Bölüm 3 – Yapısal Programlama

Konular

Amaçlar

- Bu bölümde öğrenilecekler:
  - Temel problem çözme tekniklerinin anlaşılması.
  - Tepeden aşağı, adım adım sadeleştirme tekniği ile algoritma geliştirebilme.
  - `if` ve `if…else` seçim deyimlerini kullanabilme.
  - `while` tekrar deyimini kullanabilme.
  - Sayaç kontrol ve nöbetçi kontrol deyimlerini anlayabilme.
  - Yapısal programlamayı anlayabilme.
  - Artırma, azaltma ve atama operatörlerini kulanabilme.

3.1 Giriş

- Bir program yazmadan önce:
  - Problem çok iyi anlaşılmalı
  - Problemin çözümü çok iyi planlanmalı
- Bir program yazarken:
  - Uygun gruplamalar bilinmeli
  - İyi programlama prensipleri kullanılmalı

3.2 Algoritmalar

- Problemlerin çözülmesi
  - Bütün problemler belli işlemlerin uygun sırada yapılması ile çözülebilir.

- Algoritma:
  - Problemi çözmek için çalıştırılacak işlemlerin, çalışma sırasıdır.

3.3 Pseudocode

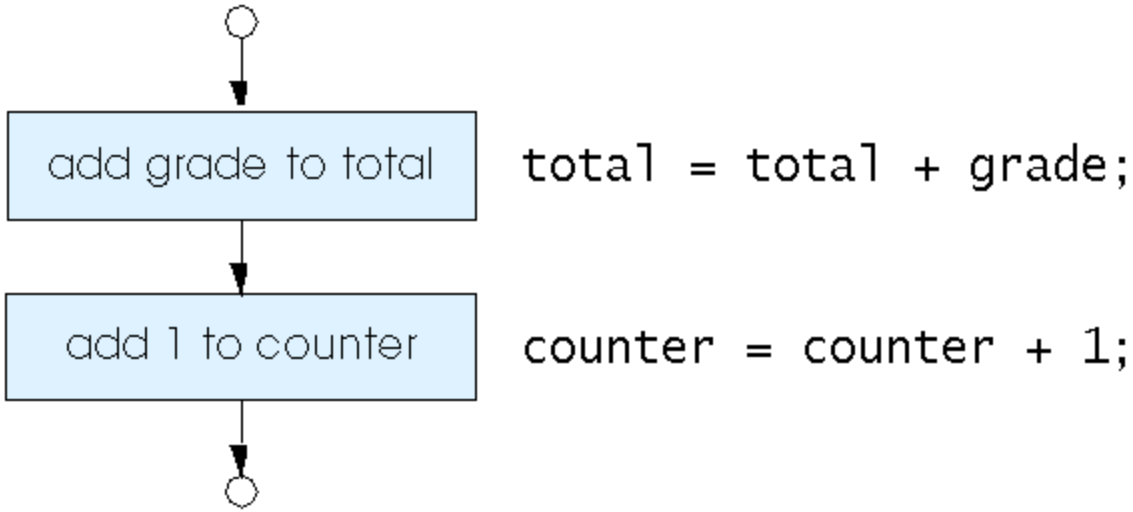
- Pseudocode (Sahte kodlar)
  - Algoritma geliştirmeye yardımcı olan yapay bir dildir.
  - Konuşma diline yakındır.
  - Bilgisayarda çalıştırılmaz.
  - Bir programı yazmadan önce, programın tümünü algılamamıza yardım eder.
    - C programlama diline çevirmek kolaydır.
    - Sadece programın ana komutlarını içerir.

3.4 Kontrol yapıları

- Sıralı çalışma
  - Komutlar programa yazıldıkları sırada teker teker çalışır.
- Kontrol transferi
  - Sıradaki komutun çalıştırılmayıp başka bir komutun çalıştırılması
- Bohm and Jacopini
  - Bütün programlar 3 kontrol yapısı ile yazılabilir.
    - Sıra yapısı: C komutları yazıldıkları sırada çalıştırılır.
    - Seçme yapısı: C de 3 tanedir: `if`, `if…else`, ve `switch`
    - Döngü yapısı: C de 3 tanedir: `while`, `do…while` ve `for`

3.4 Kontrol yapıları

Figure 3.1 C deki sıralı yapının akış şeması.



