

Doğu Akdeniz Üniversitesi  
Bilgisayar ve Teknoloji Yüksek Okulu  
Bilgisayar Programcılığı Bölümü

**BTEP 102 – Veri Yapıları ve Programlama**

**Bölüm 2: Program Denetimi**



R. KANSOY

1

**Konu Başlıkları**

2. Kontrol Yapıları

2.1 Seçimli Yapı

2.1.1 Tekil Seçimli Yapı (if)

2.1.2 İkili Seçimli Yapı (if...else, if... else if)

2.1.3 ? Üçlü Şart Operatörü

2.1.4 Çoklu Seçimli Yapı (switch-case)

2.2 Tekrarlamalı Yapı (Döngüler)

2.2.1 for döngüsü

2.2.2 while döngüleri


2.2.3 do...while döngüleri

2.2.4 Döngülerden Çıkış (break) ve Devam (continue)

BTEP 102 – Veri Yapıları ve Programlama

2/29

2




## 2. Kontrol Yapıları

- ✓ Normal şartlar altında bir program içerisindeki çalıştırılabilir satırlar, yazılmış oldukları sırada, yani birinden sonra diğerinin çalışması şeklinde ilerler. Buna sıralı yürütme adı da verilmektedir.
- ✓ İşin akışı gereği, bazı durumlarda, bir sonraki satırdaki değil, daha farklı bir satırdaki programın çalıştırılmasına ihtiyaç duyulabilir.
- ✓ Programın ya da fonksiyonun çalışma akışını kontrol eden mekanizmalara kontrol yapıları (control structure) denilir.
- ✓ Üç çeşit kontrol yapısı mevcuttur;
  - *Sıralı (sequence)*
  - *Seçimli (selection)*
    - Tekil Seçimli Yapı
    - İkili Seçimli Yapı
    - Çoklu Seçimli Yapı
  - *Tekrarlı (repetition)*
- ✓ Şu ana kadar yalnızca sıralı yapıda yazılmış programlar üzerinde durduk. Bu asamada seçimli yapı üzerinde duracağız.

3/29

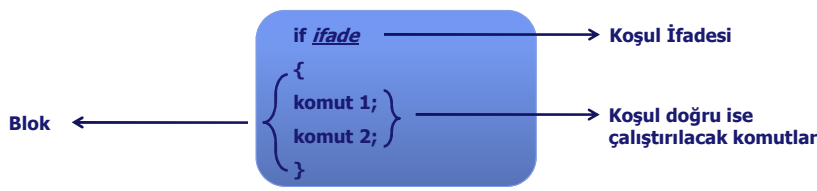
3



## 2.1 Seçimli Yapı

### 2.1.1 Tekil Seçimli Yapı (if)

- ✓ Verilen durum ya da koşula göre istenilen işlem ya da işlemleri gerçekleştirmek için kullanılır.
- ✓ if deyimi şu şekilde tanımlanır;
  - *if koşul deyimi;*
- ✓ Şarta ya da duruma bağlı olarak çalışması gereken birden fazla komut ya da çalıştırılabilir ifade mevcutsa blok açılır.



4/29

4

## 2.1 Seçimli Yapı

### 2.1.1 Tekil Seçimli Yapı (if)

✓ Bu yapıda if komutundan sonra gelen koşul ya da durum ifadesinin sonucu doğru (evet) olması halinde, if içinde belirtilen deyim (komut) çalışır. durum ya da koşula göre istenilen işlem ya da işlemleri gerçekleştirmek için kullanılır.

✓ Koşulun sonucu yanlış (hayır) ise, if içinde tanımlanan deyim işlem görmez, if komutundan sonra gelen komutlar çalışır.

```

graph TD
    Start(( )) --> Koşul{KOŞUL ?}
    Koşul -- Doğru --> Komut[KOMUT(lar)]
    Koşul -- Yanlış --> Exit(( ))
    Komut --> Exit
  
```

Şekil 2.1: if deyiminin çalışma biçimi

5/29

5

## 2.1 Seçimli Yapı

### 2.2.2 İkili Seçimli Yapı (if...else)

✓ Bu yapıda **if** komutundan sonra gelen koşul ya da durum ifadesinin sonucu doğru (true) ise şarttan sonra gelen komut(lar) çalışır.

