Akış Kontrolü: Döngü ve Tekrarlar

Ders 5

İçerik

- Tekrar: if ve switch cümleleri
- Döngüler: while, for, do-while döngüleri
- Negatif sayı gösterimi ve binary operasyonlar hakkında kısa not

Tekrar: Bloklar

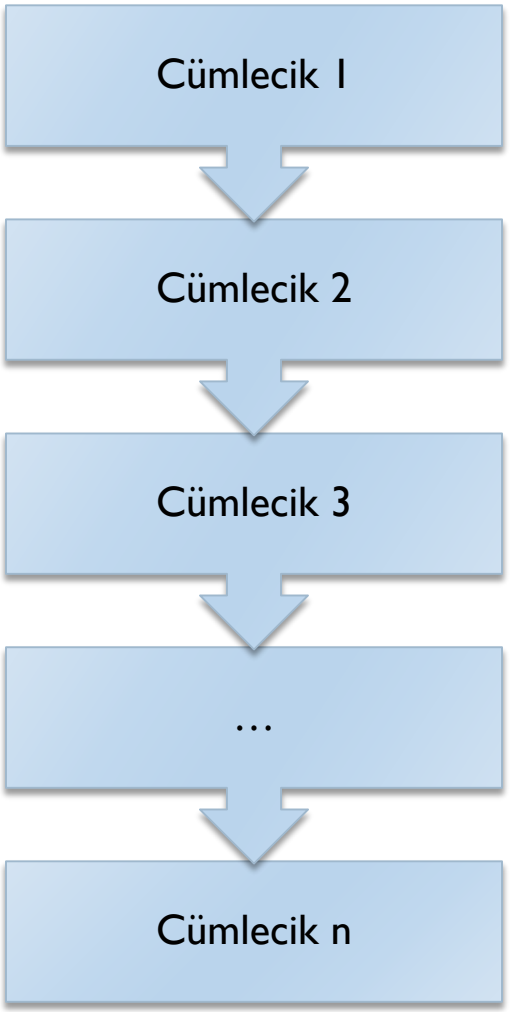
```
{
        int b1 = 0;
        /* sadece b1 görünmekte */
        {
                int b2 = 0;
                /* hem b1 hem de b2 görünmekte */
                {
                        int b3 = 0;
                        /* b1, b2 ve b3 görünmekte */
                }
                /* hem b1 hem de b2 görünmekte */
        }
        /* sadece b1 görünmekte */
}
```

Tekrar: Değişken gölgeleme

```
#include <stdio.h>
int main()
{
    int x = 1;/* blogun disinda*/
    int y = 1;
    printf("%d-%d\n", x, y);
    {
       int x = 1;/* blogun icinde */
        x++;
        y++;
        printf("%d-%d\n", x, y);
    }
    printf("%d-%d\n", x, y);
    return 0;
}
```

