Kontrol Yapıları

Giriş

- Program yazmaya başlamadan önce
  - Problemi anlayın
  - Çözüm için bir yaklaşım düşünün
  - Yapılandırılacak blokları anlayın

Algoritmalar

- Prosedür
  - Bir programın gerçekleştireceği işlemlerdir
  - İşlemlerin gerçekleştirileceği sıradır
  - Algoritma olarakta adlandırılır
- Program kontrol
  - Bilgisayarın doğru bir şekilde işlemleri gerçekleştirebilmesi için görevlerin sıralanmasıdır

Pseudocode (Kaba kod)

- Pseudocode
  - Yapay ve günlük dil kullanır
  - Programcıya algoritmayı planlamasında yardımcı olur
  - Gerçek bir programlama dili değildir
  - C# koduna basit bir şekilde çevrilir

Kontrol Yapıları

- Program kontrolü
  - Program bir deyimden (statement) bir sonraki deyime geçerek çalışır
    - Sıralı çalışma (Sequential execution)
  - Bir sonraki satırdan başka bir satırı çalıştırmak için kontrol yapıları kullanılır
    - Seçme Yapısı (Selection structure)
      - **`if`** ve **`if/else`** deyimleri
      - **`goto`** deyimi
        - Kesinlikle gerekmedikçe kullanılmamalıdır
        - Programın okunmasında problemler oluşturabilir
    - Tekrar Yapısı (Repetition structure)
      - **`while`** ve **`do/while`** döngüleri (loops)
      - **`for`** ve **`foreach`** dögüleri

Kontrol Yapıları

- Akış şemaları (Flow carts)
  - Programın haritalanması için kullanılır
  - Sırayla gelecek olayları gösterir
    - Dikdörtgen bir işlemi gösterir (Toplama, Yazdırma v.b.)
    - Oval başlama ve bitiş gösterimlerinde kullanılır
    - Daireler bağlantılar için kullanılır
    - Paralel kenar kararişlemlerini gösterir
  - Kontrol yapılarının birleştirilmesi
    - Yığın
      - Birinden diğerine geçer
    - İçiçe
      - Birisini diğerinin içine ekler

Kontrol Yapıları

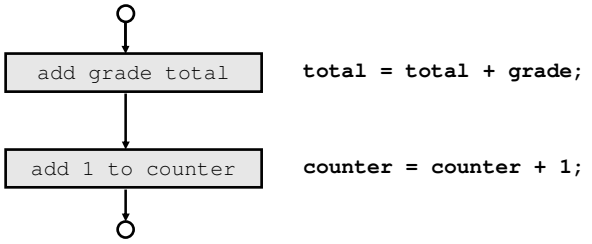


Fig. 4.1 Flowcharting C#'s sequence structure.

Kontrol Yapıları

C# Keywords				
abstract	as	base	bool	break
byte	case	catch	char	checked
class	const	continue	decimal	default
delegate	do	double	else	enum
event	explicit	extern	false	finally
fixed	float	for	foreach	get
goto	if	implicit	in	int
interface	internal	is	lock	long
namespace	new	null	object	operator
out	override	params	private	protected
public	readonly	ref	return	sbyte
sealed	set	short	sizeof	stackalloc
static	string	struct	switch	this
throw	true	try	typeof	uint
ulong	unchecked	unsafe	ushort	using
value	virtual	void	volatile	while

Fig. 4.2 C# keywords.

`if` Seçme Yapısı

- **`if`** yapısı
  - Programın seçme yapmasını sağlar
  - Belirtilen şarta göre işlem yapar
    - **`bool`** türünde bir ifadeyi değerlendirir
    - True: işlemi yapar
    - False: işlemi atlar
  - Noktalı virgül gerektirmez

`if` Seçme Yapısı

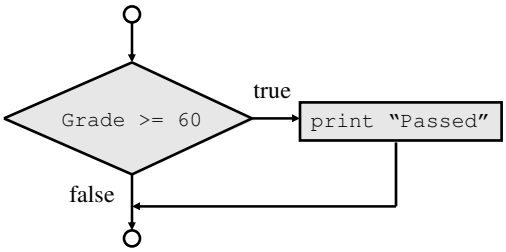


Fig. 4.3 Flowcharting a single-selection **`if`** structure.

