Desktop/C course/weeks/week02#
```



## Rastgele Sayı Üretme ve `<time.h>` Kütüphanesi

Rastgele sayılara zar atma, loto vb. şans oyunlarında ihtiyaç duyulur. Rastgele sayılar üretmek için kullanılması gerekenler:

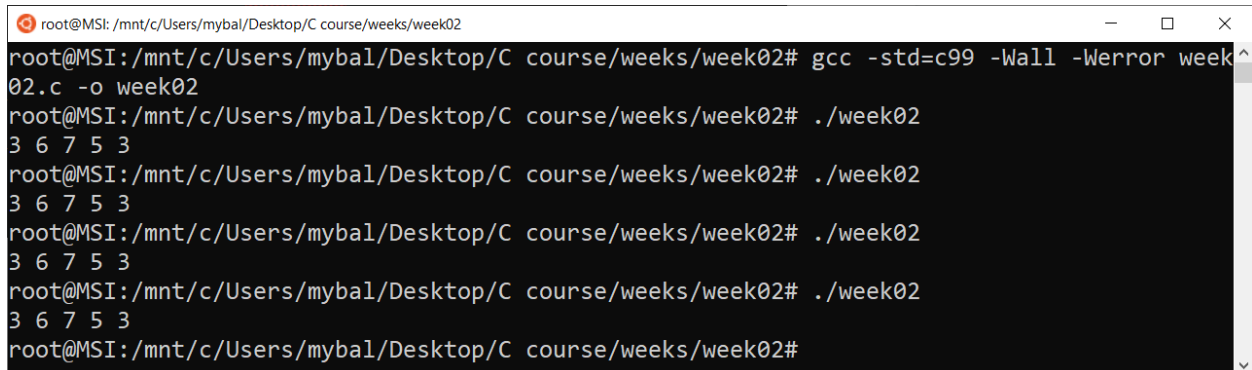
- `<stdlib.h>` kütüphanesinde bulunan **rand()** fonksiyonu
- `<stdlib.h>` kütüphanesinde bulunan **srand()** fonksiyonu
- `<time.h>` kütüphanesinde bulunan **time()** fonksiyonu

### rand() Fonksiyonu

C’de rastgele sayı üretmek için **rand()** fonksiyonu kullanılır. **rand()** fonksiyonu 0 ve RAND\_MAX aralığında sahte rastgelelik üretir. Sahte rastgelelik, daha önceden belirlenmiş sayılar dizisinden sırayla seçim yapılmasıdır.

```
#include <stdio.h>
#include <stdlib.h>
int main(){
    int ust_sinir = 10;
    for(int i = 0; i < 5; i++){
        printf("%d ", rand() % ust_sinir);
    }
    printf("\n");
    return EXIT_SUCCESS;
}
```

Yukarıdaki kodda mod (%) operatörü kullanılarak üst sınır belirlenmiştir. Bu yöntemin kullanılmasının nedeni matematiksel olarak düşünüldüğünde mod işleminin aslında belirli bir aralık oluşturabilmesidir. Örneğin tüm doğal sayıların mod 2’deki değeri ya 1 ya da 0 olabilir. Burada mod 2 değeri sadece 1 ve 0 dan oluşan bir sayı dizisi belirlemiştir.



```
root@MSI: /mnt/c/Users/mybal/Desktop/C course/weeks/week02
root@MSI:/mnt/c/Users/mybal/Desktop/C course/weeks/week02# gcc -std=c99 -Wall -Werror week02.c -o week02
root@MSI:/mnt/c/Users/mybal/Desktop/C course/weeks/week02# ./week02
3 6 7 5 3
root@MSI:/mnt/c/Users/mybal/Desktop/C course/weeks/week02# ./week02
3 6 7 5 3
root@MSI:/mnt/c/Users/mybal/Desktop/C course/weeks/week02# ./week02
3 6 7 5 3
root@MSI:/mnt/c/Users/mybal/Desktop/C course/weeks/week02# ./week02
3 6 7 5 3
root@MSI:/mnt/c/Users/mybal/Desktop/C course/weeks/week02#
```

Yukarıda görüldüğü üzere aynı kodun tekrar tekrar çalıştırılması durumunda üretilen sayılar tamamen aynıdır. Bu olay sahte rastgeleliktir.

Bu durumu engellemek için **<stdlib.h>** kütüphanesinden **srand()** fonksiyonu ve **<time.h>** kütüphanesinden **time()** fonksiyonu kullanılmalıdır.

## srand() ve time() fonksiyonları

**srand()** ve **time()** fonksiyonlarını **rand()** fonksiyonu kullanılmadan önce birlikte çalıştırarak, yukarıda bahsedilen sorun çözülebilir ve rastgele sayılar üretilebilir.

**srand()** fonksiyonu aldığı parametre ile **rand()** fonksiyonunu besleyerek belirli olan rastgele sayı dizisinde başlangıç noktasını belirler. Bu fonksiyona **time(NULL)** ya da **time(0)** parametre olarak verildiğinde *Unix zamandan(1 Ocak 1970)* şimdiki zamana kadar geçen süreyi saniye şeklinde döndürür ve bu saniye ile **rand()** fonksiyonunu "öngörülemez" bir şekilde besler. Bu sayede nispeten gerçek rastgele sayılar üretilmesini sağlar.

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
int main() {
    srand(time(NULL));
    int ust_sinir = 10;
    for(int i = 0; i < 5; i++){
        printf("%d ", rand() % ust_sinir);
    }
    printf("\n");
    return EXIT_SUCCESS;
}
```

```
root@MSI: /mnt/c/Users/mybal/Desktop/C course/weeks/week02
root@MSI: /mnt/c/Users/mybal/Desktop/C course/weeks/week02# gcc -std=c99 -Wall -Werror week02.c -o week02
root@MSI: /mnt/c/Users/mybal/Desktop/C course/weeks/week02# ./week02
2 7 7 0 9
root@MSI: /mnt/c/Users/mybal/Desktop/C course/weeks/week02# ./week02
1 2 7 4 1
root@MSI: /mnt/c/Users/mybal/Desktop/C course/weeks/week02# ./week02
6 1 8 6 0
root@MSI: /mnt/c/Users/mybal/Desktop/C course/weeks/week02# ./week02
9 2 0 4 0
root@MSI: /mnt/c/Users/mybal/Desktop/C course/weeks/week02#
```

Görüldüğü üzere artık program gerçek manada rastgele sayı üretebilir durumdadır.

Alt ve üst sınırı belirli olan bir sayı üretmek için yazı düzeni şu şekildedir:

**(rand() % (üst sınır - alt sınır + 1)) + alt sınır**