3.4 Kontrol yapıları

- Akış diyagramı
  - Algoritmanın grafiksel gösterimi
  - Özel şekiller birbirine çizgilerle bağlanır ve oklar akış yönünü gösterir.
  - Dikdörtgen şekli (işlem sembolu):
    - Herhangi bir işlemi gösterir.
  - Oval şekil:
    - Programın veya programın bir bölümünün başlangıcını ve sonunu gösterir.
  - Baklava sembolü (karar işareti)
    - Karar verme durumunda olduğunu gösterir.
    - Programın yapılandırılmasını kolaylaştırır.

3.5 `if` Seçim Deyimi

- Seçim yapısı:
  - İşlem gruplarından birini seçmek için kullanılır.
  - Pseudocode:

    Eğer(if) öğrencinin notu, 60dan büyük veya eşit ise
    Ekrana "Geçti" yazdır.

- 'If' deyimindeki koşul doğru(true) ise
  - Yazdır deyimi yürütülür ve program bir sonraki deyimden devam eder.
  - If koşulu yanlış(`false`) ise, yazdırma işlemi yaptırılmaz ve program bir sonraki deyimden devam eder.
  - Satır başındaki boşluklar programın anlaşılmasını kolaylaştırmak içindir.
    - C boşlukları ve satır sonlarını dikkate almaz.

3.5 `if` Seçim Yapısı

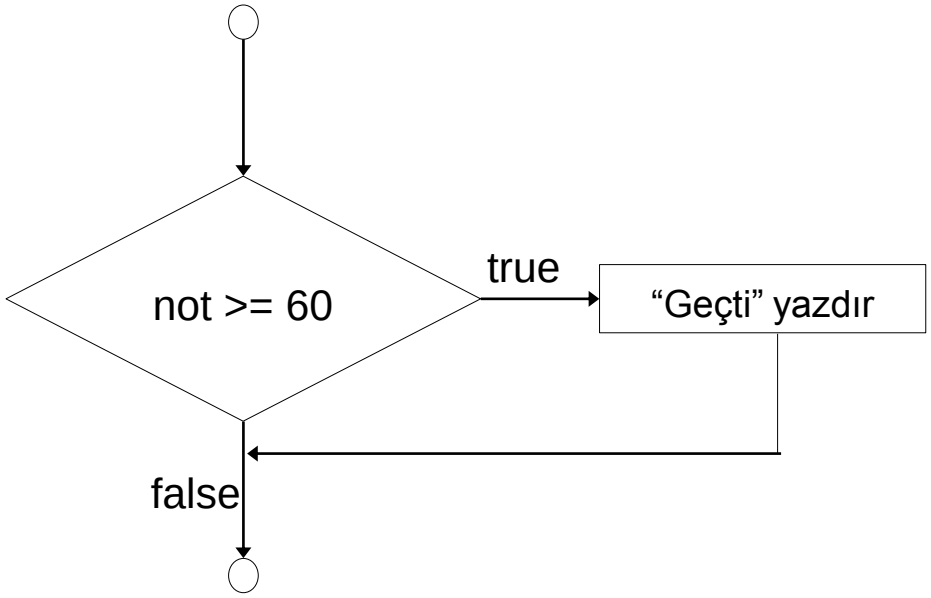
- Sahte kod (pseudocode):
  - C kodu:

```
if ( grade >= 60 )
   printf( "Passed\n" );
```

  - C programlama pseudocode'a çok yakındır.
- Baklava sembolü (karar işareti)
  - Karar verme durumunda olduğunu gösterir.
  - Bir koşul içerir `ve` bu koşul `doğru (true)` veya `yanlış (false)` olabilir.
  - Koşulu test eder, uygun yolu izler.

3.5 `if` Seçim Deyimi

- `if` deyimi tek giriş tek çıkışlı bir deyimdir.



