

BLM111 Programlama Dilleri I

Hafta 9

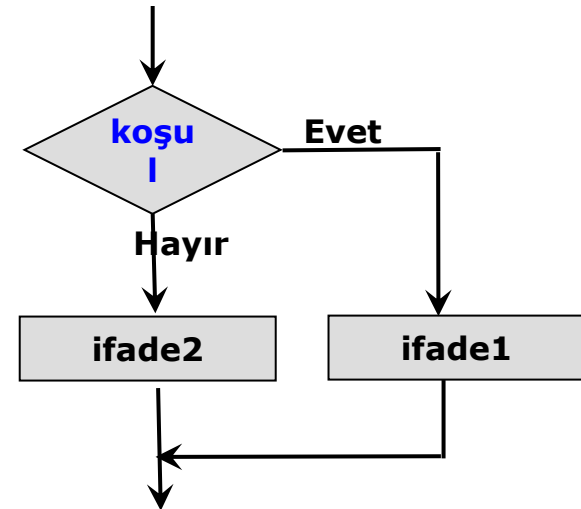
C Dilinde Kontrol ve Döngü Yapıları

Mehmet Zahid YILDIRIM

# Karar (Karşılaştırma) Komutları

- ▶ Koşulların kontrolünde kullanılan komutlardır.
- ▶ Koşulların doğru olup olmamasına (sağlanıp sağlanamamasına) göre işlem akışını yönlendirirler.

```
if (koşul)
    ifade1;
[else
    ifade2;]
```



- ▶ Eğer koşul doğru ise **true** (1) durum1 çalışacak değilse durum2 çalışacak.

# if-else Örnekler

```
int finalNotu;  
  
printf("Final notunu girin: ");  
scanf("%d", &finalNotu);  
if (finalNotu >= 45)  
    printf("Geçti \n");
```

```
int finalNotu;  
  
printf("Final notunu girin: ");  
scanf("%d", &finalNotu);  
if (finalNotu >= 45)  
    printf("Geçti \n");  
else  
    printf("Kaldı!\n");
```

# Eğer birden fazla durum işleteceksek ne yapmalıyız?

```
int finalNotu;  
...  
if(finalNotu >= 45)  
{  
    printf("Geçti!\n");  
    printf("Tebrikler!\n");  
}  
else  
{  
    printf("Kaldı!\n");  
    printf("Daha iyi çalış.\n");  
} /* end-else */
```

**blok**  
(birleşmiş  
durumlar)

# Süslü Parantezlerin Yeri

- ▶ Süslü parantezlerin yeri bir stil sorunudur
  - Compiler için bir sorun değildir.

```
int finalNotu;  
...  
if(finalNotu >= 45) {  
    printf("Geçti!\n");  
    printf("Tebrikler!\n");  
} else {  
    printf("Kaldı!\n");  
    printf("Daha iyi çalış.\n");  
} /* end-else */
```

# Lojik Operatörlerini Kullanmak

- ▶ if deyimlerinin içine lojik operatörler kullanılabilir.

```
/* If a eşittir 4 VEYA a eşittir 10 */  
if (a == 4 || a == 10) {  
    ...  
} else {  
    ...  
} /* end-else */
```

```
/* x 2 VE 20 arasında mı */  
if (x >= 2 && x <= 20) {  
    ...  
} /* end-if */
```

```
/* y 20 den büyük VE x 30 a eşit değil mi */  
if (y > 20 && x != 30) {  
    ...  
} /* end-if */
```

# Lojik Kademeli if Deyimleri

- ▶ Bazen birden fazla koşulu test etmek isteriz, ta ki biri sağlanana kadar.
  - Örneğin “n” in 0 a eşit , 0 dan büyük ve 0 dan küçük olmasını test etmek istiyoruz.

```
if (n < 0)      printf("n < 0\n");  
else {  
    if (n == 0) printf("n == 0\n");  
    else       printf("n > 0\n");  
} /* end-else */
```

- İkinci if deyimini else içinde kullanmaktansa onun yerine aşağıdaki gibi kullanabiliriz. Kademeli if olarak adlandırılıyor.

```
if (n < 0)          printf("n < 0\n");  
else if (n == 0)    printf("n == 0\n");  
else                printf("n > 0\n");
```