✓ Koşul yanlış (false) ise **else**'den sonra gelen komut(lar) çalışır.

```

graph TD
    Start(( )) --> Koşul{KOŞUL ?}
    Koşul -- Doğru --> Komut1[KOMUT(lar)]
    Koşul -- Yanlış --> Komut2[KOMUT(lar)]
    Komut1 --> Exit(( ))
    Komut2 --> Exit
  
```

**if ifade**

```

{
  komut 1;
  komut 2;
  ...
  komut n;
}
else
{
  komut 1;
  komut 2;
  ...
  komut n;
}

```

Koşul ifadesinin neticesi "doğru" ise bu blok çalışır

Koşul ifadesinin neticesi "yanlış" ise bu blok çalışır

Şekil 2.2: if...else deyiminin çalışma biçimi

6/29

6

## 2.1 Seçimli Yapı

**if...else (örnek)**

**KOD 2.1** Girilen Sayının Tek-Çift Ayırımı

```
#include <stdio.h>
int main()
{
    int a;
    printf("Sayıyı giriniz:");
    scanf("%d", &a);
    if( a % 2 ==0 ) /*2 ye bölündüğünde kalan 0 ise çifttir */
        printf("Girilen sayı çifttir");
    else
        printf("Girilen sayı tektir");
    return 0;
}
```

7/29

7

## 2.1 Seçimli Yapı

**İç içe if Deyimleri (if...else if)**


```
if (koşul_1)
{
    komut(lar);
}
else if (koşul_2)
{
    komut(lar);
}
else if (koşul_3)
{
    komut(lar);
}
else
{
    komut(lar);
}
```

```

graph TD
    Start(( )) --> D1{KOŞUL ?}
    D1 -- doğru --> C1[KOMUT(lar)]
    D1 -- yanlış --> D2{KOŞUL ?}
    D2 -- doğru --> C2[KOMUT(lar)]
    D2 -- yanlış --> D3{KOŞUL ?}
    D3 -- doğru --> C3[KOMUT(lar)]
    D3 -- yanlış --> C4[KOMUT(lar)]
    C1 --> Join(( ))
    C2 --> Join
    C3 --> Join
    C4 --> Join
    Join --> End(( ))
    
```

Şekil 2.3: if...else if deyiminin çalışma biçimi 8/29

8



## 2.1 Seçimli Yapı

### if...else if (örnek)

**KOD 2.2 Sayısal Değerin Pozitif, Negatif ya da Sıfır Olup Olmadığını Saptama**

```
#include <stdio.h>
int sayi;

main()
{
    printf("Bir Sayı Giriniz:");
    scanf("%d", &sayi);


    if (sayi>0)
        printf("Pozitif");
    else if (sayi<0)
        printf("Negatif");
    else
        printf("Sıfır");
}
```

**Sonuç:**  
Bir Sayı Giriniz: -77  
Negatif

9/29

BTEP 102 – Veri Yapıları ve Programlama

9



## 2.1 Seçimli Yapı

### if...else if (örnek)


**KOD 2.3 vize ve final notuna göre harf notunun hesaplanması**

```
#include <stdio.h>
int main()
{
    int vize,final;
    float ortalama;
    char harf;
    printf("\n\nÖğrencinin vize notu: ");
    scanf("%d",&vize);
    printf("\n\nÖğrencinin final notu: ");
    scanf("%d",&final);
    ortalama=vize*0.4+final*0.6;
    if(ortalama<50)
        harf='F';
    else if(ortalama<60)
        harf='D';
    else if(ortalama<70)
        harf='C';
    else if(ortalama<80)
        harf='B';
    else
        harf='A';
    printf("-----\n");
    printf("Başarı ortalaması = %.2f\n",ortalama);
    printf("Öğrencinin harf notu = %c\n",harf);
    return 0;
}
```