Bir hesaplama sonucu da karar vermek için kullanılabilir. Eğer sonuç

sıfır ise – yanlış(`false`)

Sıfır olmayan sayılar için – doğru(`true`)

Örnek:

(3–4) - `true`

3.6 `if...else` Seçim Deyimi

- `if`
  - Eğer koşul doğru ise bir işlem yürütür.
- `if...else`
  - Koşul doğru ise bir işlem, değilse başka bir işlem yürütür.
- Psuedocode:

Eğer(if) öğrencinin notu, 60dan büyük veya eşit ise
Ekrana "Geçti" yazdır.
Aksi halde
Ekrana "Kaldı" yazdır.

Not: Satır girintilerine dikkat

3.6 `if...else` Seçim Deyimi

- C program parçası:

```
if ( grade >= 60 )
   printf( "Passed\n");
else
   printf( "Failed\n");
```

- Üçlü koşul operatörü (`?:`)
  - Üç argüman alır (koşul, eğer doğru ise bir değer, eğer yanlış ise başka değer)
  - Aşağıdaki ifade

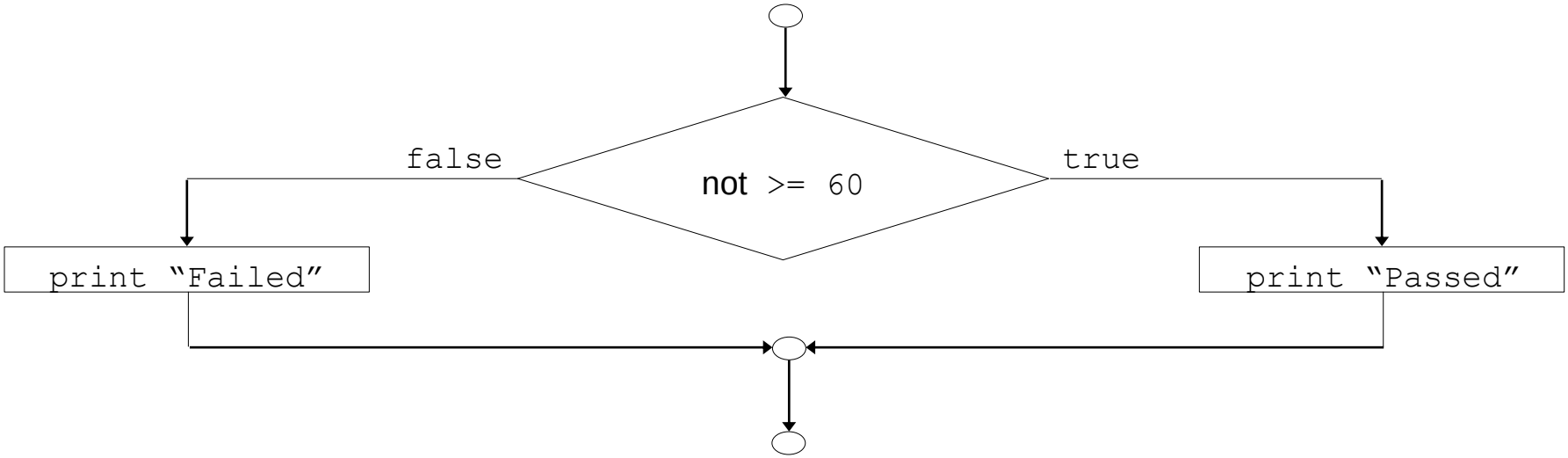
```
printf( "%s\n", grade >= 60 ? "Passed" :
  "Failed" );
```

  - şöyle de yazılabilir.:

```
grade >= 60 ? printf( “Passed\n” ) : printf(
  “Failed\n” );
```

3.6 `if...else` Seçim Deyimi

- `if...else` deyiminin akış diyagramı



- İçiçe `if...else` deyimi
  - Bir çok durumu `if...else` deyimi içine yazacağımız başka `if...else` deyimleri ile test edebiliriz.
  - İlk doğru olan `if...else` deyiminden sonrakiler atlanır.