# Kademeli if Syntax

```
if (durum1)
    ifade1;
else if (durum2)
    ifade2;
else if (durum3)
    ifade3;
...
else
    ifadeN;
```



# Kademeli if Örneği

```
int finalNotu;  
...  
if (finalNotu >= 90)  
    printf("Geçti: Notun A \n");  
else if (finalNotu >= 85)  
    printf("Geçti: Notun A- \n");  
else if (finalNotu >= 80)  
    printf("Geçti: Notun B+ \n");  
else if (finalNotu >= 75)  
    printf("Geçti: Notun B \n");  
else if (finalNotu >= 70)  
    printf("Geçti: Notun B- \n");  
else if (finalNotu >= 55)  
    printf("Geçti: Notun C \n");  
else  
    printf("Kaldı \n");
```

# Koşul Operatörü ?

- Koşulun durumuna göre ilgili değeri veya işlem sonucunu belirtilen değişkene aktarır.

```
if (koşul)
    ifade1;
else
    ifade2;
```

- Bu ifadeyi daha kısa yazabilecek bir operatör var
  - **koşul ? ifade1 : ifade2**
- Örnekler:

```
min = (a < b) ? a : b;
```

min a ve b den minimum olanını alır.

```
a = (b >= 0) ? b : -b;
```

a b'nin mutlak değerini alır.

# switch Deyimi

## ► Seçimli form yapısıdır.

- Bu durumda kademeli if leri aşağıdaki gibi kullanabiliriz.

```
if (not == 5)      printf("Mükemmel\n");  
else if (not == 4) printf("iyi\n");  
else if (not == 3) printf("geçer\n");  
else if (not == 2) printf("zayıf\n");  
else if (not == 1) printf("kalır\n");  
else              printf("geçersiz not\n");
```

## ► Bir alternatif olarak kademeli if deyimlerinin yerine C switch yapısını destekliyor.

# switch Deyimi Örnek

- Önceki kodun kademeli if yerine switch ile yeniden yazımı

```
switch(not) {  
    case 5: printf("Mükemmel\n");  
            break;  
    case 4: printf("İyi");  
            break;  
    case 3: printf("Geçer\n");  
            break;  
    case 2: printf("Zayıf\n");  
            break;  
    case 1: printf("Kalır\n");  
            break;  
    default: printf("Geçersiz not\n");  
            break;  
} /* end-switch */
```

# switch Deyimi Sözdizimi

- İfadedeki koşula göre seçeneklerden bir tanesini çalıştırır.

```
switch (koşul) {  
  case değer1:  
    durum1;  
    ...  
    [break;  
  case değer2:  
    durum2;  
    ...  
    [break;  
  default:  
    durumN;  
    ...  
    [break;  
} /* end-switch */
```

# switch Deyimi Örnek

```
char operator;
int sonuc;
int deger;
...
switch(operator) {
case '+':
    sonuc += deger;
    break;
case '-':
    sonuc -= deger;
    break;
case '*':
    sonuc *= deger;
    break;
case '/':
    if(deger == 0) {
        printf("Error: sifıra bölme hatası \n");
        printf("        işlem iptal edildi \n");
    }
    else
        sonuc /= deger;
    break;
default:
    printf("bilinmeyen işlem\n");
    break;
} /* end-switch */
```

# switch içindeki break

- ▶ “break” bizi switch bloğunun dışına atar.
- ▶ Eğer **break** unutulur veya yazılmazsa kontrol bir sonraki **case** den devam eder.

```
switch(not) {  
    case 5: printf("Mükemmel\n");  
    case 4: printf("İyi\n");  
    case 3: printf("Geçer\n");  
    case 2: printf("Zayıf\n");  
    case 1: printf("Kalır\n");  
    default: printf("Geçersiz not\n");  
} /* end-switch */
```

- ▶ If not == 3, aşağıdaki yazılacak
  - Geçer
  - Zayıf
  - Kalır
  - Geçersiz not