`if/else` seçme yapısı

- **`if/else`** yapısı
  - Birden fazla durumun test edilmesinde kullanılır
  - ({) parantezi kullanılarak birden fazla satır işlem yapılabilir

`if/else` seçme yapısı

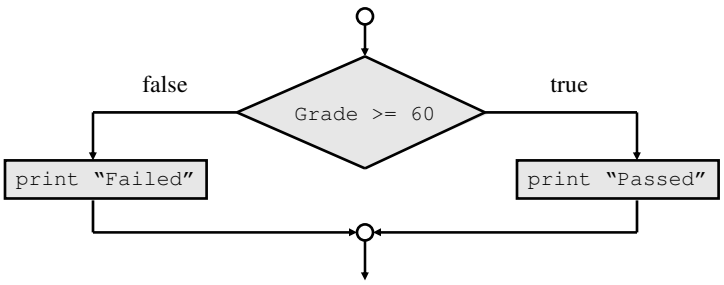


Fig. 4.4 Flowcharting a double-selection **`if/else`** structure.

Şart operatörü (?:)

- Şart operatörü (?:)
  - C#'ın tek üç parçalı operatörüdür
  - **`if/else`** yapısına benzer
  - Yazımı şöyledir:
    - (*boolean value* ? *if true* : *if false*)

`while` tekrar yapısı

- Tekrar yapısı
  - Tekrarlı yapılan işlemleri ifade eder
    - While deyimi doğru (true) olduğu sürece işlem tekrarlanır
    - While deyimi yanlış (false) olduğunda işlem biter
  - Bir satır veya paranteze alınmış bir bloğu içerir
    - Şartın mutlaka sonlandırılmış olması gerekir
      - Sonlandırılmamış şartlarda sonsuz dögüye girer

`while` tekrar yapısı

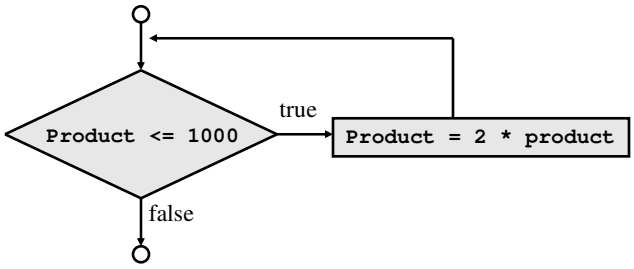


Fig. 4.5 Flowcharting the **`while`** repetition structure.

Algoritmanın düzenlenmesi: Örnek Çalışma 1 (Sayaç Kontrollü Tekrar)

- Sayaç (Counter) Kontrollü Tekrar
  - Bir zamanda bir bilgi girişinde kullanılır
  - Sayaç döngünün ne zaman biteceğini belirlemek için kullanılır
  - Sayaç bir belirlenen değere ulaştığında işlemler bitirilir

Algoritmanın düzenlenmesi: Örnek Çalışma 1 (Sayaç Kontrollü Tekrar)

total değişkenini sıfır yap
grade sayacını bir yap

grade sayacı 10'a eşit veya küçük ise
Bir sonraki grade değerini gir
grade değerini total değerine ekle
grade sayacını bir artır

total değerini 10'a bölerek average değerini bul
average değerini ekrana yaz

Fig. 4.6 Sınıf ortalaması probleminin sayaç kontrollü çözümü

```
1    // Fig. 4.7: Average1.cs
2    // Class average with counter-controlled repetition.
3
4    using System;
5
6    class Average1
7    {
8       static void Main( string[] args )
9       {
10         int total,          // sum of grades
11             gradeCounter,   // numbe
12             gradeValue,     // grade
13             average;        // average of all grades
14
15         // initialization phase
16         total = 0;          // clear total
17         gradeCounter = 1;   // prepare to l
18
19         // processing phase
20         while ( gradeCounter <= 10 ) // loop 10 times
21         {
22            // prompt for input and read grade from user
23            Console.Write( "Enter integer grade: " );
24
25            // read input and convert to integer
26            gradeValue = Int32.Parse( Console.ReadLine() );
27
28            // add gradeValue to total
29            total = total + gradeValue;
30
31            // add 1 to gradeCounter
32            gradeCounter = gradeCounter + 1;
33         }
```

total değişkenini 0 olarak başlat (initiliaze)

gradeCounter değerini 1 olarak başlat

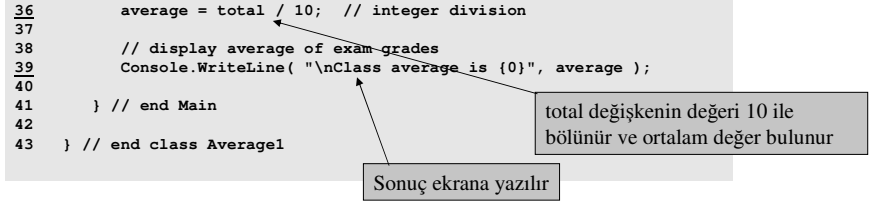
while döngüsü 10 defa çalışır ve 10 öğrencinin notunu alır

grade değeri girişi için kullanıcı beklenir

10 adet grade değeri toplanır

Sayaca 1 eklenir ve 10 dan büyük olunca döngüden çıkılır

```
34
35          // termination phase
36          average = total / 10;  // integer division
37
38          // display average of exam grades
39          Console.WriteLine( "\nClass average is {0}", average );
40
41       } // end Main
42
43    } // end class Average1
```




```
Enter integer grade: 100
Enter integer grade: 88
Enter integer grade: 93
Enter integer grade: 55
Enter integer grade: 68
Enter integer grade: 77
Enter integer grade: 83
Enter integer grade: 95
Enter integer grade: 73
Enter integer grade: 62

Class average is 79
```

