DÖNGÜLER

DÖNGÜ MANTIĞI VE WHİLE

Döngüler belirlediğimiz kodları birden fazla çalıştırmamızı sağlar. Hatırlarsanız bir öğrencinin harf notunu hesaplayıp programı bitirmiştik. birden fazla öğrenci varsa programı tekrar tekrar çalıştırmamız gerekiyordu. Bu durumdan kurtulmak için döngüleri kullanıyoruz.

Şimdi ilk döngü çeşidimizi tanıyalım: ve bir örnek yapalım

```
while( koşullu ifade ){

    koşul sağlanınca çalışacak kodlar

}
```

ilk önce koşullu ifadeye bakılır. eğer sağlanıyorsa blok yani { } arasındaki kodlar çalışır. bu kodlar bittiğinde koşullu ifade tekrar kontrol edilir. eğer koşul sağlanıyorsa bloktaki kodlar tekrar çalışır.

koşul sağlanmadığında döndü çalışmaz ve devam edilir

Diyelim ki 1 den 10 a kadar olan sayıları yazdırmak istiyoruz

koşul: sayının 10 dan küçük veya eşit olması
sayinin 11 den küçük olması da denebilir.

Çalışacak kod: sayinin ekrana yazdırılması.

```c
#include <stdio.h>

int main(){

    int sayi = 1;

    while( sayi <= 10){

        printf("%d\n",sayi);

        sayi++;
        //bu sekilde bir sonraki sayiya geciyoruz
    }

    return 0;
}
```

Eğer 11. satırdaki sayi++; ifadesini yazmazsak sayi hiç büyümez ve 1 olarak kaldığı için döngü hep çalışır hiç sonlanmaz. Sonlanmayan döngüye sonsuz döngü “infinite loop” denir.

```c
#include <stdio.h>

int main(){

    int sayac = 1;

    while( sayac <= 10){

        printf("Hello World\n");

        sayac++;

    }

    return 0;
}
```

Ekrana 10 defa “Hello world” yazdıralım

BİRKAÇ ÖRNEK:

SORU 1: 1 ile 10 arası çift sayıları bastırın 10 sınıra dahil

```c
int sayi = 2;

while( sayi <= 10){

    printf("%d\n",sayi);
    sayi += 2;

}
```

```c
int sayi = 1;

while( sayi <= 10){

    if(sayi % 2 == 0){
     printf("%d\n",sayi);
    }

     sayi++;

}
```