# Döngü Komutları

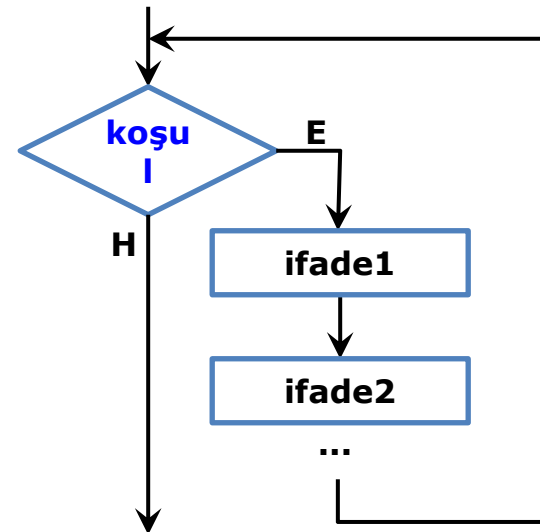
- ▶ Ardışık veya tekrarlı işlemlerin yapılmasını sağlayan komutlardır.
- ▶ C programlama dilinde üç döngü yapısı bulunmaktadır.
  - **while** döngüleri
  - **do-while** döngüleri
  - **for** döngüleri
- ▶ Aşağıdaki yardımcı ifadeler döngülerin içerisinde çıkarmak için kullanılabilir.
  - break
  - continue



# while İfadesi

- ▶ C dilindeki ön koşullu döngüdür.
- ▶ Verilen koşul sağlandığı (doğru olduğu) sürece döngü içindeki işlemler gerçekleştirilir.
- ▶ Döngünün kaç defa çalışacağı bilinmediği durumlarda kullanılır.

```
while(koşul)
{
    ifade1;
    ifade2;
    ...
}
```



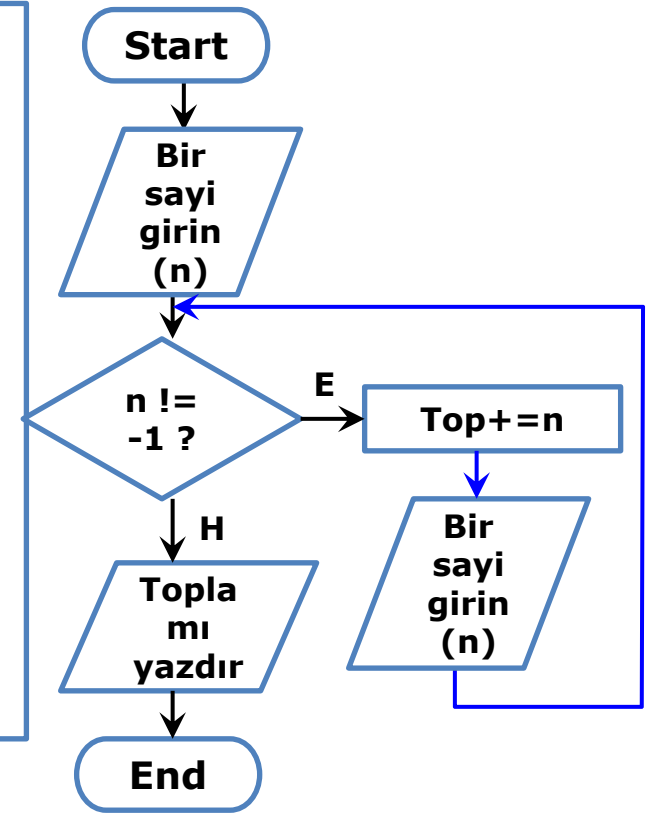
# Klavyeden Girilen Sayıların Toplamını Hesaplama

```
int toplam = 0;
int n;

printf("Bir sayi girin(durmak icin -1):
");
scanf("%d", &n);

while (n != -1){
    toplam += n;
    printf("sonraki sayi(durmak icin -1):");
    scanf("%d", &n);
}

printf("Toplam =%d\n", toplam);
```



# while Örnek

```
int i = 0;

printf("C programlamayı nasıl buldun?\n");
while(i < 10)
{
    printf("C Programlama çok zevkli!\n");
    i++;
}
```

- ▶ 10 defa tekrarlar (0 dan 9'a kadar)
- ▶ Aynı mesajı 10 defa yazar

# while Örnek

```
int i = 20;

printf("C programlamayı nasıl buldun?\n");
while(i < 10)
{
    printf("Programlama çok zevkli!\n");
    i++;
} /* end-while */
```