3.6 `if...else` Seçim Deyimi

– İçiçe `if...else` deyiminin pseudocode'u

Eğer(If) öğrencinin notu 90'a eşit ya da 90'dan büyükse
"A" yazdır
aksi takdirde
Eğer(If) öğrencinin notu 80'a eşit ya da 80'dan büyükse
"B" yazdır
aksi takdirde
Eğer(If) öğrencinin notu 70'a eşit ya da 70'dan büyükse"C" yazdır
aksi takdirde
Eğer(If) öğrencinin notu 60'a eşit ya da 60'dan büyükse"D" yazdır
aksi takdirde
"F"yazdır

3.6 `if...else` Seçim Deyimi

- Birleşik deyim:
  - Birden fazla deyim iki süslü parantez içine alınır.
  - Örnek:

```
if ( grade >= 60 )
   printf( “Geçti.\n" );
else {
   printf( “Kaldı.\n" );
   printf( “Dersi yeniden almalısın.\n" );
  }
```

  - Parantezler olmasaydı:

```
printf( “Dersi yeniden almalısın.\n" );
```

    Her durumda çalıştırılacaktı.

3.6 `if...else` Seçim Deyimi

- Blok (Block):
  - Birleşik deyimler grubu
- Biçim hataları (Syntax errors)
  - Derleyici tarafından yakalanan hatalar
- Mantık hataları (Logic errors):
  - Çalışma sırasında etkisi görünen hatalardır.
  - Non-fatal: program çalışır fakat yanlış sonuç verir.
  - Fatal: program zamanından önce sona erer.

3.7 `while` tekrar deyimi

- Tekrar yapısı
  - While daki koşul doğru olduğu sürece bir grup işlemi tekrarlayan yapıdır.
  - Psuedocode:

alışveriş listemde birden fazla malzeme bulunduğu sürece(while)

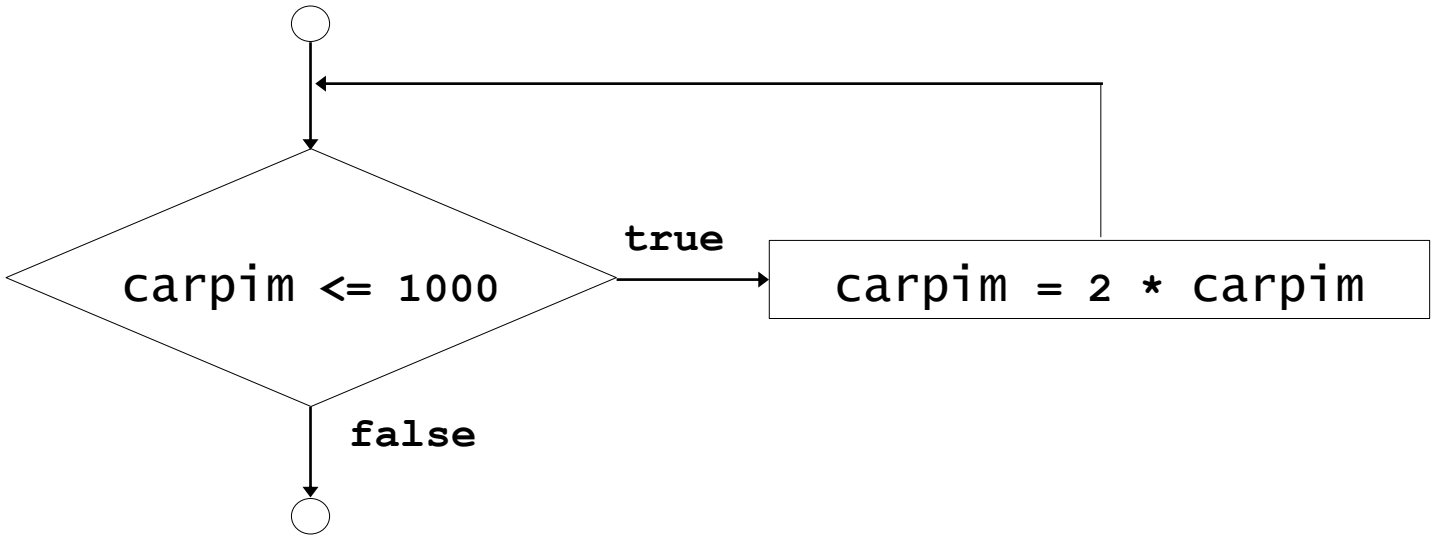
bir sonraki malzemeyi al ve alışveriş listemden bu malzemeyi çıkar

  - `while` daki koşul yanlış olana kadar işlemler tekrar eder.

