Algoritmalar ve Programlama I – BLM 103

Hafta 5: Seçim ve Kontrol Yapıları

Fenerbahçe Üniversitesi

5. Hafta İçeriği

- Durum Yapıları
  - Eğer (if) İfadesi
  - Eğer/Değilse (İf/Else) İfadesi
- Döngü Yapıları
  - While İfadesi
  - For İfadesi
  - Do-While İfadesi
- Diğer Kontrol Mekanizmaları
  - Switch
  - Break ve Continue İfadeleri

Kontrol Mekanizmaları

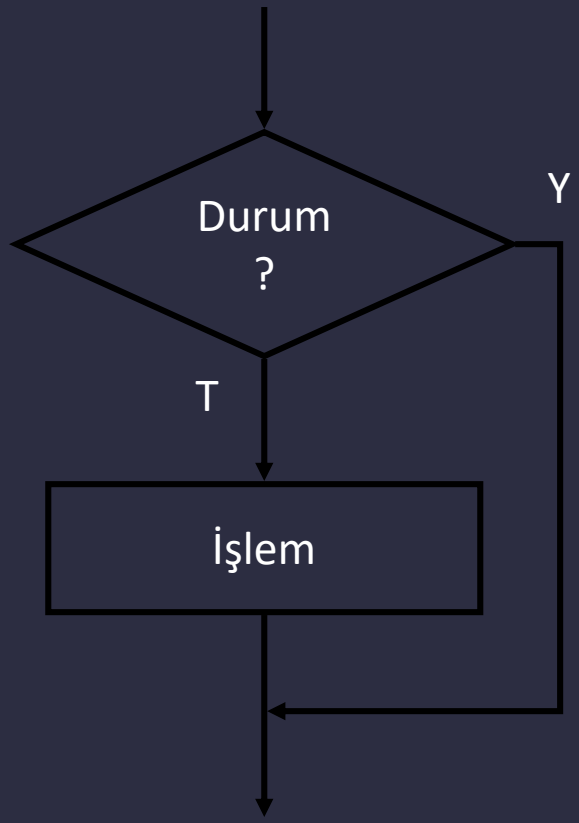
- Koşul İfadeleri
  - Bir koşula başlı olarak programın çeşitli kararlar vermesini sağlayan yapılardır.
  - `Eğer (if)`
  - `Eğer-Değilse(if-else)`
  - `Anahtar (switch)`

- Tekrar ifadeleri
  - Bir kod parçacığını birden çok defa çalıştırmak için kullanılan yapılardır.
  - `while`
  - `for`
  - `do-while`

Eğer (If)

- `if (durum)`
  `çalıştırılacak olan kod;`

Durum *Doğru veya Yanlış olarak hesaplanan bir ifadedir.*
İşlem durum'un doğru olması durumunda çalıştırılacak olan kod parçacığıdır.



Eğer (if) Örnekleri

- ```
  if (x <= 10)
     y = x * x + 5;
  ```

- ```
  if (x <= 10) {
     y = x * x + 5;
     z = (2 * y) / 3;
  }
  ```

Her iki satır, ifadenin doğru olduğu durumda çalıştırılacaktır

- ```
  if (x <= 10)
     y = x * x + 5;
     z = (2 * y) / 3;
  ```

İfadenin doğru olması durumunda, ilk satır çalıştırılacaktır. İkinci satır bir koşula bağlı değildir. Her zaman çalıştırılmaktadır.
  ```

Eğer (if) Örnekleri

- ```
  if (0 <= yas && yas <= 11)
     cocuk += 1;
  ```

- ```
  if (ay == 4 || ay == 6 ||
        ay == 9 || ay == 11)
     printf("Girilen ay 30 gündür.\n");
  ```

- ```
  if (x = 2)
     y = 5;
  ```

Eşitli her zaman doğrudur, yanlış bir kullanım!

- == yerine = yazılmamalıdır.
  ```

İç İçe Eğer (if)

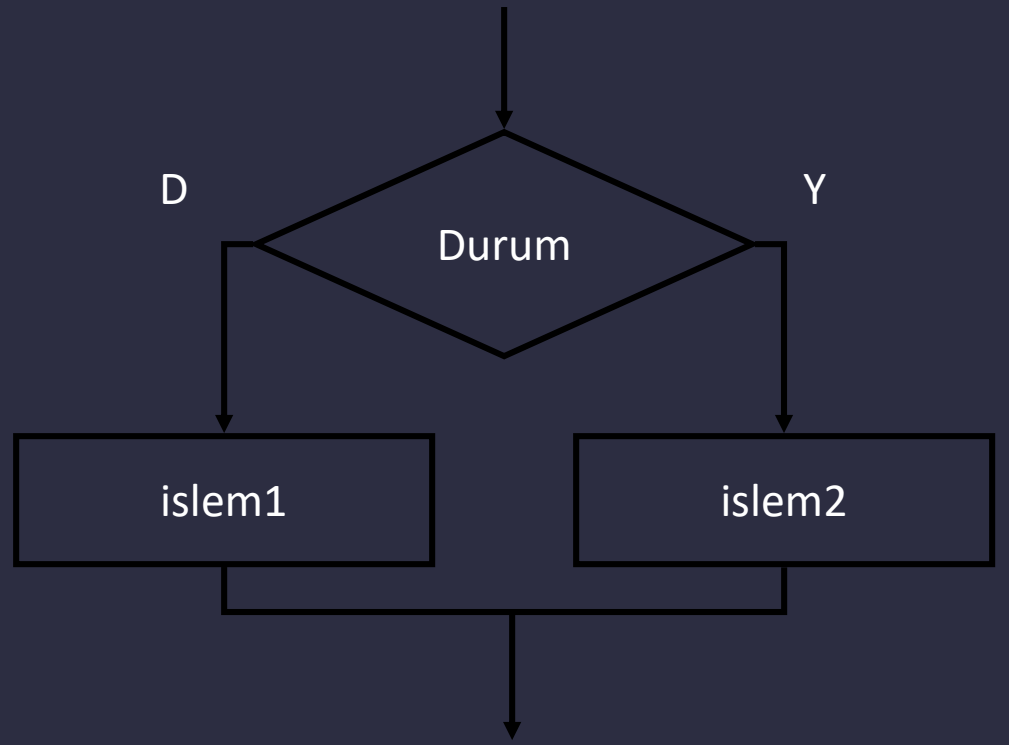
```
if (x == 3)
  if (y != 6) {
    z = z + 1;
    w = w + 2;
  }
```