- ▶ 0 defa tekrarlar (i = 20, 10'dan küçük değil)
- ▶ Hiç mesaj yazmayacak

# while Örnek

## ► Problem: $a^n$ işlemini hesapla

```
int i;  
int sayi;  
int us;  
int sonuc=1;  
  
printf("sayi girin: ");  
scanf("%d", &sayi);  
printf("üs girin: ");  
scanf("%d", &us);  
  
i = 1;  
while (i<= us){  
    sonuc *= sayi;  
    i++;  
} /*end-while*/  
printf("sayi^us =  
%d\n", sonuc);
```

```
int i;  
int sayi;  
int us;  
int sonuc=1;  
  
printf("sayi girin: ");  
scanf("%d", &sayi);  
printf("üs girin: ");  
scanf("%d", &us);  
  
i = 1;  
while (i++<= us)  
    sonuc *= sayi;  
  
printf("sayi^us =  
%d\n", sonuc);
```

# while Örnek

- Problem:  $1+2+3+4+\dots+N$  işlemini hesapla

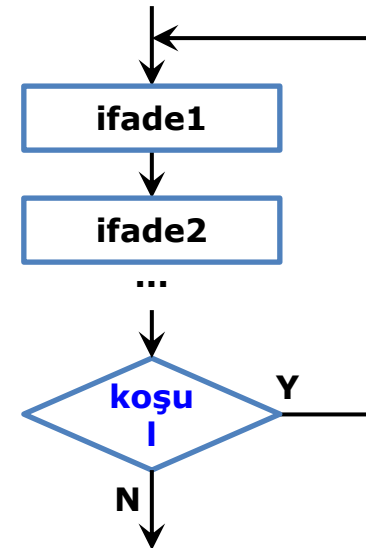
```
int i;  
int n;  
int top = 0;  
  
printf("\n girin:");  
scanf("%d", &n);  
i = 1;  
while (i<= n){  
    top += i;  
    i++;  
} /* end-while */  
printf("Toplam: %d\n", top);
```

```
int i;  
int n;  
int top = 0;  
  
printf("\n girin: ");  
scanf("%d", &n);  
i = 1;  
while (i<= n)  
    top += i++;  
  
printf("Toplam: %d\n", top);
```

# do while İfadesi

- ▶ Son koşullu döngüdür.
- ▶ "while" ile belirtilen koşul sağlandığı (doğru olduğu) sürece döngüdeki işlemler yapılır.
- ▶ Bazı durumlarda döngü bir kere çalıştıktan sonra devam edip etmemeye karar vermek isteriz. Bu durumlarda "do while" döngüsü kullanılır.
- ▶ Döngünün gövdesi en az bir kere çalışmaktadır.

```
do
{
    ifade1;
    ifade2;
    ...
} while(koşul);
```



# do while Örnek

```
int i = 0;

printf("C Programlamayı nasıl buldun?\n");
do {
    printf("Programlama çok zevkli!\n");
    i++;
} while(i < 10);
```

- ▶ 10 defa tekrarlar (0 dan 9 a kadar)
- ▶ Aynı mesajı 10 defa tekrarlar



# do while Örnek

```
int i = 20;

printf("C Programlamayı nasıl buldun?\n");
do {
    printf("Programlama çok zevkli!\n");
    i++;
} while(i < 10);
```

- ▶ 1 kere tekrarlar (i = 20 için)
- ▶ Aynı mesajı bir kere yazar

# while ile do while Arasındaki Fark Nedir?

while	do while
Döngüye koymak için kontrol eder	Döngüden çıkarmak için kontrol eder.
Döngü çalışabilir veya çalışmaz	Döngü en az bir kere çalışır