10/29

BTEP 102 – Veri Yapıları ve Programlama

10

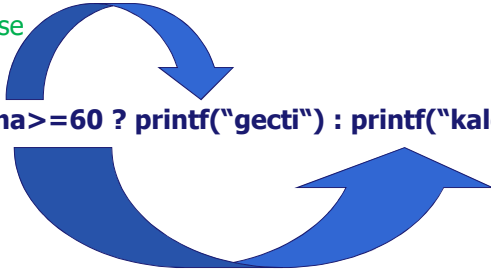


## 2.1 Seçimli Yapı

### 2.1.3 ? Üçlü Şart Operatörü

- ✓ C programlarında if...else deyimleri yerine bazı özel durumlarda ? operatörü kullanılarak sonuçlara ulaşılabilir.
- ✓ Bu operatör şu şekilde kullanılır:
  - ifade1? ifade2:ifade3;
- ✓ ifade1'in sonucu 0 dan farklı bir değere sahipse yani doğru ise ifade2, sıfıra eşit ise yani sonuç yanlış ifade3 gerçekleşir.

Doğru ise




Yanlış ise

**Ortalama >= 60 ? printf("geçti") : printf("kaldı");**

BTEP 102 – Veri Yapıları ve Programlama

11/29

11



## 2.1 Seçimli Yapı

### ? Üçlü Şart Operatörü (örnek)

**KOD 2.4 ? Operatörünün Kullanımı**

```

/*Bu program havanın soğuk olup olmadığına karar verir*/

#include <stdio.h>

int main()
{
    int a;
    printf("Hava derecesini giriniz:");
    scanf("%d", &a);
    a <= 10 ? Printf("hava soğuk") : printf("hava o kadar da soğuk değil");
    return 0;
}
```

BTEP 102 – Veri Yapıları ve Programlama

12/29

12

## 2.1 Seçimli Yapı

### 2.1.4 Çoklu Seçimli Yapı (switch-case)

✓ Eğer bir değişkenin değeri belirli sabitlerle karşılaştırılacak ve bunun sonucunda farklı işlemler yapılacak ise if deyimi yerine switch deyimi kullanılabilir.

✓ Bu deyim şu şekilde tanımlanmaktadır:

```

switch (değişken) {
    case <değer 1> : komutlar 1;
    [break;]

    case <değer 2> : komutlar 2;
    [break;]

    :

    case <değer n> : komutlar N;
    [break;]
}

```

Şekil 2.4: switch çalışma biçimi

13/29

13

## 2.1 Seçimli Yapı

### switch-case (örnek)

BTEP 102 – Veri Yapıları ve Programlama

```

KOD 2.5 Girilen İl Koduna Göre Şehir İsmi Görüntüleme

#include <stdio.h>
main()
{
    int kod;
    printf("İl Trafik Kodu:");
    scanf("%d", &kod);

    switch (kod)
    {
        case 6:
            printf("Ankara");
            break;

        case 34:
            printf("İstanbul");
            break;

        case 35:
            printf("İzmir");
            break;

        default:
            printf("Diğer");
    }
}

```

14/29

14

## 2.2 Tekrarlamalı Yapı (Döngüler)

- ✓ Bir ifade kümesinin tekrarlanması, yani birden fazla çalıştırılması işlemine **döngü** (loop) denilir.
- ✓ C programında döngü işlemleri aşağıda gösterildiği gibi, farklı biçimlerde gerçekleştirilmektedir;
  - for döngüsü
  - while döngüleri
  - do...while