Aynı İfadedir

```
if ((x == 3) && (y != 6)) {
  z = z + 1;
  w = w + 2;
}
```

Eğer-Değilse (If-else)

```
if (durum)
   islem1;
else
   islem2;
```



Eğer-Değilse Örneği

```
if (x)
{
   y++;
   z--;
}
else {
   y--;
   z++;
}
```

Else ile If Eşleştirme

- Else ifadesi daima en yakın if ile eşleşmektedir.

```
if (x != 10)
  if (y > 3)
    z = z / 2;
  else
    z = z * 2;
```

Aynı işlem:

```
if (x != 10) {
  if (y > 3)
    z = z / 2;
  else
    z = z * 2;
}
```

Farklı işlem:

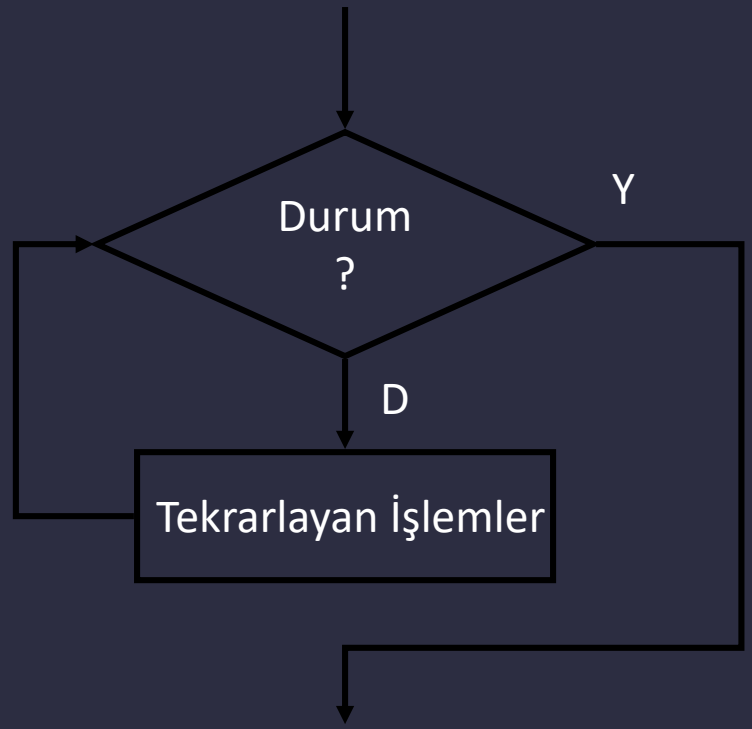
```
if (x != 10) {
  if (y > 3)
    z = z / 2;
}
else
    z = z * 2;
```

If-Else Zincirleri

```
if (ay == 4 || ay == 6 || ay == 9 || ay == 11)
   printf("Girilen ay 30 gündür.\n");
 else if (ay == 1 || ay == 3 ||
          ay == 5 || ay == 7 ||
          ay == 8 || ay == 10 ||
          ay == 12)
   printf("Girilen ay 31 gündür.\n");
 else if (ay == 2)
   printf("Girilen ay 28 veya 29 gündür.\n");
 else
   printf("Tanımsız ay.\n");
```

While

```
while (durum)
    tekrarlayan islemler;
```



Durum değeri doğru kaldıkça, tekrarlayan işlemler çalıştırılır.

Durum koşulu, tekrarlayan işlemlere başlanmadan önce kontrol edilmektedir.

While Örneği

```
x = 0;
while (x < 10) {
  printf("%d ", x);
  x = x + 1;
}
```

Sonsuz Döngüler

- Aşağıdaki döngü hiçbir zaman sona ermez:

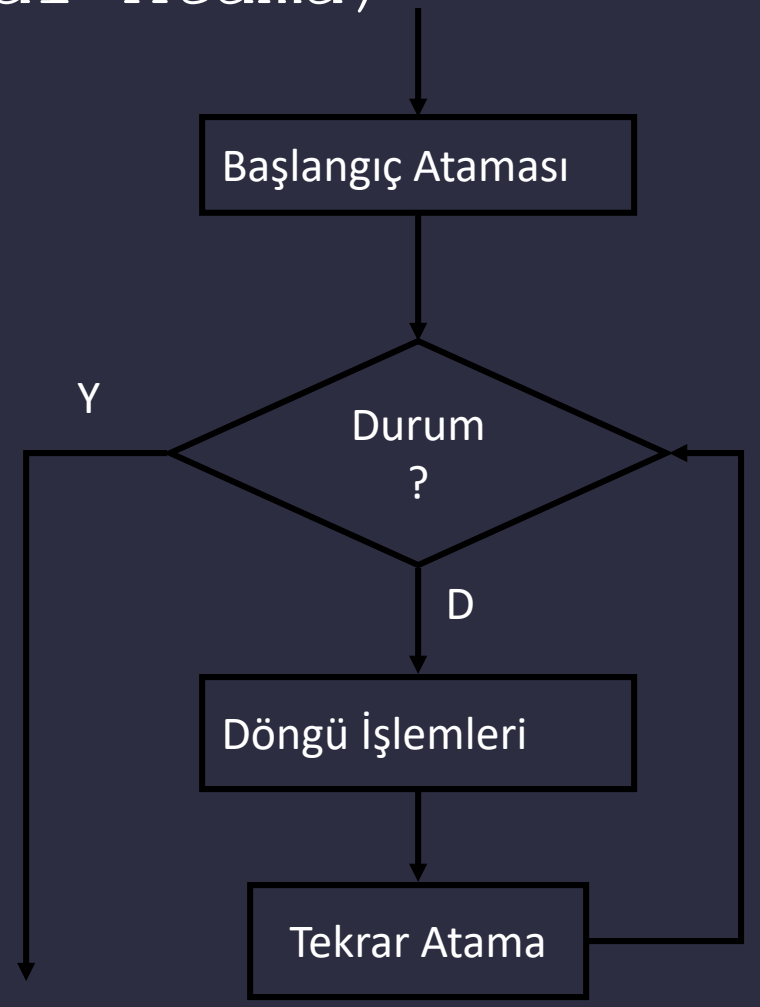
- ```
  x = 0;
  while (x < 10)
    printf("%d ", x);
  ```