# for Döngüsü

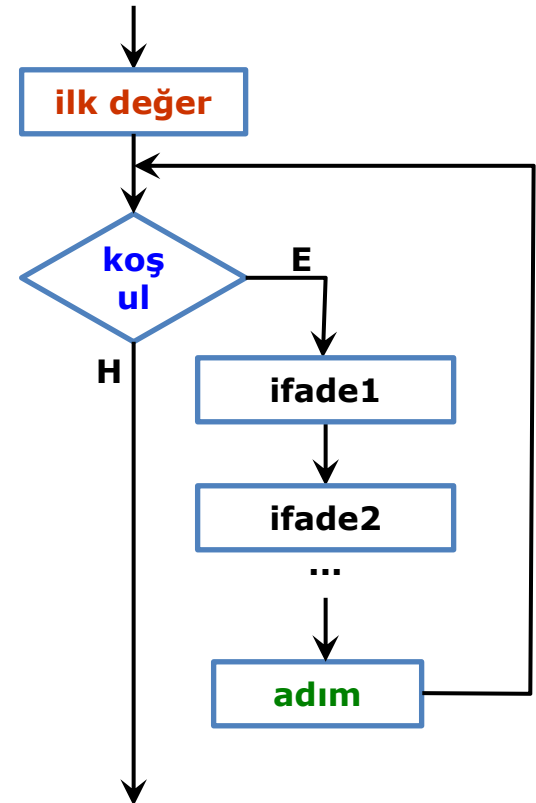
- ▶ Sayıcı döngü komutudur.
  - Örneğin döngünün N defa çalışmasını istiyoruzdur.
- ▶ Daha sık kullanılır.
- ▶ Koşul sağlandığı (doğru olduğu) sürece döngü bloğu işlemleri yapılır.

```
for(başlangıç değerleri ; koşul ; adım)
{
    ...
    işlemler
    ...
}
```

# for Akış Diyagramı ve while Eşiti

```
for (ilk değer; koşul; adım)  
{  
    ifade1;  
    ifade2;  
    ...  
}
```

```
ilk değer;  
while (koşul)  
{  
    ifade1;  
    ifade2;  
    ...  
    adım;  
}
```



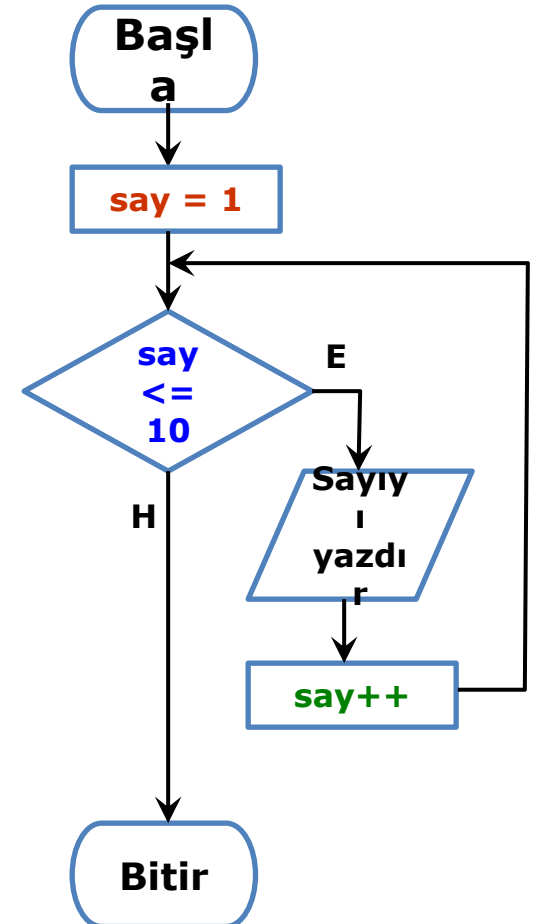
# for Örnek

- Problem: 1 den 10'a kadar sayıları yazdır

```
int say;  
for(say = 1; say <= 10; say++)  
{  
    printf("%d ", say);  
}  
printf("\n");
```

output:

1 2 3 4 5 6 7 8 9 10



# for İfadesinin Kullanımı

- ▶ for ifadesi genellikle bir değişkeni artırmak veya azaltmak için en iyi tercihtir.

```
for (i=0; i<N; i++) ...
```

- 0 dan N-1 e kadar sayar

```
for (i=1; i<=N; i++)  
...
```

- 1 den N e kadar sayar

```
for (i=N-1; i>=0; i--) ...
```

- N-1 den 0 a kadar sayar

```
for (i=N; i>=1; i--) ...
```

- N den 1 e kadar sayar

# for Örnek

- Problem:  $1+2+3+4+\dots+N$  işlemini hesapla

```
top = 0;  
for(i=1; i<=N; i++) {  
    top += i;  
}
```