3.7 `while` tekrar deyimi

- Örnek:

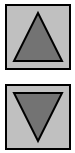
```
int carpim = 2;
while ( carpim <= 1000 )
       carpim = 2 * carpim;
```



3.8 Algoritma planlama (Sayaç kontrol deyimi)

- Sayaç kontrol deyimi
  - Sayaç belirli bir sayıya erişene kadar gruptaki işlemler tekrar eder.
  - Belirli tekrar: tekrar sayısı bellidir.
  - Örnek: 10 öğrenci quiz oldu. Notlar (0 -100 e kadar tamsayılar) size verilmiş. Sınıf ortalamasını nasıl bulursunuz ?
  - Pseudocode:

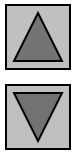
Toplamı sıfır yap
Sayacı 1 yap
Sayac 10'a eşit ya da 10'dan küçükken (while)
Diğer notu gir
Girilen notu toplama ekle
Sayacı bir artır
Sınıf ortalamasını, toplamı 10'a bölerek bul
Sınıf ortalamasını yazdır



Outline

fig03_06.c (Part 1 of 2)

```
1  /* Fig. 3.6: fig03_06.c
2     /* Sayac kontrollü döngü ile sınıf ortalamısının bulunması */
3  #include <stdio.h>
4
5
6  int main()
7  {
8     int sayac;
9     int not;
10    int toplam;
11    int ortalama;
12
13    /* ilk değerlerin verilmesi */
14    toplam = 0;
15    sayac = 1;
16
17    /* işlem bölümü */
18    while ( sayac <= 10 ) {
19       printf( "Notu girin: " );
20       scanf( "%d", &not );
21       toplam = toplam + not;
22       sayac = sayac + 1;
23    }
24
```



Outline

```
25    /* Bitiş bölümü */
26    ortalama = toplam / 10;
27
28
29    printf( "Sınıf ortalaması %d dir.\n", ortalama );
30
31    return 0;
32
33 } /* program başarılı bir şekilde bitti */
```

fig03_06.c (Part 2 of 2)

```
Notu girin: 98
Notu girin : 76
Notu girin : 71
Notu girin : 87
Notu girin : 83
Notu girin : 90
Notu girin : 57
Notu girin : 79
Notu girin : 82
Notu girin : 94
Sınıf ortalaması 81 dir
```

Program Output

3.9 Tepeden aşağı, Adımsal iyileştirme ile Algoritma planlama

- Problem :

  Öğrenci sayısı baştan belli olmayan bir grup öğrencinin not ortalamısını bulan program yazın.

  – Program bu durumda nasıl sonlanacak?

- Nöbetçi değer kullanma (sentinel value)
  – 'Sentinel value' yerine, sinyal değer(signal value), aptal değer (dummy value), veya bayrak değer (flag value) olarak da adlandırılır.
  – Bilgi girişinin bittiğini gösterir.
  – Kullanıcı nöbetçi değeri girince, döngüden çıkar.
  – Nöbetçi değer bilgi olarak anlam ifade etmeyen bir değer olmak zorundadır. (bu durumda `-1` olabilir)

3.9 Tepeden aşağı, Adımsal iyileştirme ile Algoritma planlama

- Tepeden aşağı, adımsal iyileştirme
  - En tepedeki problem cümlesi ile başlar:

    Quizin sınıf ortalamasını belirle.
  - Bu problem küçük parçalara bölünüp sıraya konur:

    Değişkenlere başlangıç değeri ver.

    Notları gir,topla ve say.

    Sınıf ortalamasını hesapla ve ekrana yazdır.
- Bir çok programın üç aşaması vardır:
  - Başlangıç: Program değişkenlerine ilk değer verilir.
  - İşlem: data değerlerini alır ve sayaçları değiştirir.
  - Sonlandırma: sonuçları hesaplar ve ekrana yazar.