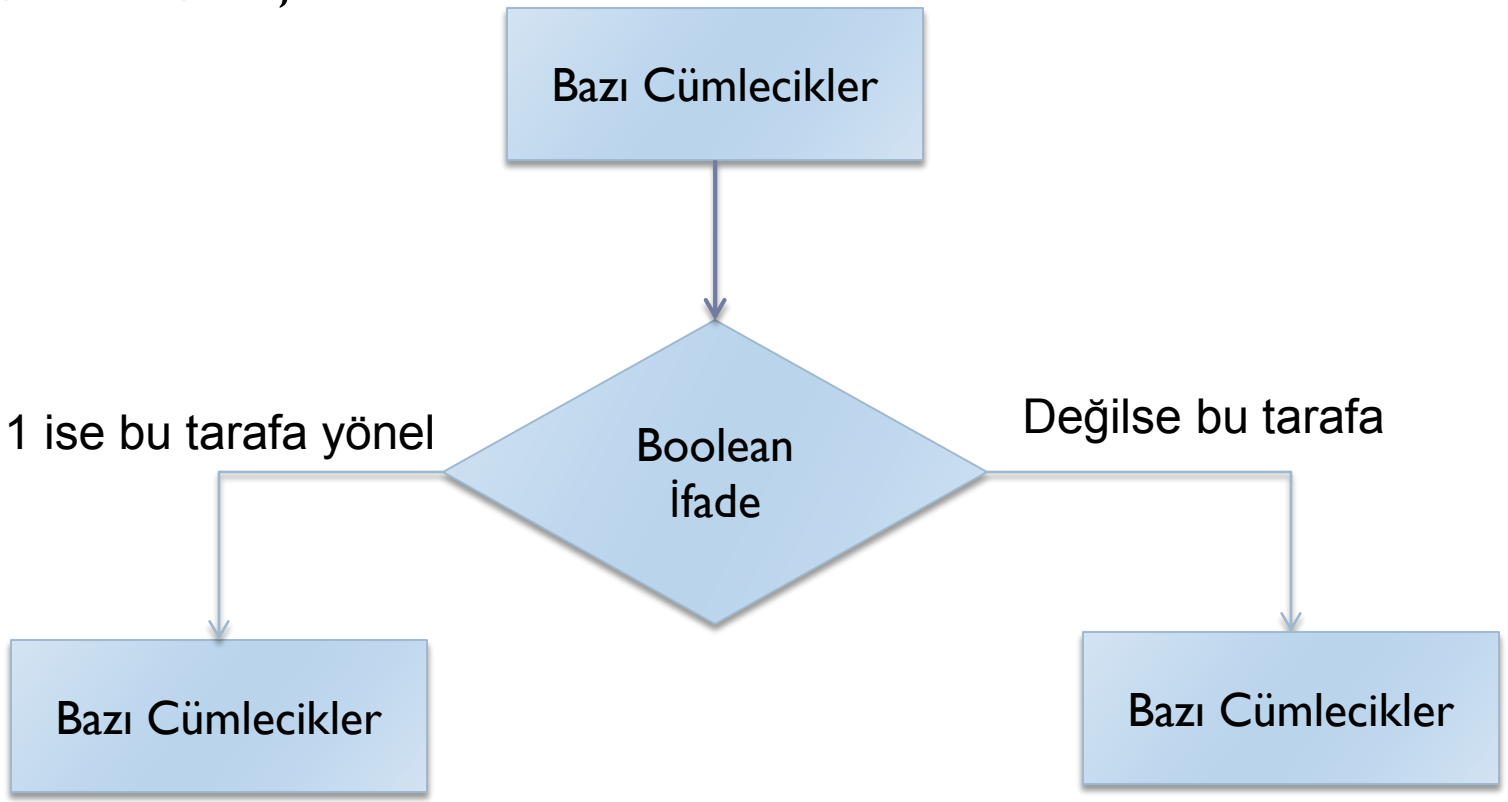
Akış Kontrolü

- Sıralı Çalıştırma



Akış Kontrolü

- Seçmeli Çalıştırma



Tekrar: If Cümleleri

```
if(boolean_ifade) {
   /* cümle(ler) */
}
```

```
int a = 10;

/* boolean şartını değerlendir */
if( a < 20 ) {

   /* Eğer true (doğru) ise, o zaman yazdır */
   printf("a  20 den kucuktur\n" );
}

printf("a nin degeri: %d\n", a);
```

Tekrar: If Cümleleri

```
if(boolean_ifade) {
   /* cümle(ler) */
}
else {
   /* cümle(ler) */
}
```

```
int score = 75;
char grade;

if (score >= 90) {
    grade = 'AA';
}
else if (score >= 80) {
    grade = 'BB';
}
 else {
    grade = 'FF';
}
```

Tekrar: If Cümleleri

```
if( boolean_ifade1) {
   /* cümleler */
   if(boolean_ifade2) {
      /* cümleler */
   }
}
```

```
int a = 100;
int b = 200;

if( a == 100 ) {

   if( b == 200 ) {
      printf("a ve b nin degeri 100 ve 200dur\n");
   }
}
printf("anin degeri: %d\n", a);
printf("bnindegeri: %d\n", b);
```