SORU2: 10 dan 1e doğru geri sayım yapın

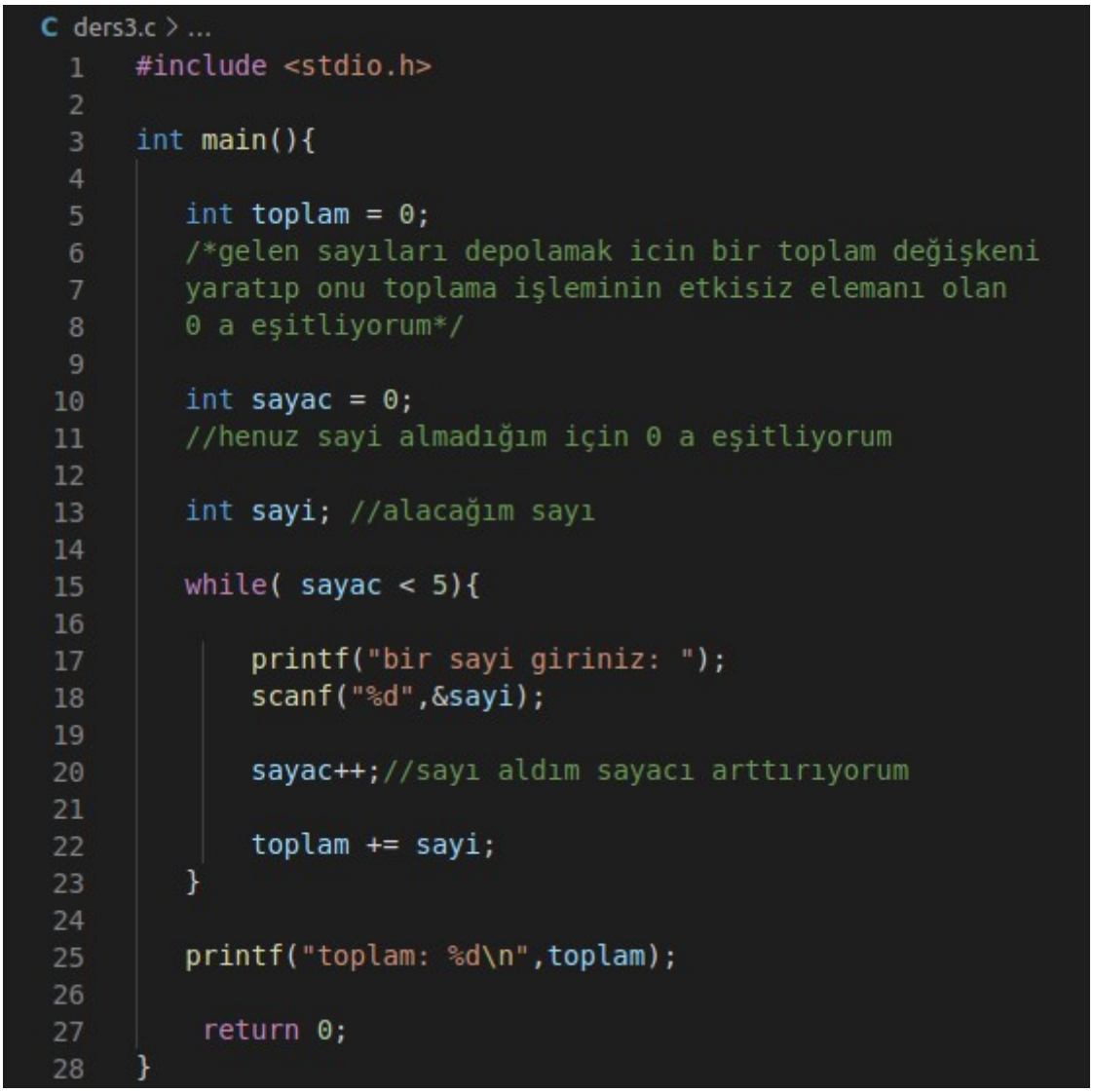
ders2.2.c > ...

```c
#include <stdio.h>

int main(){

    int sayi = 10;

    while( sayi > 0){

        printf("%d\n",sayi);

        sayi--;

    }

     return 0;
}
```

COUNTER CONTROLLED LOOP

bir döngünün kaç defa çalışacağı bir sayaca bağlıysa veya kaç defa çalışacağı önceden belliyse o döngüye counter controlled loop denir.

SORU: kullanıcıdan alınan 5 sayının toplamını bulun.



```c
#include <stdio.h>

int main(){

    int toplam = 0;
    /*gelen sayıları depolamak icin bir toplam değişkeni
    yaratıp onu toplama işleminin etkisiz elemanı olan
    0 a eşitliyorum*/

    int sayac = 0;
    //henuz sayi almadığım için 0 a eşitliyorum

    int sayi; //alacağım sayı

    while( sayac < 5){

        printf("bir sayi giriniz: ");
        scanf("%d",&sayi);

        sayac++;//sayı aldım sayacı arttırıyorum

        toplam += sayi;
    }

    printf("toplam: %d\n",toplam);

     return 0;
}
```

Birden fazla sayı alacağımız için scanf'i döngüye koyuyoruz.