15/29

15

## 2.2 Tekrarlamalı Yapı (Döngüler)

### 2.2.1 for döngüsü

- ✓ C programı içinde bir ya da daha fazla sayıda deyimın belirli bir koşulun gerçekleşmesine dek tekrarlanması söz konusu ise **for** deyimi kullanılır.
- ✓ Koşul gerçekleştiğinde, yani doğruluk değeri “yanlış” olduğunda döngü terk edilerek bir sonraki deyim işlem görmeye başlar.
- ✓ Bu deyim şu şekilde tanımlanmaktadır;
  - For (başlangıç ifadesi; koşul; artırma/azaltma ifadesi)
 

```
{
deyimler;
}
```

```

graph TD
    Start(( )) --> A[Değer Atama]
    A --> B{KOŞUL}
    B -- True --> C[Komutlar]
    C --> D[Artırma / Azaltma]
    D --> B
    B -- False --> Exit(( ))
  
```

Şekil 2.5: for deyiminin çalışma biçimi

16/29

16



## 2.2 Tekrarlamalı Yapı (Döngüler)

### for döngüsü (örnek)

KOD 2.6 2'den 100'e Kadar Olan Çift Sayıların Toplamı

```
#include<stdio.h>
int main(void)
{
    int toplam = 0; /* Toplamı tanımla ve ilk değerini sıfırla */
    int sayi; /* sayı adında kontrol degiskenini tanımla */
    for (sayi = 2; sayi <= 100; sayi += 2) /* sayı ilk degerini 2 yap */
        /* sayı degerini her tekrarda 2 arttır */
    {
        /* sayı degeri 100'e esit ve küçük olduğu sürece kod bloğunu işlet */
        toplam += sayi; /* sayıyı toplama ekle */
    }
    printf("Toplam = %d",toplam); /* toplamı ekrana yazdır */
    return 0; /* fonksiyondan çık */
}
```

BTEP 102 – Veri Yapıları ve Programlama

17/29

17

## 2.2 Tekrarlamalı Yapı (Döngüler)

### 2.2.2 while döngüsü

- ✓ Bir koşulun gerçekleşmesi durumunda belirli işlemlerin tekrarlanması söz konusu ise while döngülerinden yararlanılır.
- ✓ Koşul denetimi döngü bloğunun başında yapılır.
- ✓ While döngüsü içindeki deyimler, belirlenen koşul “doğru” olduğu sürece çalışacak olan deyimlerdir.
- ✓ Eğer while ifadesine başlamadan önce koşulun değeri yanlış (0) ise, döngünün içi hiç çalıştırılmayacaktır.
- ✓ Döngü işlemleri esnasında koşulun doğruluk derecesi “yanlış” olduğu anda döngü terk edilir.
- ✓ While döngüsü şu şekilde tanımlanmaktadır:
  - while (denetim ifadesi)
 


```
{
                        Deyim;
                        ...
                    }
```

Şekil 2.6: while deyiminin çalışma biçimi

BTEP 102 – Veri Yapıları ve Programlama

18/29

18



## 2.2 Tekrarlamalı Yapı (Döngüler)

### while döngüsü (örnek)


KOD 2.7 1'den 10'a Kadar Olan Sayıların Ekrana Yazdırılması

#include <stdio.h>	Sonuç:
/* main fonksiyonuyla program çalışmaya başlar */	1
int main(void)	2
{	3
int sayac = 1; /* sayacı tanımla ve ilk deger ata */	4
while (sayac <= 10) /* Tekrarlama kosulu */	5
{	6
printf("%d\n",sayac); /* Sayac degerini ekrana yazdır */	7
++sayac; /* sayacı arttır */	8
} /* while döngüsü sonu */	9
return 0; /* Programın basarıyla tamamlandığını raporla */	10
}	

BTEP 102 – Veri Yapıları ve Programlama

19/29

19



## 2.2 Tekrarlamalı Yapı (Döngüler)

### Birbirine denk döngü ifadeleri: for ve while


for döngüsü ile	while döngüsü ile
#include <stdio.h>	#include <stdio.h>
int main()	int main()
{	{
int i;	int i;
for(i = 1; i <= 10; i++)	i=1;
printf("Ali\n");	while( i<=10 ){
}	printf("Ali\n");
	i = i+1;
	}
	}