Tekrar: If Cümleleri

- Süslü parantezlerle, açık olmayan(karışıklığa sebep olabilecek) if cümlelerinden kaçının.

```
int x = 6, y = 0;
if (x%4 == 0)
    if (x%2 == 0)
        y = 2;
  else
     y = 1;
printf("y = %d", y);
```

```
int x = 6, y = 0;
if (x%4 == 0){
    if (x%2 == 0)
        y = 2;
}
else
    y = 1;
printf("y = %d", y);
```

Tekar: Switch Cümleleri

```
/* ifadenin tipi int veya char
olabilir */
switch(ifade) {
   case deger1:
        cumle(ler);
        break; /* opsiyonel */
   case deger2:
        cumle(ler);
        break; /* opsiyonel */
   default : /* opsiyonel */
        cumle(ler);
}
```

```
char grade = 'B';
switch(grade) {
   case 'A':
      printf("Excellent!\n" );
      break;
   case 'B':
   case 'C':
      printf("OK\n" );
      break;
   case 'D':
   case 'F':
      printf("Failed\n" );
      break;
   default:
      printf("Invalid\n" );
}
printf("Senin notun: %c\n",
          grade);
```

?: Operatörü

```
boolean_ifade? Ifade1: Ifade2;
```

```
if(boolean_ifade)
   Ifade1;
else
   Ifade2;
```

```
printf("%s", (month == 8)? "True\n": "False\n");
```

?: Operatörü

boolean_ifade? Ifade1: Ifade2;

```
int r1 = -1;
int r2 = -1;
char s1 = 'a';
char s2 = 'b';

r1 = s2 == s1? 1: 0;
r2 = s2 != s1? 1: 0;

printf("%d\t", r2 > r1? 1: 0);
```

Döngüler

- Döngüler bir cümleciğin veya bloğun tekrar tekrar çalıştırılması için kullanılırlar.
  - While döngüleri
  - For döngüleri
  - Do-while döngüleri

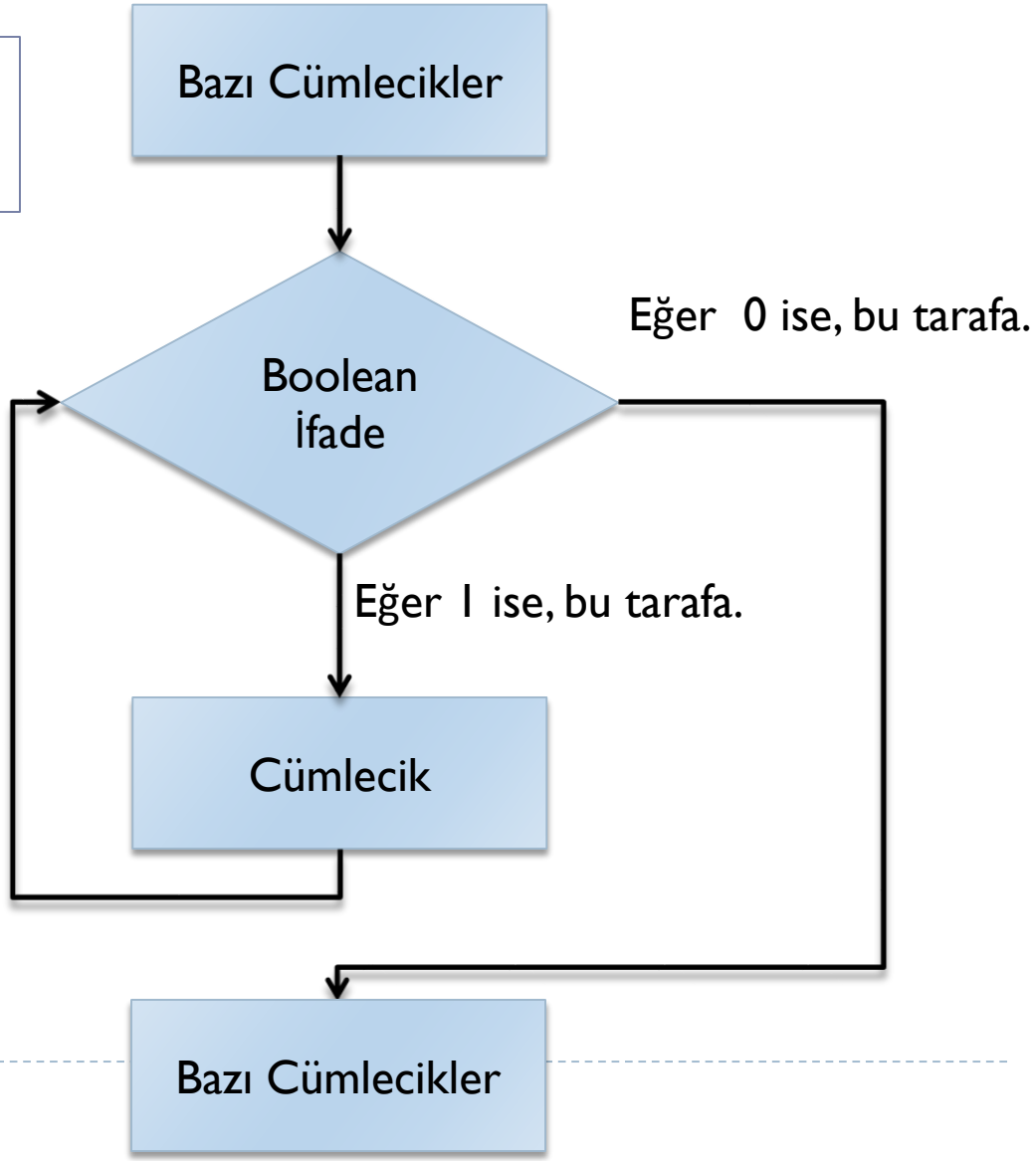
While Döngüleri

```
while(boolean_ifade)
        cümlecik;
```