Toplam sayac ve sayi değişkenlerini döngü dışında tanımlıyoruz. İçeride tanımlarsak ne olacağını derste gördük(bu kısım için c tutordan yararlanabilirsiniz). Size diyebileceğim şey kısaca: toplam değişkenine toplamanın etkisiz elemanını atayın ve toplam ile sayaç değişkenlerini döngü dışında tanımlayın

SENTİEL-CONTROLLED LOOP

SORU: Kullanıcıdan alınan sayıların toplamını bulun(kullanıcı negatif sayı girmeyecek)

Bu sorunun önceki dersten farkı kullanıcının kaç tane sayı gireceğini bilmiyoruz. Ama programın sonlanması gerekiyor, o yüzden bir sayı belirleyip o sayı girildiğinde programın sonlanmasını sağlayacağız. Bu sayıya sentiel(nöbetçi) veya flag(bayrak) value denir. Negatif sayı girilmeyecek o yüzden diyelim ki alınan sayı -1 olduğunda döngü bitsin.
Bu -1 sayısını sitelerdeki “kaydet ve çık” butonuna benzetebilirsiniz.

ders4.c > ...

```c
#include <stdio.h>

int main(){

    int toplam = 0;

    int sayi = 0; /*döngüye girmek zorundayım o yüzden
    -1 dışında bir başlangıç değeri veriyorum*/

    //sayi -1 olduğunda döngü çalışmayacak
    while( sayi != -1 ){

        printf("bir sayi giriniz(cikis icin -1 yazin): ");
        //kullanıcıya nasıl çıkacağını söylemek zorundayız
        scanf("%d",&sayi);

       //-1 girildiğinde toplama işlemi yapılmasın
        if(sayi != -1 )
            toplam += sayi;

    }

   printf("toplam: %d\n",toplam);

    return 0;
}
```

BREAK VE CONTİNUE

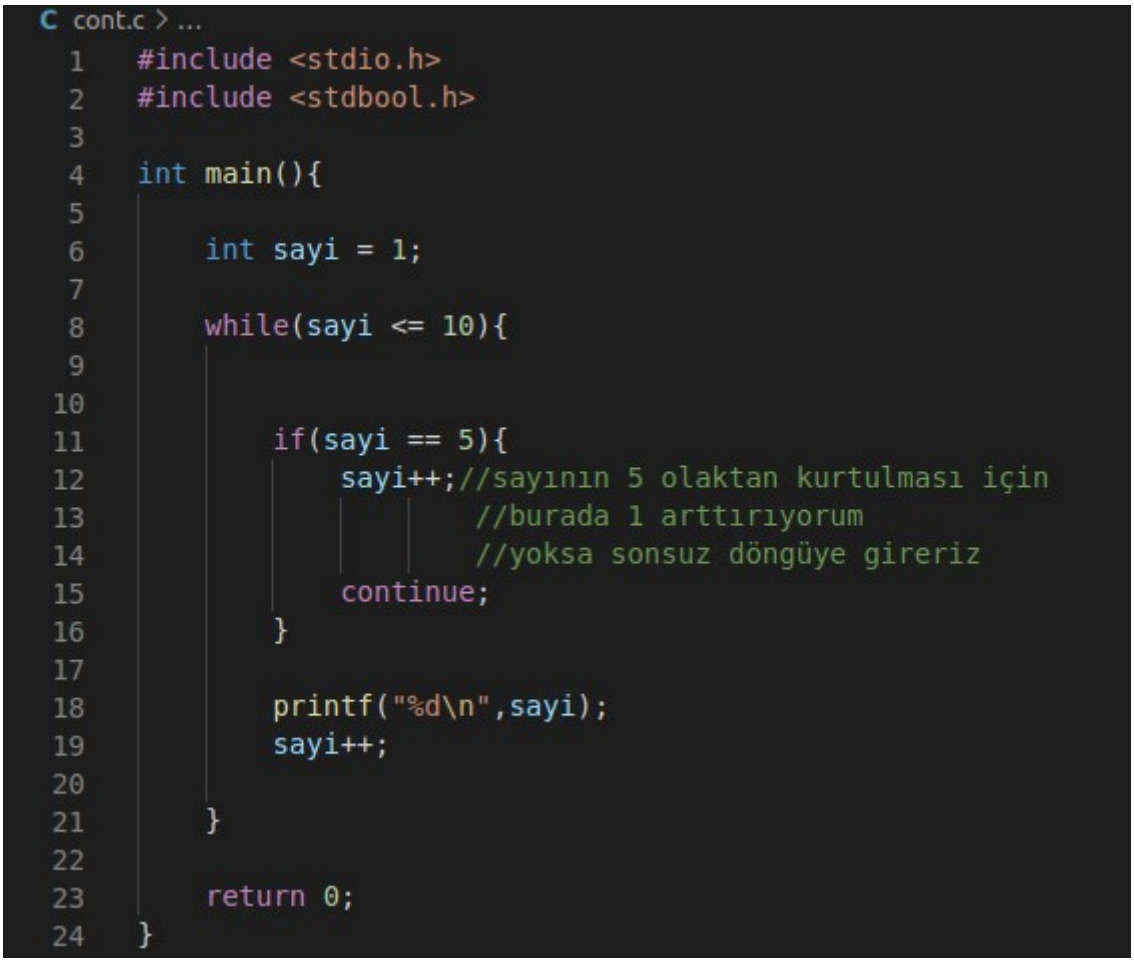
Sentiel value kullanmanın bir yolu daha var: while(true) ve break kullanmak. Break ifadesi döngünün sonlanmasını sağlar