Algoritmanın düzenlenmesi: Örnek Çalışma 2 (Sentinel Kontrollü Tekrar)

- Sentinel kontrollü tekrar
  - İstenen sayıda tekrarlanır
  - Sentinel değer
    - döngüden çıkmayı sağlar

Algoritmanın düzenlenmesi: Örnek Çalışma 2 (Sentinel Kontrollü Tekrar)

total değeri sıfır olarak başlat
sayacı sıfır olarak başlat

ilk grade değerini gir (sentinel değer olabilir)

kullanıcı sentinel değer girmediyse
    grade değerini total değerine ekle
    grade sayacına bir ekle
    bir sonraki grade değerini gir (sentinel olabilir)

If sayaç değeri sıfır değilse
    total değerini sayaca bölerek average değerini bul
    average değerini ekrana yaz
Else
    average değeri girilmediğini ekrana yaz

Fig. 4.8 Sınıf ortalaması probleminin sentinel kontrollü döngüyle çözümü

```
1    // Fig. 4.9: Average2.cs
2    // Class average with sentinel-controlled repetition.
3
4    using System;
5
6    class Average2
7    {
8       static void Main( string[] args )
9       {
10         int total,           // sum of grades
11             gradeCounter,    // number of grades ent
12             gradeValue;      // grade value
13
14         double average;      // average of all grades
15
16         // initialization phase
17         total = 0;           // clear total
18         gradeCounter = 0;    // prepare to loop
19
20         // processing phase
21         // prompt for input and convert to integer
22         Console.Write( "Enter Integer Grade, -1 to Quit: " );
23         gradeValue = Int32.Parse( Console.ReadLine() );
24
```

average değişkeni double türünde tanımlandı

gradeCounter ve total değişkenleri 0 olarak başlatıldı

gradeValue değişkeni için kullanıcıdan bir değer alındı

```
25          // loop until a -1 is entered by user
26          while ( gradeValue != -1 )
27          {
28             // add gradeValue to total
29             total = total + gradeValue;
30
31             // add 1 to gradeCounter
32             gradeCounter = gradeCounter + 1;
33
34             // prompt for input and read grade from u
35             // convert grade from string to integer
36             Console.Write( "Enter Integer Grade, -1 to Quit: " );
37             gradeValue = Int32.Parse( Console.ReadLine() );
38
39          } // end while
40
41          // termination phase
42          if ( gradeCounter != 0 )
43          {
44             average = ( double ) total / gradeCounter;
45
46             // display average of exam grades
47             Console.WriteLine( "\nClass average is {0}", average );
48          }
49          else
50          {
51             Console.WriteLine( "\nNo grades were entered" );
52          }
53
54       } // end method Main
55
56    } // end class Average2
```

gradeValue değişkeni -1 değerinden farklı olduğu sürece döngüyü çalıştırır

Grade değerlerini toplar

Öğrenci sayısını belirlemek için sayaca bir ekle

Kullanıcının başka bir grade değeri girmesi beklenir

Girilen grade değerinin 0 olmadığı kontrol edilir

average değeri hesaplanır

average değeri ekrana yazılır

grade değeri girilmediyse kullanıcı bilgilendirilir

```
Enter Integer Grade, -1 to Quit: 97
Enter Integer Grade, -1 to Quit: 88
Enter Integer Grade, -1 to Quit: 72
Enter Integer Grade, -1 to Quit: -1

Class average is 85.6666666666667
```

Algoritmanın düzenlenmesi: Örnek Çalışma 2 (İçiçe Kontrollü Tekrar)

- İçiçe Kotrol
  - Bir kontrol yapısı diğerinin içerisinde yer alır
    - Çoklu dögü (Multiple loops)
    - `if` deyimleri ile döngüler