- İlk başta, boolean ifade değerlendirilir,
  - Eğer doğru ise, o zaman takip eden cümlecik çalıştırılır.
  - Eğer doğru değilse, o zaman takip eden cümlecik çalıştırılmaz.
- Eğer cümlecik çalıştırıldıysa, boolean ifade tekrar değerlendirilir.
  - Eğer doğru ise, o zaman takip eden cümlecik çalıştırılır.
  - Eğer doğru değilse, o zaman takip eden cümlecik çalıştırılmaz.
- Şartın(boolean ifadenin) bir noktada yanlış olmasını sağlamalısınız:
  - Aksi takdirde, her zaman doğru ise, sonsuz bir döngünüz olur.

While Döngüsü Akış Şeması

```
while(boolean_ifade)
        cümlecik;
```



While Döngüsü

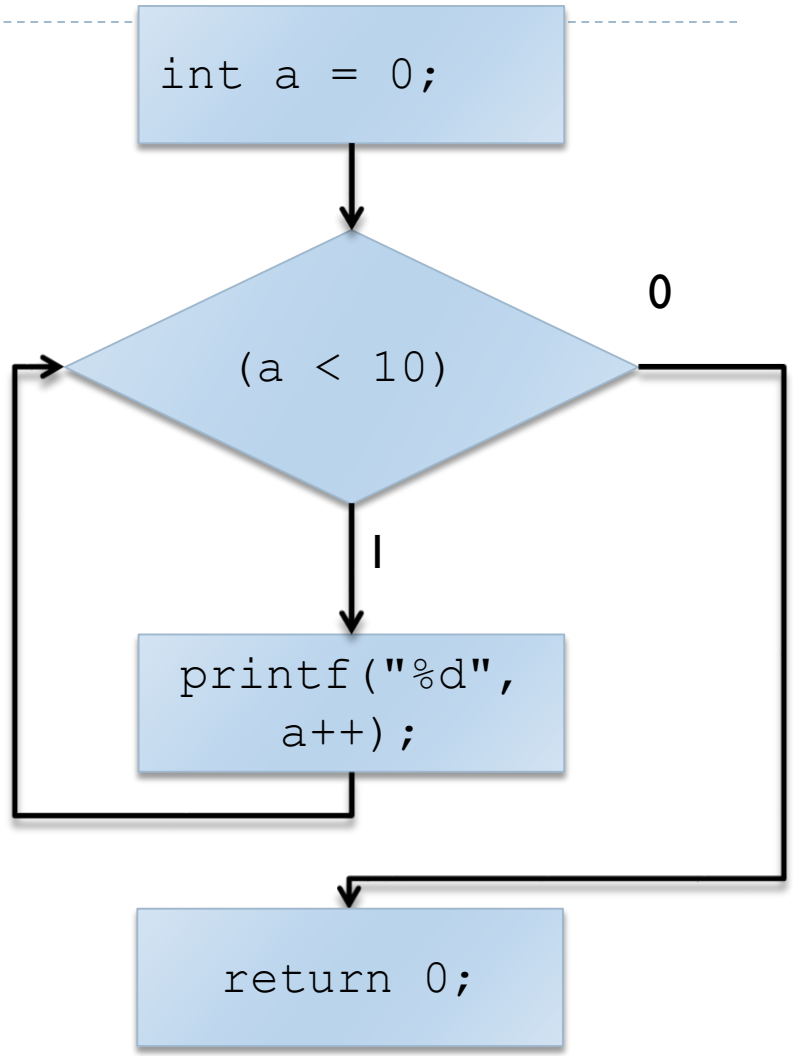
```
while(boolean_ifade)
        cümlecik;
```

```
#include <stdio.h>
int main(){

    int a = 0;

    while (a < 10)
        printf("%d", a++);

    return 0;
}
```