SORU:Kullanıcıdan alınan sayıların toplamını bulun(kullanıcı negatif sayı girmeyecek)
sentiel value tekrar -1 olsun

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

int main(){

    int toplam = 0;

    int sayi;

    while( true ){

        printf("bir sayi giriniz(cikis icin -1 yazin): ");
        scanf("%d",&sayi);

        if(sayi == -1 )
            break;

        toplam += sayi;
    }

   printf("toplam: %d\n",toplam);

    return 0;
}
```

Continue ifadesi ise döngü bitmeden bir sonraki tekrara geçmeyi sağlar.

Diyelim ki 5 hariç 1 ile 10 arasındaki sayıları yazdırmak istiyoruz.



bankamatik

ders6.c > ...

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

int main(void){

    int bakiye = 0;
    int para;

    while(true){

        int secim;
        printf("lutfen yapmak isteginiz islemi secin\n1- bakiye sorgulama\n2- para yatırma\n3- para çekme\n");
        printf("cikis icin -1 girin: ");
        scanf("%d",&secim);

        if(secim == -1)
            break;

        if( !(-1 <= secim && secim <= 3) ){
            printf("gecersiz islem\n\n");
            continue;
        }

        switch (secim){
        case 1:
            printf("bakiyeniz: %d\n\n",bakiye);
            break;

        case 2:
            printf("ne kadar para yatiracaksiniz: ");
            scanf("%d",&para);

            bakiye += para;

            printf("yeni bakiyeniz: %d\n\n",bakiye);
            break;

        case 3:
            printf("ne kadar para cekeceksiniz: ");
            scanf("%d",&para);

            bakiye -= para;

            printf("yeni bakiyeniz: %d\n\n",bakiye);
        }

    }

    return 0;
}
```

FOR DÖNGÜSÜ

For döngüsü genellikle belirli bir sayıda çalışan (counter controlled) ikinci bir döngü tipidir.

```
for(atama veya tanımlama ; koşullu ifade ; statement ){

    kodlar

}
```

Aşağıdaki kodu inceleyelim

ders7.c > main()

```c
#include <stdio.h>

int main(){

//EKRANA 10 DEFA HELLO WORLD YAZDIRMA

    int i;

    for(i = 0; i < 10 ; i++){

        printf("hello world\n");

    }

    return 0;
}
```

1)i değişkeni 0 a eşitlenir

2) koşullu ifadeye bakılır, sağlanırsa blok çalışır; sağlanmazsa döngüden çıkılır

3)döngü içindeki kodlar çalışır

4)i değişkeni 1 arttırılır

5) koşullu ifade tekrar kontrol edilir (i değişkeni tekrar 0 a eşitlenMEZ)

i yerine counter adlı bir değişken kullanabilirsiniz ama i yazmak gelenektir. Kullanmakta sıkıntı yoktur.