3.9 Tepeden aşağı, Adımsal iyileştirme ile Algoritma planlama

- *Değişkenleri belirle* bölümünü açalım:

  ‘*toplam’ adlı değişkeni sıfır yap*
  ‘sayac’ adlı değişkeni sıfır yap

- *Notları gir, topla ve say* bölümünü açalım

  Birinci notu al
  Kullanıcı nöbetçiyi girmediği sürece (while)
  Bu notu o andaki toplam değere ekle
  ‘sayac’ı bir arttır
  Sıradaki notu al (bu değer nöbetçi değer olabilir)

3.9 Tepeden aşağı, Adımsal iyileştirme ile Algoritma planlama

- *Sınıf ortalamasını hesapla ve yazdır*

  Eğer(if) 'sayac' sıfıra eşit değilse
  'toplam'ı 'sayac'a bölerek ortalamayı hesapla
  Ortalamayı ekrana yazdır
  aksi takdirde(else)
  'Not girilmemiştir' yazdır.

3.9 Tepeden aşağı, Adımsal iyileştirme ile Algoritma planlama

‘toplam’ adlı değişkeni sıfır yap

‘sayac’ adlı değişkeni sıfır yap

Birinci notu al

(while) Kullanıcı nöbetçiyi girmediği sürece

Bu notu o andaki toplam değere ekle

sayac’ı bir arttır

Sıradaki notu al (bu değer nöbetçi değer olabilir)

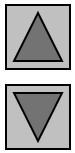
Eğer(if) ‘sayac’ sıfıra eşit değilse

‘toplam’ı ‘sayac’a bölerek ortalamayı hesapla

Ortalamayı ekrana yazdır

aksi takdirde(else)

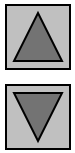
‘Not girilmemiştir’ yazdır.



Outline

fig03_08.c (Part 1 of 2)

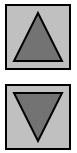
```
1  /* Fig. 3.8: fig03_08.c
2     /*sayac kontrollü döngülerle sınıf ortalaması bulan program */
3  #include <stdio.h>
4
5  int main()
6  {
7     float ortalama;
8     int sayac,not,toplam;
9
10    /* ilk değer atama */
11    toplam = 0;
12    sayac = 0;
13
14    /*işlem */
15    printf( "Notu giriniz, Çıkış için -1 : " );
16    scanf( "%d", &not );
17
18    while ( not != -1 ) {
19       toplam = toplam + not;
20       sayac = sayac + 1;
21       printf( "Notu giriniz, Çıkış için -1: " );
22       scanf("%d", &not);
23       }
```



Outline

```
24
25
26    /* sonlandırma */
27
28    if ( sayac != 0 ) {
29       ortalama = ( float ) toplam / sayac;
30       printf( "Sınıf ortalaması %.2f\n", ortalama );
31    }
32    else {
33       printf( "Hiç not girilmemiştir\n" );
34    }
35    return 0;
36 }
```

fig03_08.c (Part 2 of 2)



Outline

Program Output

```
Notu giriniz, Çıkış için -1 : 75
Notu giriniz, Çıkış için -1 : 94
Notu giriniz, Çıkış için -1 : 97
Notu giriniz, Çıkış için -1 : 88
Notu giriniz, Çıkış için -1 : 70
Notu giriniz, Çıkış için -1 : 64
Notu giriniz, Çıkış için -1 : 83
Notu giriniz, Çıkış için -1 : 89
Notu giriniz, Çıkış için -1 : -1
Sınıf ortalaması 82.50
```

İlk not olarak -1 girildiği durumda ekran görüntüsü

Notu giriniz, Çıkış için -1: -1
Hiç not girilmemiştir

3.10 İçiçe geçmiş kontrol yapıları

- Problem
  - Elimizde 10 kişilik bir sınıfın sene sonu notları var(1 = geçti, 2 = kaldı)
  - Sonuçları analiz eden bir program yazınız
    - Eğer 8 den fazla öğrenci geçtiyse , ekrana “Yüksek başarı“ yaz
- Dikkat
  - Program 10 kişini notunu analiz edecek
    - Sayaç- kontrollu döngü kullanılacak
  - İki sayaç kullanılacak
    - Biri geçenleri saymak için, diğeri kalanları saymak için
  - Her not 1 veya 2 değerini alacak
    - Eğer not 1 değilse 2 dir.