While Döngüsü

```
while(boolean_ifade) {

    cümlecik(ler);

}
```

```
/* lokal degisken tanimi*/
int a = 1;

/* while-dongu blogu */
while(a < 100) {
      /* sart dogru iken,
     asagidakileri calistir*/

   printf("a nin degeri: %d\n", a);
   a++;
}
```

While Döngüsü

- Bir tamsayının faktoriyelini bulan bir program yazınız:
  - Girişler?
  - Çıkışlar?
  - Algoritma?

While Döngüsü

- Bir tamsayının faktoriyelini bulan program:

```
#include <stdio.h>
int main(){
    int a = 18;
    int i = 0;
    long int fact = 1;
    while(i < a){
        i++;
        fact = fact * i;
    }
    printf("%d nin faktoriyeli: %d.\n", a, fact);
    return 0;
}
```

Do-while Döngüleri

- Önce cümlecikler çalıştırılır sonra şart (boolean ifade) kontrol edilir.
- Sonundaki noktalı virgüle dikkat!

```
do {
    cümlecik(ler);

} while( boolean_ifade) ;
```

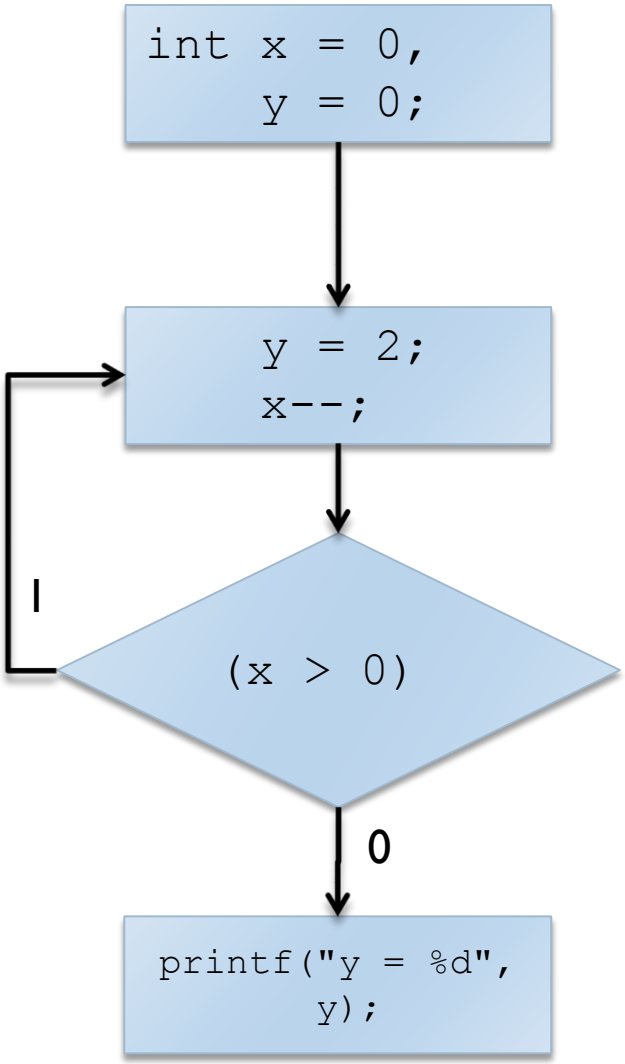
- Önce şart (boolean ifade) kontrol edilir sonra cümlecikler çalıştırılır.
- Noktalı virgül yok.