BTEP 102 – Veri Yapıları ve Programlama

20/29

20

BTEP 102 – Veri Yapıları ve Programlama



## 2.2 Tekrarlamalı Yapı (Döngüler)

### 2.2.3 do...while döngüsü

- ✓ **while** tekrarlama ifadesine çok benzerdir.
- ✓ **while** tekrarlama ifadesinde döngü devam kosulu, döngü blogundaki kod satırları isletilmeden test edilir. Kosul saglanmıyorsa bloktaki kodlar çalıştırılmadan döngüden çıkılır.
- ✓ **do...while** tekrarlama ifadesinde döngü devam kosulu, döngü blogundaki kod satırları gerçekleştirildikten sonra test edilir. Böylece, kod blogunun **en az bir kere çalıştırılması garanti edilir.**

21/29

21

BTEP 102 – Veri Yapıları ve Programlama



## 2.2 Tekrarlamalı Yapı (Döngüler)

### 2.2.3 do...while döngüsü

- ✓ Bu deyim şu şekilde tanımlanır:


```
do
{
Deyim
....
} while(kontrol ifadesi)
```
- ✓ Tekrarlama ifadelerinin tümünde, sonsuz döngü oluşmaması için, koşulun nihai olarak bozulması garanti edilmelidir.

22/29



Şekil 2.7: do...while deyiminin çalışma biçimi

22



## 2.2 Tekrarlamalı Yapı (Döngüler)

### do...while (örnek)

**KOD 2.8 do...while Döngüsü**

```

/* do-while kullanımı */
#include <stdio.h>
main()
{
    int sayi;
    do
    {
        printf("Bir sayı girin : ");
        scanf("%d",&sayi);
        printf("Bu sayının iki katı : %d\n",2*sayi);
    } while( sayi>0 ); /* koşul */
    printf("Çevrim sona erdi.");
    return 0;
}

```


**Sonuç:**

Bir sayı girin : 2 Bu sayının iki katı : 4  
 Bir sayı girin : 5 Bu sayının iki katı : 10  
 Bir sayı girin : 9 Bu sayının iki katı : 18  
 Bir sayı girin : 0 Bu sayının iki katı : 0  
 Çevrim sona erdi.

BTEP 102 – Veri Yapıları ve Programlama

23/29

23



## 2.2 Tekrarlamalı Yapı (Döngüler)


### Birbirine denk döngü ifadeleri: while ve do...while

KOD 2.9: while döngüsü	KOD 2.10: do...while döngüsü	SONUÇ:
#include <"stdio.h">	#include <"stdio.h">	1
int main(void)	int main(void)	2
{	{	3
int sayac = 1;	int sayac = 1;	4
while (sayac <= 10)	do	5
{	{	6
printf("%d\n",sayac);	printf("%d\n",sayac);	7
++sayac;	++sayac;	8
}	} while (sayac <= 10)	9
return 0;	return 0;	10
}	}	

BTEP 102 – Veri Yapıları ve Programlama

24/29

24



## 2.2 Tekrarlamalı Yapı (Döngüler)


### İç-İçe Döngüler (Nested Loops)

- ✓ Bir döngü içerisinde başka bir döngü bulunuyorsa, bu tür yapılara iç-içe döngüler denir.
- ✓ Bu durumda içteki döngü dıştaki döngünün her adımında yeniden çalıştırılacaktır.

KOD 2.11 İç-içe for Döngüsü	Sonuç:
<pre>#include &lt;stdio.h&gt; int main(void) {     int i,j;     for (i=1; i&lt;=5; i++) /*dıştaki döngü*/     {         for (j=1; j&lt;=i; j++) /*içteki döngü*/         {             printf("*");         }         printf("\n");     } }</pre>	<pre>* ** *** **** *****</pre>

25/29

25



## 2.2 Tekrarlamalı Yapı (Döngüler)


### 2.2.4 Döngülerden Çıkış (**break;**) ve Devam (**continue;**)

- ✓ **break** ve **continue** ifadeleri akışın kontrolünü değiştirmek için kullanılır.
- ✓ **break** ifadesi, kullanıldığı **while**, **do...while**, **for** veya **switch** ifadesinden ani çıkış yapmak için kullanılır.
- ✓ Yürütme, ifadeden hemen sonra gelen satırla devam eder.
- ✓ Bu ifade **break;** şeklinde tanımlanır.