- Kontrol edilen x<10 işlemini, değiştirecek bir atama olmadığı için, döngü sonsuza kadar dönecektir.
  ```

For

- `for (Başlangıç Ataması; Durum?; Tekrar Atama)`
  `döngü islemleri`

İlk olarak başlangıç ataması gerçekleştirilir.
Sonra Durum kontrol edilir
Eğer doğru ise döngü işlemi gerçekleştirilir.
Tekrar atama yapılır.
Durum kontrolü yanlış olana kadar süreç devam eder.



For Örneği

```
for (i = 0; i < 10; i++)
  printf("%d ", i);
```

For Örneği

```
for (i = 0; i <= 10; i ++)
  printf("%d ", i);
```

```
harf = 'a';
for (c = 0; c < 26; c++)
  printf("%c ", harf+c);
```

İç İçe Döngüler

```
for (mp1 = 0; mp1 < 10; mp1++) {
  for (mp2 = 0; mp2 < 10; mp2++) {
    printf("%d\t", mp1*mp2);
  }
  printf("\n");
}
```

İç İçe Döngüler

```
for (dis = 1; dis <= input; dis ++) {
   for (ic = 0; ic < dis; ic++) {
      toplam += ic;
   }
}
```

For ve While Karşılaştırması

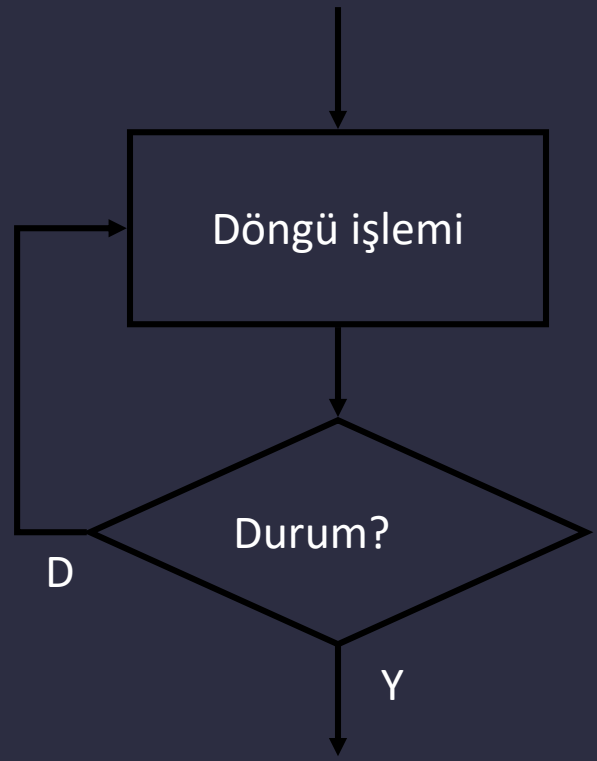
- For döngüsü, ilk çalıştırıldığında bir atama yaparak başlar. Atamadan sonra kontrol ifadesi gelir, ve for'un kapsamındaki kod parçacıkları çalıştırılır. Ardından ikinci atama bloğu çalıştırılır. Tekrar kontrol bloğu kontrol edilerek, yanlış olana kadar işlem devam ettirilir.

- While döngüsünde sadece tek bir koşul kontrol edilir. Koşul doğru olduğu sürece döngü devam eder.

- Her iki tür döngü çeşiti, birbirleri cinsinden yazılabilir.

Do-While

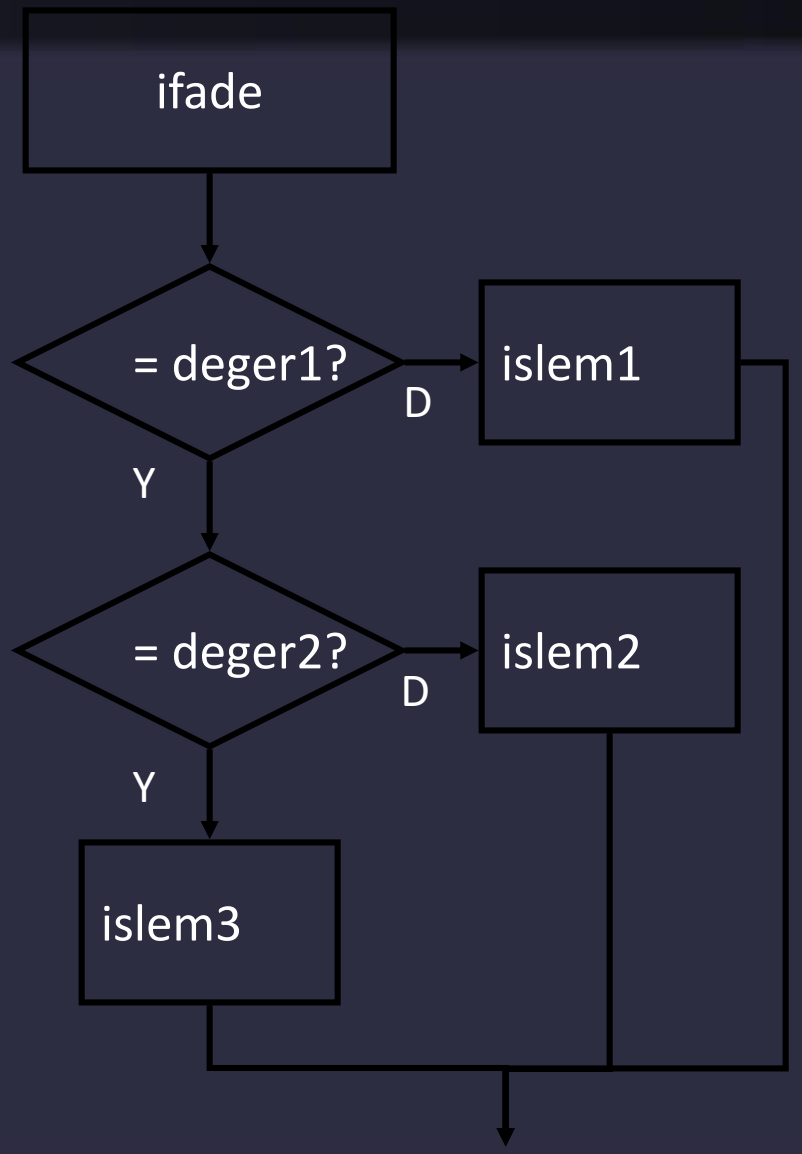
- `do`
  `döngü işlemi;`
- `while (durum);`



İlk başka herhangi bir koşula bakmadan döngü işlemini çalıştırır. Sonrasında durum'u kontrol eder.

Switch