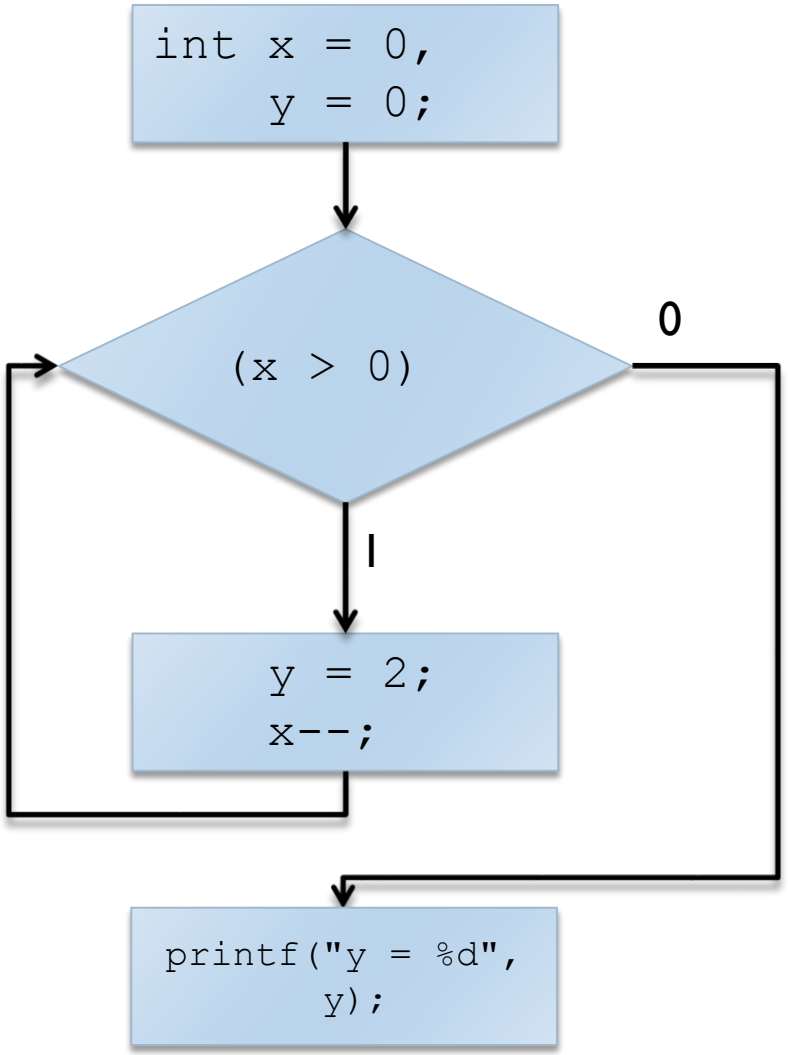
```
while(boolean_ifade) {
   cümlecik(ler);
}
```

```
int x = 0, y = 0;
do {
    y = 2;
    x--;
}while (x > 0);
printf("y = %d", y);
```

```
int x = 0, y = 0;
while (x > 0){
    y = 2;
    x--;
}
printf("y = %d", y);
```

Akış Şemaları





For Döngüsü

```
for (başlatma; boolean_ifade; artırma/azaltma) {
   cümlecik(ler);
}
```

```
int a;

/* for dongusu*/
for(a = 10; a < 100; a++){
   printf("a nin degeri: %d\n", a);
}
```

```
for(float t = 1.7;  t < 3.5; t = t + 0.1 ){
    printf("%f\n", t);
}
```