19, 27, 35, 43, 51 sayılarını for döngüsü ile yazdıralım

```c
#include <stdio.h>

int main(){

    //19, 27, 35, 43, 51 sayılarını for döngüsü ile yazdıralım

    int i;
    for(i = 19; i <= 51 ; i+=8){//i++ yazmak zorunda değiliz :)

        printf("%d\n",i);

    }

    return 0;
}
```

20, 14, 8, 2, –4, –10 sayılarını ekrana yazdıralım

```c
int i;

for(i = 20 ; i >= -10 ; i -= 6){

    printf("%d\n",i);

}
```

Gelin bu soru aracılığıyla for döngüsünü biraz kurcalayalım

ilk başta bir atama yapmayıp boş bırakabilirsiniz

```c
int i = 20;
for(; i >= -10 ; i -= 6){

    printf("%d\n",i);

}
```

koşullu ifadeyi de yazmak zorunda değilsiniz ama bu durumda sonsuz döngü oluşur. Break kullanmak gerekir.

```c
int i;
for(i = 20;; i -= 6){

    if(i < -10){
        break;
    }

    printf("%d\n",i);

}
```

Statement kısmını da parantez içinde yazmayıp blok içinde yazabilirsiniz

```c
int i;
for(i = 20; i >= -10;){

    printf("%d\n",i);

    i -= 6;

}
```

C99

```
for(atama veya tanımlama ; koşullu ifade ; statement ){

    kodlar

}
```

Şu taslağa tekrar bakalım.
Atama veya tanımlama yazıyor biz sadece atama yaptık. Ama parantez içinde tanımlama da yapabiliriz.

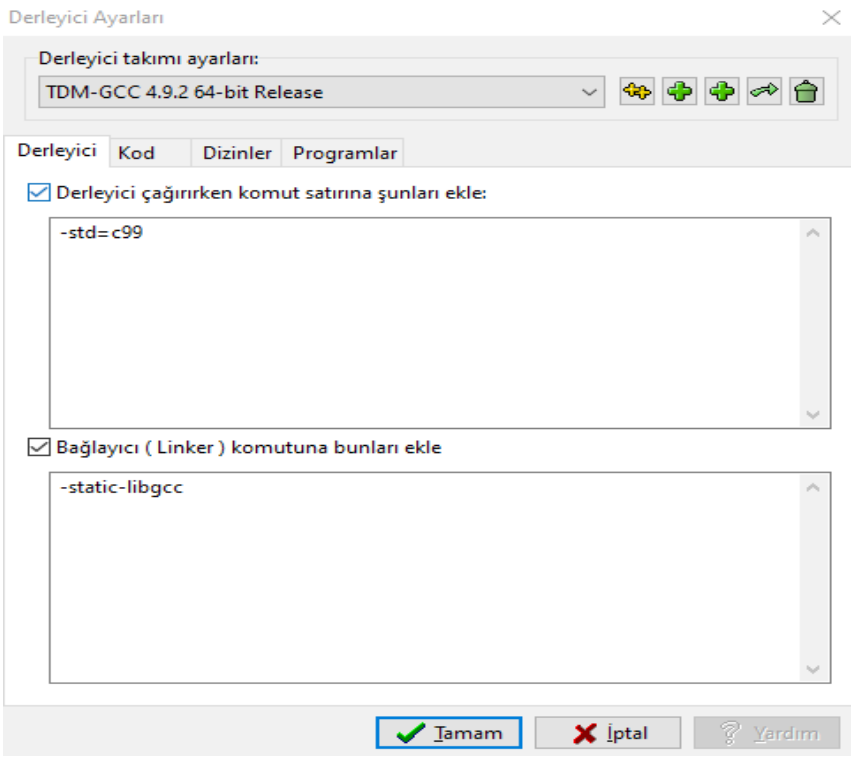
```
#include <stdio.h>

int main(){

    for(int i = 0;i < 10; i++){
        printf("hello\n");
    }

    return 0;

}
```