```
switch (ifade) {
  case deger1:
    islem1; break;
  case deger2:
    islem2; break;
  default:
    islem3;
}
```



Switch Örneği

```
switch (ay) {
   case 4:
   case 6:
   case 9:
   case 11:
     printf("Ay 30 gündür.\n");
     break;
   case 1:
   case 3:
     printf("Ay 31 gündür.\n");
     break;
  case 2:
    printf("Ay 28 veya 29 gündür.\n");
    break;
  default:
    printf("Tanımsız ay.\n");
}
```

Switchler

- Case ifadeleri sabit sayı olmak zorundadır.

- Eğer satırlardan sonra break konulmazsa, diğer case'den işlem yapmaya devam edecektir.

```
switch (a) {
  case 1:
    printf("A");
  case 2:
    printf("B");
  default:
    printf("C");
}
```

Eğer a 1 ise, "ABC".
Eğer a is 2, "BC"
Değilse, "C"
Bastırılacaktır.

C ile Problem Çözme

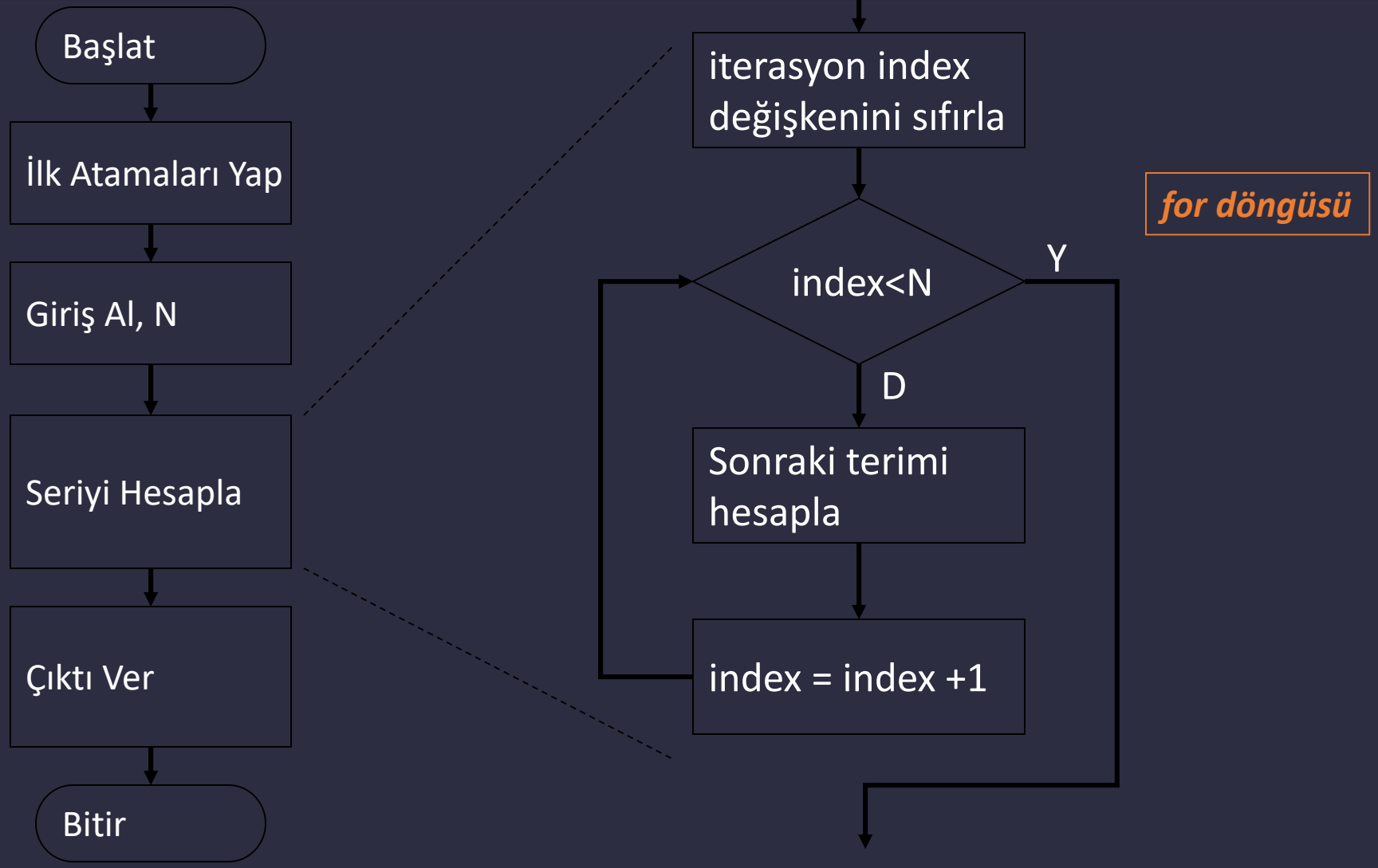
- Basit Yapılar