KOD 2.12: break deyiminin uygulaması	Sonuç:
<pre>#include &lt;stdio.h&gt; int main() {     int i;     for (i=0; i&lt;=5; i++)     {         if(i == 3) /*i'nin değeri 3 ise */             break; /*döngü dışına çık*/         printf("i= %d \n",i);     }     return 0; }</pre>	<pre>i=0 i=1 i=2</pre> <p>Break deyimi işlem sırasını döngü dışına aktarır.</p>

26/29

26




## 2.2 Tekrarlamalı Yapı (Döngüler)

### continue; ifadesi

- ✓ Bir döngüyü terk etmeden bir adımın atlanması söz konusu ise **continue;** ifadesi kullanılır.
- ✓ Bu ifade döngünün işlemlerini sona erdirmez, sadece bir sonraki döngü adımına geçilmesini sağlar.
- ✓ Bir başka deyişle **continue;** ifadesi, kullanıldığı **while**, **do...while**, **for** veya **switch** ifadesinde içinde kendinden sonraki satırları atlayarak, döngünün koşul satırına ilerler.
- ✓ Eğer **for** döngüsü kullanılıyorsa, işlem sırası bu ifadeye geldiğinde, bu ifadeden döngü sonuna kadar olan deyimler çalışmaz, döngü bir artırılarak sonraki döngüye geçer.
- ✓ Eğer **while** döngüsü kullanılıyorsa, **continue;** ifadesine sıra geldiğinde, döngü içinde bu ifadeden sonraki tüm deyimler atlanır ve koşul sağlandığı sürece döngüye devam edilir.

27/29

27




## 2.2 Tekrarlamalı Yapı (Döngüler)

### continue; (örnek)

KOD 2.13: for döngüsü ve continue	KOD 2.14: while döngüsü ve continue	SONUÇ:
<pre>#include &lt;stdio.h&gt; int main() {     int i;     for (i=0; i&lt;=5; i++)     {         if(i==3) /*i'nin değeri 3 ise */             continue; /* döngü başı yap*/         printf ("i = %d \n",i);     }     return 0; }</pre>	<pre>#include &lt;stdio.h&gt; int main() {     int i=0;     while (i&lt;=5)     {         if(i==3) /*i'nin değeri 3 ise */             continue; /* döngü başı yap*/         printf ("i = %d \n",i);         ++i;     }     return 0; }</pre>	<p><b>i=0</b></p> <p><b>i=1</b></p> <p><b>i=2</b></p> <p><b>i=4</b></p> <p><b>i=5</b></p>

28/29

28



## 2.2 Tekrarlamalı Yapı (Döngüler)

**break; ve continue; (örnek)**

KOD 2.15: break;	KOD 2.16: continue ;
<pre>#include &lt;stdio.h&gt; int main(void) {     int x;     for (x = 1; x &lt;= 10; x++) {         if (x == 5) {             break;         }         printf("%-3d", x);     }     printf("%d. adımda döngünün dışına çıkıldı", x);     return 0; }</pre>	<pre>#include &lt;stdio.h&gt; int main(void) {     int x;     for (x = 1; x &lt;= 10; x++) {         if (x == 5) {             continue;         }         printf("%-3d", x);     }     return 0; }</pre>
<p>Sonuç:</p> <p>1 2 3 4</p> <p>5. Adımda döngünün dışına çıkıldı</p>	<p>Sonuç:</p> <p>1 2 3 4 6 7 8 9 10</p>

BTEP 102 – Veri Yapıları ve Programlama

29/29