3.10 İçiçe geçmiş kontrol yapıları

- Problem

  Sene sonu notlarını analiz et ve yüksek başarı durumunun gerçekleşip gerçekleşmediğine karar ver

- İlk düzenleme

  Değişkenlere başlangıç değeri ver

  10 notu gir, geçenleri ve kalanları say

  Son durumu ekrana yaz ve yüksek başarı sağlandı mı karar ver

- İkinci düzenleme

- *Değişkenlere başlangıç değeri ver*

  Geçenler değişkenini sıfır yap

  Kalanlar değişkenini sıfır yap

  Öğrenci değişkenini bir yap

3.10 İçiçe geçmiş kontrol yapıları

- *10 notu gir, geçenleri ve kalanları say*

  Öğrenci sayısı 10'a eşit ya da 10'dan küçükken yeni sınav sonucunu al
  Eğer(If) öğrenci geçmişse
  Geçenlere bir ekle
  Aksi takdirde(else)
  Kalanlara bir ekle
  Öğrenci sayısına bir ekle

- *Son durumu ekrana yaz ve yüksek başarı sağlandı mı karar ver*

  Geçenlerin sayısını yazdır
  Kalanların sayısını yazdır
  Eğer 8'den fazla öğrenci geçmişse
  "Yüksek Başarı" yazdır

3.10 İçiçe geçmiş kontrol yapıları

Geçenler değişkenini sıfır yap

Kalanlar değişkenini sıfır yap

Öğrenci değişkenini bir yap

Öğrenci sayısı 10'a eşit ya da 10'dan küçükken yeni sınav sonucunu al

Eğer(If) öğrenci geçmişse

Geçenlere bir ekle

Aksi takdirde(else)

Kalanlara bir ekle

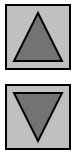
Öğrenci sayısına bir ekle

Geçenlerin sayısını yazdır

Kalanların sayısını yazdır

Eğer 8'den fazla öğrenci geçmişse

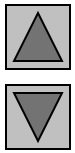
"Yüksek Başarı" yazdır



Outline

fig03_10.c (Part 1 of 2)

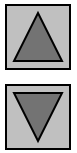
```
1  /* Fig. 3.10: fig03_10.c
2     Sınav sonuçlarının analizi */
3  #include <stdio.h>
4
5
6  int main()
7  {
8     /*Değişkenlere ilk değer verilmesi*/
9     int gecenler = 0;
10    int kalanlar = 0;
11    int ogrenci = 1;
12    int sonuc;
13
14    /*sayac kontrollü döngü ile 10 sonucun incelenmesi */
15    while ( ogrenci <= 10 ) {
16       printf( "Sonucu girin ( 1=geçti,2=kaldı ): " );
17       scanf( "%d", &sonuc );
18      if ( sonuc== 1 )
19          gecenler = gecenler + 1;
21       else
22          kalanlar = kalanlar + 1;
23       ogrenci = ogrenci + 1;
24    }
```



Outline

fig03_10.c (Part 2 of 2)

```
25
26
27
28
29
30
31
32
33    printf( "Geçenler %d\n", gecenler );
34    printf( "Kalanlar %d\n", kalanlar );
35
36
37    if ( gecenler > 8 ) {
38       printf( "Yüksek başarı\n" );
39    }
40
41    return 0;
42
43 }
```



Outline

Program Output

```
Sonucu Girin (1=geçti,2=kaldı): 1
Sonucu Girin (1=geçti,2=kaldı): 2
Sonucu Girin (1=geçti,2=kaldı): 2
Sonucu Girin (1=geçti,2=kaldı): 1
Sonucu Girin (1=geçti,2=kaldı): 1
Sonucu Girin (1=geçti,2=kaldı): 1
Sonucu Girin (1=geçti,2=kaldı): 2
Sonucu Girin (1=geçti,2=kaldı): 1
Sonucu Girin (1=geçti,2=kaldı): 1
Sonucu Girin (1=geçti,2=kaldı): 2
Geçenler 6
Kalanlar4
```

```
Sonucu Girin (1=geçti,2=kaldı): 1
Sonucu Girin (1=geçti,2=kaldı): 1
Sonucu Girin (1=geçti,2=kaldı): 1
Sonucu Girin (1=geçti,2=kaldı): 2
Sonucu Girin (1=geçti,2=kaldı): 1
Sonucu Girin (1=geçti,2=kaldı): 1
Sonucu Girin (1=geçti,2=kaldı): 1
Sonucu Girin (1=geçti,2=kaldı): 1
Sonucu Girin (1=geçti,2=kaldı): 1
Sonucu Girin (1=geçti,2=kaldı): 1
Geçenler 9
Kalanlar   1
Yüksek başarı
```

3.11 Atama Operatörleri

- Atama operatörleri atama deyimlerinin kısalmasını sağlar

  `c = c + 3;` toplama işlemi atama operatörünü kullanarak

  `c += 3` olarak yazılabilir.