  - C İfadeleri, Operatörleri
  - Kontrol İfadeleri, if-else, switch
  - Döngü İfadeleri, while, for, do-while

Örnek 1: Pi Hesaplama

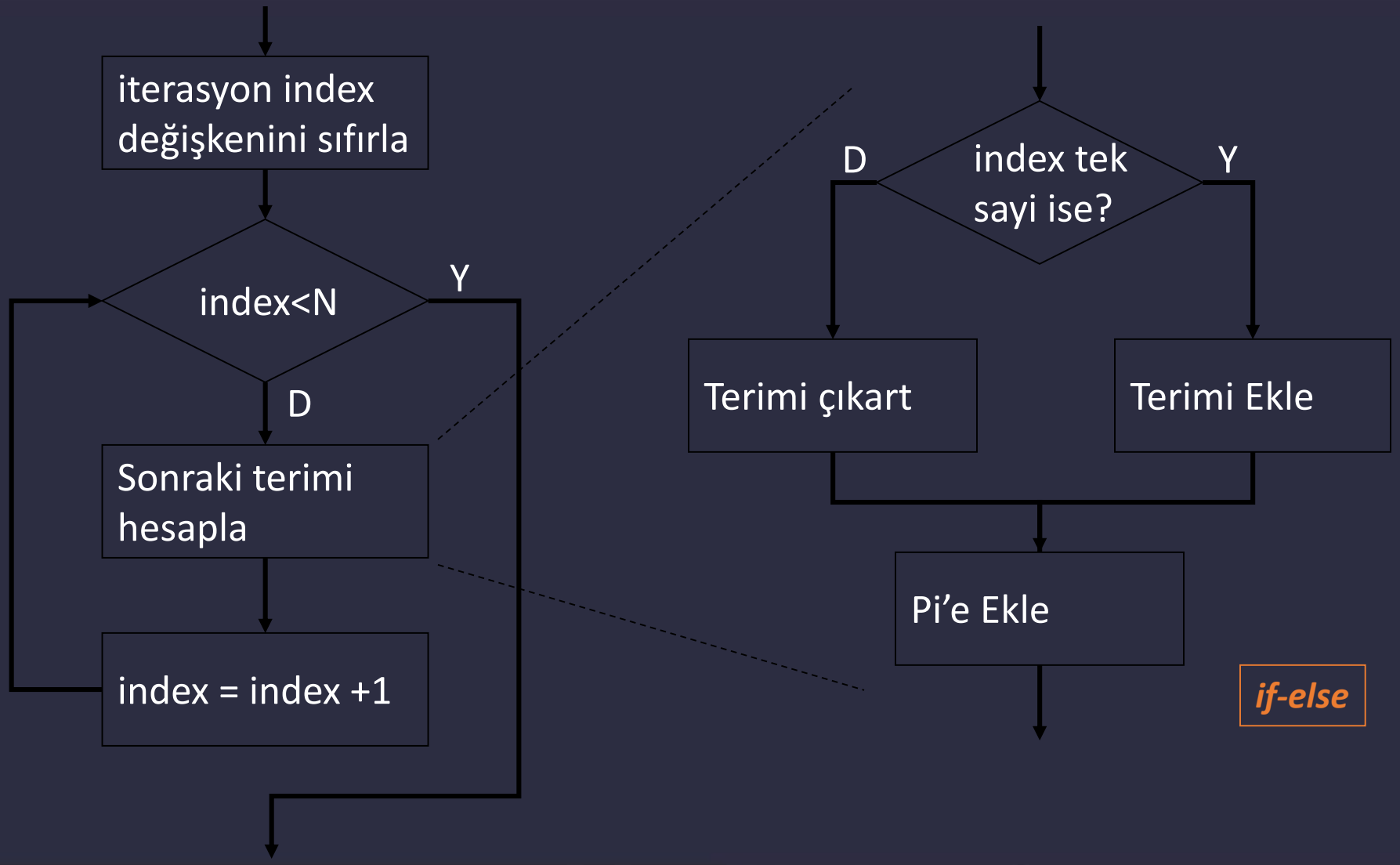
- π serisi açılımını kullanıcıdan alınacak N sayısı kadar bulunuz.

$$\pi = 4 - \frac{4}{3} + \frac{4}{5} - \frac{4}{7} + \cdots + (-1)^{n-1}\frac{4}{2n+1} + \cdots$$

Örnek 1: Pi Hesaplama



Örnek 1: Pi Hesaplama



Örnek 1: Pi Hesaplama

```
for (index=0; index < N; index ++) {
   if (index % 2) {

     pi -= 4.0 / (2 * index + 1);
   }
   else {
     pi += 4.0 / (2 * index + 1);
 }
```

Örnek 1: Pi Hesaplama

```
#include <stdio.h>


int main() {
  double pi = 0.0;
  int N, index;

  printf("N sayisini giriniz ");
  scanf_s("%d", &N);

  for (index =0; index < N; index ++) {
        if (index % 2) {
                pi -= 4.0 / (2 * index + 1);