Doğru yazdık fakat Devc++ kullananlar bir hata alacaklar

C:\Users\Mehmet Emre Topdal\Desktop\CC++\EGITI...	In function 'main':
C:\Users\Mehmet Emre Topdal\Desktop\CC++\EGITIM\...	[Error] 'for' loop initial declarations are only allowed in C99 or C11 mode
C:\Users\Mehmet Emre Topdal\Desktop\CC++\EGITIM\...	[Note] use option -std=c99, -std=gnu99, -std=c11 or -std=gnu11 to compile your code

hata diyor ki: sadece c99 veya c11 standartlarında for içinde tanımlama yapabilirsin. İlk bölümde standartlardan bahsetmemizin bir sebebi de bu.
Kodu c99 ile çalıştırmak için
araçlar -> derleyici ayarları -> üstteki kutucuğa -std=c99 yazıp sol üstteki küçük kutucuğu işaretlemeliyiz.

(parantezde tanımlanan değişken döngü bitince yok edilir)



artık kodlarımız c99 standardına göre derlenecek

visual studio ve mingw kullanırken bu hatayı almayacağız fakat kodumuzu c99 standardına göre çalıştırmak için

gcc -std=c99 ders8.c -o ders8 komutunu kullanacağız.

BİR SAYININ ASAL SAYI OLUP OLMADIĞINI BULAN KOD

C ders9.c > ...

```c
#include <stdio.h>

int main(){

    int sayi;
    printf("lutfen bir sayi girin: ");
    scanf("%d",&sayi);

    int bolen_sayisi = 0;

    for(int i = 2; i <= sayi/2 ; i++){

        if(sayi % i == 0){
            bolen_sayisi++;
        }

    }

    /*1 girilirse döngüye girilmeden bölen sayısı
    0 olur bu yüzden sayının 1 olmadığını da kontrol etmeliyiz*/
    if(bolen_sayisi == 0 && sayi != 1){
        printf("sayi asal\n");
    }
    else{
        printf("sayi asal degil\n");
    }

    return 0;
}
```

FAKTORİYEL BULMA

faktoriyel sonuçta bir çarpımdır o yüzden daha önce toplam değişkeni tanımladığımız gibi bir çarpım değişkeni tanımlayıp o değişkene çarpmanın etkisiz elemanı olan 1 değerini verin

İÇ İÇE DÖNGÜLER – NESTED LOOPS

if bölümünde olduğu gibi bir döngünün içine ikinci bir döngü tanımlayabiliriz. İki boyutta çalışmamız gerektiğinde veya tablo yapacağımız iç içe döngüler kullanırız ve genellikle for içine for döngüsü koyarız

Hemen örnek yapalım

*
**
***
****
***** Şeklini bastırmaya çalışalım.

dışarıdaki döngü satır içerideki döngü sütün anlamına gelir . 5 satır var ve sutun daki * sayısı satırın sırasına eşit.

```c
#include <stdio.h>

int main(){

    for(int satir = 1; satir <= 5 ;satir++){
        for(int sutun = 1; sutun <= satir ; sutun++){
            printf("*");
        }
        //sutun bitti alt satıra geçiyorum
        printf("\n");
    }

    return 0;
}
```

1 5 25
2 10 50
3 15 75
4 20 100 Tablosunu yazdırın

```c
for(int i = 1 ; i <= 4; i++){
    for(int j = i; j <= i * 25 ; j *= 5){
        printf("%d ",j);
    }
    printf("\n");
}
```

DO-WHİLE DÖNGÜSÜ

bir döngüde kodların en az 1 kere çalışmasını garantilemek için do-while loop kullanabililiriz. yine de bu döngü yerine while döngüsü kullanmak çok daha yaygındır.

```
do{

kodlar

}while( koşullu ifade ) ; //dikkat burda noktalı virgül var
```

ekrana 10 defa hello world yazdıralım

```
int i = 1;

do{

printf("hello world\n);
i++;

}while(i <= 10);
```

---

kurs boyunca hiç kullanmayacağız fakat yabancı gelmesini istemediğim için göstermek istedim