For Döngüsü

```
for (başlatma; boolean_ifade; artırma/azaltma) {
   cümlecik(ler);
}
```

```
int a;

/* for dongusu*/
for(a = 10; a < 100; a++){
   printf("a nin degeri: %d\n", a);
}
```

```
int a;
a = 10;
while( a < 100){
   printf("a nin degeri: %d\n", a);
   a++;
}
```

Örnek

- 1 den 100'e kadar olan tek sayıların toplamını bulan program.

Örnek

- 1 den 100'e kadar olan tek sayıların toplamını yazdıran kod parçası

```
int sum = 0;
for (int i = 1; i <= 100; i++){
    if (i%2 != 0){
        sum += i;
    }
}
printf("toplam: %d", sum);
```

Sonsuz döngüler

- Eğer şart her zaman doğru ise program sonsuz döngüye girer.

```
for( ; ; ){
    …
}
```

```
while(1){
    …
}
```

İç içe Döngüler

- Çarpım tablosunu yazdıran bir program yazınız.

İç içe döngüler

- Çarpım tablosunu yazdıran basit bir program.

```
#include <stdio.h>

int main(){

    for(int i=1; i<10; i++){
        for(int j=1; j<10; j++){
            printf("%d\t", i*j);
        }
        printf("\n");
    }
    return 0;
}
```

Örnek

- Verilen iki sayının en büyük ortak bölenini (EBOB) bulan program yazınız.
  - Giriş: a ve b
  - Çıkış: ebob(a, b)
  - Euclid Algoritması
    - ebob(a, 0) = 0
    - ebob(a, b) = ebob(b, a mod b)

break anahtar kelimesi

- **break;** cümleciği switch ve döngü bloklarının çalışmasını bitirir.
- Program akışı bloğu takip eden ilk cümlecikle devam eder.

```
int q;
while (1){
    scanf("%d", &q);
    if (q < 0)
        break;
}
```

Continue anahtar kelimesi

- continue; cümleciği for, while , veya do-while döngülerinde o anki iterasyon adımının atlanması için kullanılır.

```
int q;
while (1){
    scanf("%d", &q);
    if (q < 0){
        printf("sadece pozitif sayilar");
        continue;
    }
}
```

goto anahtar kelimesi

- Programda etiketle belirtilen her hangi bir noktaya atlamak için kullanılır.
- Çok ihtiyacınız olmadığı sürece kullanmayınız.

```
int i = 0;
labelA:
    i++;
    if (i < 5){
        printf("%d\n",i);
        goto labelA;
    }
    else
        goto labelB;
labelB:
    printf("Label B");
```

Akış Kontrol Özet

- If ve switch cümleleri, for, while, do-while döngüleri
- break, continue, goto anahtar kelimeleri

Negatif Sayıların Temsili

- ~ 1'e tamamlayan (1's complement)
  - Herbir bitin tersi alınır.
  - 8 bit-binary formda 5: 0000 0101
    - ~(5) = 1111 1010
- 8 bit bir sayının 2'ye tamamlayanı (2's complement), 2^8 *den o sayının çıkarılması ile bulunur:*
  - Ayrıca sayının 1'e tamamlayanına 1 eklenerekde bulunabilir
    - 5 in 2'ye tamamlayanı
      - ~(5) + 1 = 1111 1010 + 1 = 1111 1011
- Negatif sayılar bilgisayarda mutlak değerlerinin 2'ye tamamlayanları şeklinde temsil edilirler:
  - -5 = ~(5) + 1 = 1111 1010 + 1 = 1111 1011

Binary (bit) İşlemler

- &
  - And (ve)
- |
  - Or (veya)
- ^
  - Xor
- >>, <<
  - Sağ ve sola kaydırma

p	q	p & q	p | q	p ^ q
0	0	0	0	0
0	1	0	1	1
1	1	1	1	0
1	0	0	1	1

Haftaya

- Diziler(Arrays)