        }
        else {
                pi += 4.0 / (2 * index + 1);

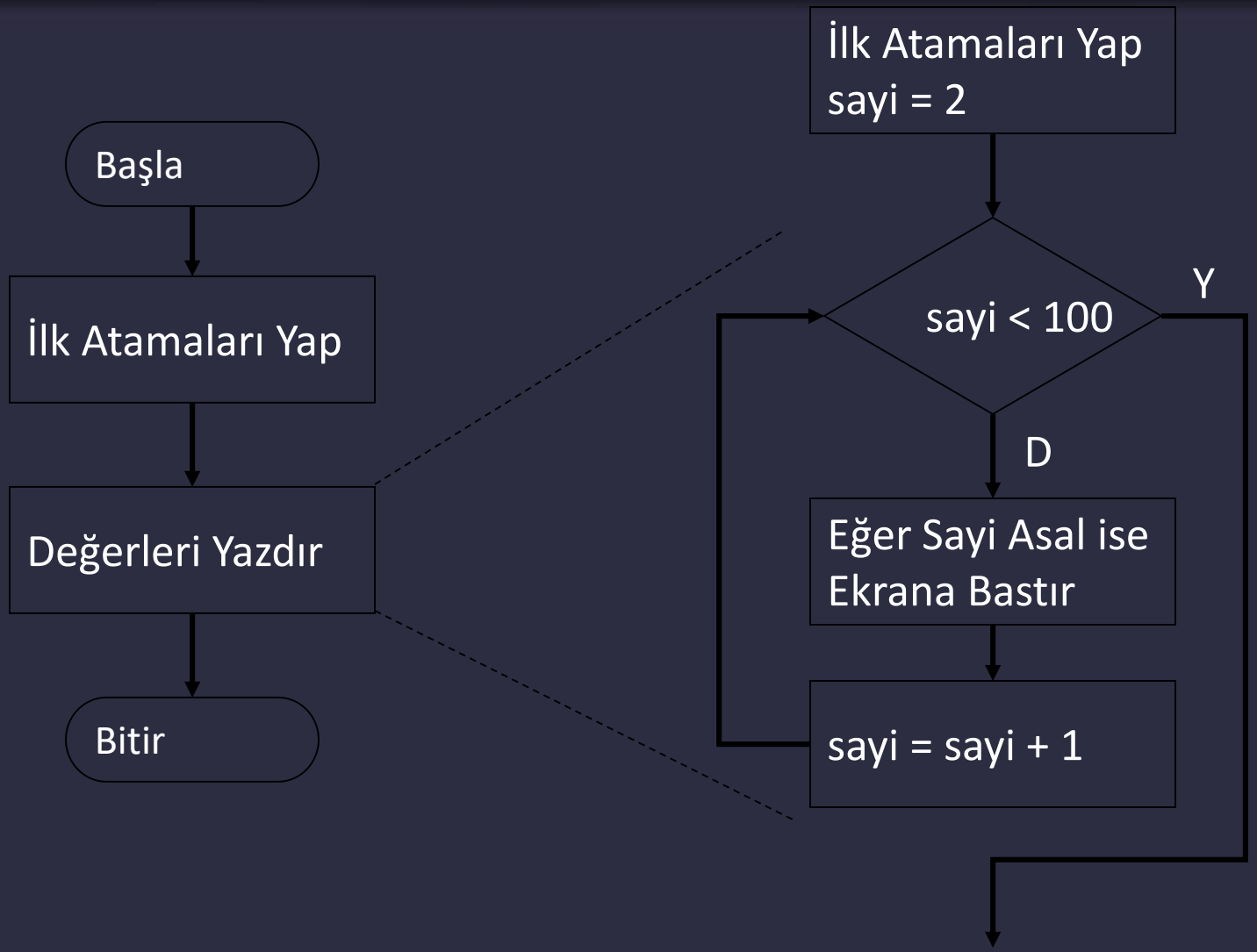
        }
  }
  printf("Sonuc Pi: %f\n", pi);