  Biçim

  değişken = *değişken operatör deyim*;

  değişken `operatör`= *deyim*; olarak yazılabilir.

- Atama operatörü örnekleri:

```
d -= 4      (d = d - 4)
e *= 5      (e = e * 5)
f /= 3      (f = f / 3)
g %= 9      (g = g % 9)
```

3.11 Atama Operatörleri

Varsayım: `int c = 3, d = 5, e = 4, f = 6, g = 12;`

Atama operatörü	Örnek deyim	Açıklama	Atama
+=	`c += 7`	`c = c + 7`	**10**'u **c** ye
-=	`d -= 4`	`d = d - 4`	**1**'i **d** ye
*=	`e *= 5`	`e = e * 5`	**20**'yi **e** ye
/=	`f /= 3`	`f = f / 3`	**2**'yi **f**
%=	`g %= 9`	`g = g % 9`	**3**'ü **g** ye

Aritmetik atama operatörleri.

3.12 Artırma ve azaltma operatörleri

- Artırma operatörü (++)
  - `c+=1` 'in yerine kullanılabilir
- Decrement operator (`--`)
  - `c-=1` 'in yerine kullanılabilir
- Ön artırma
  - Operatör değişkenden önce kullanılır (`++c` or `--c`)
  - Değişkenin değeri deyim hesaplanmadan önce değişir.
- Son artırma
  - Operatör değişkenden sonra kullanılır(`c++` or `c--`)
  - Değişkenin değeri deyim hesaplandıktan sonra değişir.

3.12 Artırma ve azaltma operatörleri

- Eğer `c` 'in değeri `5` ise,

```
printf( "%d", ++c ); ekrana 6  yazar
printf( "%d", c++ ); ekrana 5  yazar
```

  - Her iki durumda da, `c` nin son değeri `6` dır.

- Eğer değişken bir deyimin içinde değilse
  - Ön artırma ve son artırma aynı sonucu verir.

```
++c;
printf( "%d", c );
```

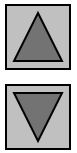
Yukarıdaki ile aşağıdaki program parçası aynı etkiyi yapar.

```
c++;
printf( "%d", c );
```

3.12 Artırma ve azaltma operatörleri

Operatör	Örnek deyim	Açıklama
++	++a	a' yı bir artır ve deyimi hesaplarken a'nın yeni değerini kullan.
++	a++	a'nın değerini deyimi hesaplarken kullan ve sonra a'nın değerini 1 artır.
--	--b	b' yı bir azalt ve deyimi hesaplarken b'nin yeni değerini kullan.
--	b--	b'nin değerini deyimi hesaplarken kullan ve sonra b'nin değerini 1 azalt.

Artırma ve Azaltma operatörleri

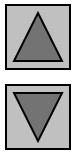


Outline

fig03_13.c

```c
/* Fig. 3.13: fig03_13.c
   Ön artırma ve son artırma*/
#include <stdio.h>


int main()
{
   int c;


   c = 5;
   printf( "%d\n", c );
   printf( "%d\n", c++ ); /* önartırma */
   printf( "%d\n\n", c );


   c = 5;
   printf( "%d\n", c );
   printf( "%d\n", ++c ); /* son artırma*/
   printf( "%d\n", c );

   return 0;

}
```

```
5
5
6

5
6
6
```



Outline

Program Çıktısı

3.12 Artırma ve azaltma operatörleri

Operatörler					Associativity	Tipi
++	--	+	-		Sağdan sola	tekil
*	/	%			Soldan sağa	çarpımsal
+	-				Soldan sağa	toplamsal
<	<=	>	>=		Soldan sağa	ilişkisel
==	!=				Soldan sağa	eşitlik
?:					Sağdan sola	koşullu
=	+=	-=	*=	/=	Sağdan sola	atama
Operatörlerin öncelikleri						