- İlk değer ( $i=1$ ), kontrol ( $i\leq N$ ) ve değiştir ( $i++$ ) ifadelerin her biri opsiyoneldir ve yazılmayabilir.

```
top=0;  
i=1;  
for(; i<=N; i++) {  
    top += i;  
}
```

```
top=0;  
for(i=1; i<=N;) {  
    top += i++;  
}
```

```
top=0;  
for(; i<=N;) {  
    top += i++;  
}
```

# İç içe Döngüler

- ▶ Döngüleri bir biri içinde kullanabiliriz
  - Çoğu programlarda bu durum gerekiyor.
- ▶ Örnek: çarpım tablosunu yazdıralım

```
int i, j;

for (i=1; i <= 10; i++){
    printf("i: %d: ", i);
    for (j=1; j <= 10; j++){
        printf("%5d", i*j);
    } /* end-for-içerdeki */
    printf("\n");
} /* end-for-dışardaki */
```



# Döngü İçinde break & continue

- ▶ switch ifadesinde break kodunun bizi switch ifadesi dışına attığını görmüştük.
- ▶ Benzer bir şekilde, break döngü içinde kullanıldığında da bizi geçerli döngünün dışına atar.

```
int toplam = 0;
int n;

while (1){ /* sonsuz döngü*/
    printf("bir sayı gir (durmak için -1): ");
    scanf("%d", &n);

    if (n == -1) break; /* döngünün dışına çık */
    toplam += n;
} /* end-while */

printf("toplam=%d\n", toplam);
```

# break

- Bazı durumlarda döngünün ortasında döngüden çıkmak için kullanılır.

```
while (1){  
    printf("bir sayı girin veya bitirmek için 0:  
    ");  
    scanf("%d", &n);  
    if (n == 0) break;  
    printf("n=%d, n*n*n*=%d\n", n, n*n*n);  
} /* end-while */
```

# continue

- ▶ Kontrolü döngünün sonuna alır.
- ▶ Unutmayın, hala döngünün içindeyiz.
- ▶ “continue” basitçe, döngünün gövdesinde kullanılan yerden sonraki kısmı atlar ve kontrolü döngünün sonuna alır.

```
int i;  
int n = 0;  
int toplam = 0;  
  
while (n<10){  
    scanf("%d", &i);  
    if (i == 0) continue; /* (B) ye geçer */  
    n++; toplam += i;  
  
    /* (B) */  
} /* end-while */
```

# Sonsuz Döngüler

- Sınırsız sayıda tekrar eden döngüler.

```
while (1){  
    ...  
}
```

```
do{  
    ...  
} while (1);
```

```
for (;;) {  
    ...  
}
```

- Peki bu tür döngülerden nasıl çıkabiliriz?
  - Basitçe döngünün bir yerlerine “break” koyarak.

```
while (1){  
    printf("Bir sayı gir (durmak için 0):  
    ");  
    scanf("%d", &n);  
    if (n == 0) break;  
    printf("n=%d, n*n*n*=%d\n", n, n*n*n);  
} /* end-while */
```

# Virgül Operatörü


- ▶ Zaman zaman birkaç ifadeyi birleştirip bir tek ifade olarak yazmak isteriz.
  - Böyle durumlarda virgül operatörünü kullanırız.

```
ifade1, ifade2, ..., ifadeN;
```

```
i=1, j=2, k=i+j;
```

eşittir

```
((i=1), (j=2), (k=i+j));
```



- Değerlendirme soldan sağa yapılır
- İfadenin sonucu = k=i+j; yani 3

# Kaynaklar

- ▶ Doç. Dr. Fahri Vatansever, “Algoritma Geliştirme ve Programlamaya Giriş”, Seçkin Yayıncılık, 12. Baskı, 2015.
- ▶ J. G. Brookshear, “Computer Science: An Overview 10th Ed.”, Addison Wisley, 2009.
- ▶ Kaan Aslan, “A’dan Z’ye C Klavuzu 8. Basım”, Pusula Yayıncılık, 2002.
- ▶ Paul J. Deitel, “C How to Program”, Harvey Deitel.
- ▶ Bayram AKGÜL, C Programlama Ders notları