  return 1;
}
```

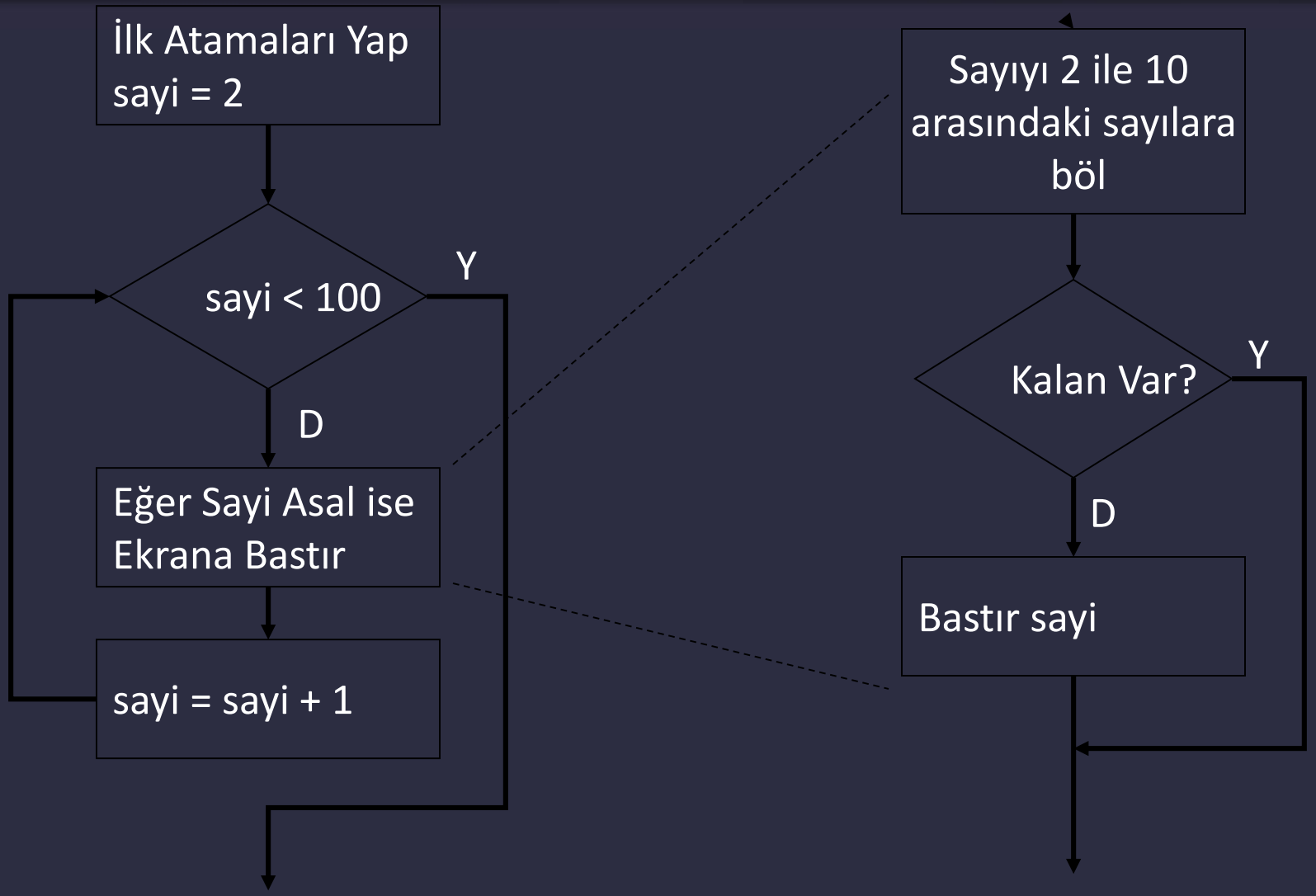
Örnek 2: Asal Sayı Hesaplama

- 100'den küçük tüm asal sayıları bulup, yazdırınız.
  - Bir asal sayı sadece kendisi ve 1'e tam bölünebilen sayılardır.
  - Asal olmayan 100'den küçük sayıların 2-10 sayıları arasında bölenleri vardır.

Örnek 2: Asal Sayı Hesaplama



Örnek 2: Asal Sayı Hesaplama



Örnek 2: Asal Sayı Hesaplama

Örnek 2: Asal Sayı Hesaplama

- Bir sayının 2-10 arası bölünüp bölünemediğinin sonucunu tutmak için bayrak isimli değişken kullandık.

- Macro'lar kod okunabilirliğinin arttırılması için faydalıdırlar.
- ```
  #define DOGRU  1
  #define YANLIS 0
  ```
  ```

Örnek 2: Asal Sayı Hesaplama

```
#include <stdio.h>
#define DOGRU  1
#define YANLIS 0

int main () {
  int sayi, bolen, asalMi;

  /* 2 - 100 arasındaki sayılar */
  for (sayi = 2; sayi < 100; sayi ++ ) {


     asalMi = DOGRU;
  /*  2-10 arasında bölünebilirlik testi */
     for (bolen = 2; bolen <= 10; bolen ++)
           if (((sayi % bolen) == 0) && (sayi != bolen))
                 asalMi = YANLIS;  /* asal Degil */

     if (asalMi)
     printf("Asal sayı: %d \n", sayi);
  }

  return 1;
}
```

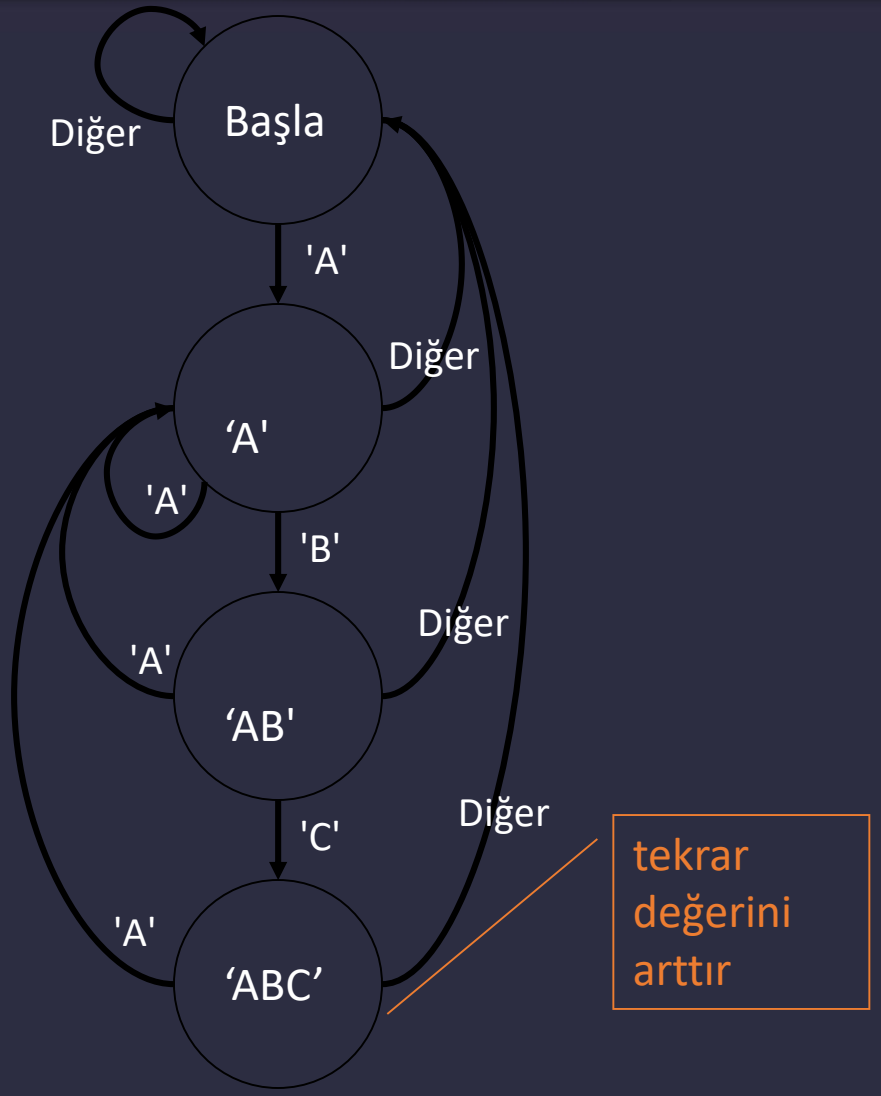
Örnek 3: Durum Makinası

- Kullanıcıdan karakterler alınız. Alınan karakterlerin içerisinde sırasıyla ABC girişinden kaç adet yapıldıysa ekrana gösteriniz.

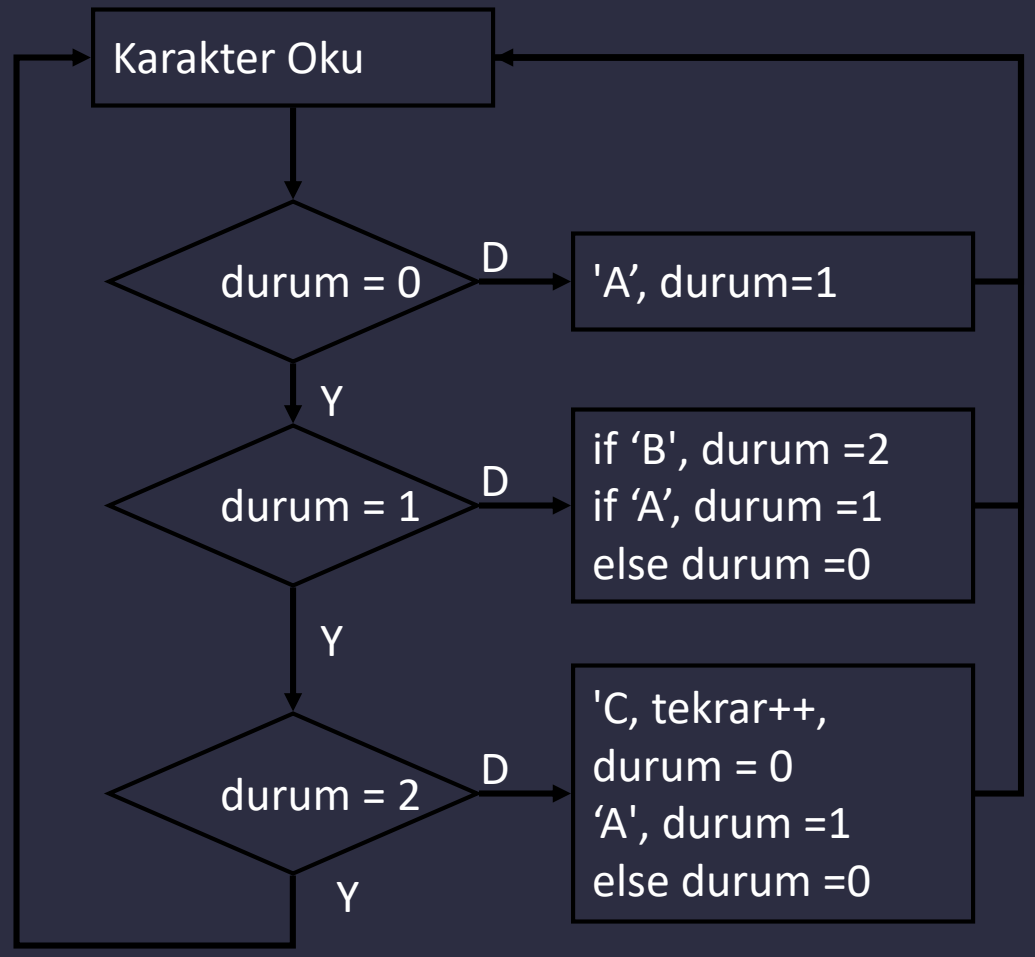
- scanf_s veya getchar() fonksiyonu içe yapılabilir.

- Durum makinaları kavramı ile gerçekleyebilirsiniz.

Örnek 3: Durum Makinası



tekrar değerini arttır



Substring: Code (Part 1)

```
#include <stdio.h>

İnt main() {
  char giriş;
  int durum = 0;
  int tekrar = 0;


  while ((giriş = getchar()) != '\n') {


    switch (durum) {


      case 0:
        if (giriş == ‘A’)
            durum = 1;
        break;
```

Substring: Code (Part 2)

```
case 1:
  if (giriş == 'B')
      durum = 2;
  else if (giriş == 'A')
      durum = 1;
  else
      durum = 0;
  break;
```

Substring: Code (Part 3)

```
        case 2:
          if (giriş == 'C') {
            tekrar++;
            durum = 0;
          }
          else if (giriş == 'A') {
              durum = 1;
          else
              durum = 0;
          break;
      }
    }
    printf("Tekrar sayisi = %d\n", tekrar);
    return 1;
  }
```

Break ve Continue

- `break;`
  - O an çalışan döngüyü kırarak, bitirir. Döngü koşulunun yanlış olması beklenmez.
- `continue;`
  - O aç çalışmakta olan döngüde, continue; satırı çalıştırıldığında, doğrudan döngünün başına gidilerek, döngü devam ettirilir.

Örnek 4: Break ve Continue

```
for (i = 0; i <= 20; i++) {
    if (i%2 == 0) continue;
    printf("%d ", i);
}
```

```
for (i = 0; i <= 20; i++) {
    if (i%2 == 0) break;
    printf("%d ", i);
}
```