Algoritmanın düzenlenmesi: Örnek Çalışma 2 (İçiçe Kontrollü Tekrar)

```
passes değişkenini sıfır yap
failures değişkenini sıfır yap
student değişkenini bir yap

Öğrenci sayacı 10'a eşit veya küçükse
        Yeni sınav sonucu gir

        If öğrenci geçtiyse
                passes değişkenine bir ekle
        Else
                failures değişkenine bir ekle

        öğrenci sayacına bir ekle

Geçen öğrenci sayısını ekrana yaz
Kalan öğrenci sayısını ekrana yaz

Sekiz öğrenciden fazla öğrenci geçtiyse
        Ekrana "Good results" yaz
```

Fig. 4.10 Pseudocode ile sınav sonuçları programı

```
1    // Fig. 4.11: Analysis.cs
2    // Analysis of Examination Results.
3
4    using System;
5
6    class Analysis
7    {
8       static void Main( string[] args )
9       {
10         int passes = 0,          // number of
11             failures = 0,        // number of
12             student = 1,         // student counter
13             result;              // one exam result
14
15         // process 10 students; counter-controlled loop
16         while ( student <= 10 )
17         {
18            Console.Write( "Enter result (1=pass, 2=fail): " );
19            result = Int32.Parse( Console.ReadLine() );
20
21            if ( result == 1 )
22               passes = passes + 1;
23
24            else
25               failures = failures + 1;
26
27            student = student + 1;
28         }
29
```

passes ve failures değişkenlerini 0, öğrenci sayacını 1 yap

While döngüsü 10 defa çalışacaktır

İçiçe if deyimiyle hangi sayaca ekleme yapılacağı belirlenir

Kullanıcı 1 girerse passes değişkenine bir ekle

Kullanıcı 1 girerse failures değişkenine bir ekle

Toplam öğrenci sayısına bir ekle

```
30         // termination phase
31         Console.WriteLine();
32         Console.WriteLine( "Passed: " + passes );
33         Console.WriteLine( "Failed: " + failures );
34
35         if ( passes > 8 )
36            Console.WriteLine( "Good results\n" );
37
38      } // end of method Main
39
40   } // end of class Analysis
```

Sonuçları ekrana yaz

Toplam geçen sayısı 8'den büyükse kullanıcıya bildir

```
Enter result (1=pass, 2=fail): 1
Enter result (1=pass, 2=fail): 2
Enter result (1=pass, 2=fail): 1
Enter result (1=pass, 2=fail): 1
Enter result (1=pass, 2=fail): 1
Enter result (1=pass, 2=fail): 1
Enter result (1=pass, 2=fail): 1
Enter result (1=pass, 2=fail): 1
Enter result (1=pass, 2=fail): 1
Enter result (1=pass, 2=fail): 1

Passed: 9
Failed: 1
Good results
```

```
Enter result (1=pass, 2=fail): 1
Enter result (1=pass, 2=fail): 2
Enter result (1=pass, 2=fail): 2
Enter result (1=pass, 2=fail): 2
Enter result (1=pass, 2=fail): 2
Enter result (1=pass, 2=fail): 2
Enter result (1=pass, 2=fail): 1
Enter result (1=pass, 2=fail): 1
Enter result (1=pass, 2=fail): 1
Enter result (1=pass, 2=fail): 1

Passed: 5
Failed: 5
```

Atama Operatörleri

- Atama operatörleri
  - Kod boyutunu azaltabilir
    - x += 2 ile x = x + 2 aynıdır
    - ++, -=, *=, /=, and %=

Atama Operatörleri

Atama operatörü	Örnek ifade	Açıklama	Sonuç
`int c = 3, d = 5, e = 4, f = 6, g = 12;`			
`+=`	`c += 7`	`c = c + 7`	`c 10 olur`
`-=`	`d -= 4`	`d = d - 4`	`d 1 olur`
`*=`	`e *= 5`	`e = e * 5`	`e 20 olur`
`/=`	`f /= 3`	`f = f / 3`	`f 2 olur`
`%=`	`g %= 9`	`g = g % 9`	`g 3 olur`

Fig. 4.12 Aritmetik atama operatörleri

Artırma ve Azaltma Operatörleri

- Artırma operatörü
  - Değişkene bir ekler
  - x++ ile x = x + 1 aynıdır
- Azaltma operatörü
  - Değişkenden bir çıkarır
  - y--
- Pre-increment ve post-increment
  - x++ veya x--
    - Önce işlem yapılır daha sonra değişkene bir eklenir / çıkarılır
  - ++x veya --x
    - Önce değişkene bir eklenir / çıkarılır daha sonra işlem yapılır

Artırma ve Azaltma Operatörleri

Operatör	Tanımı	Örnek ifade	Açıklama
`++`	preincrement	`++a`	Önce **a** değişkeni 1 artırılır daha sonra yeni değeri kullanılır.
`++`	postincrement	`a++`	Önce **a** değişkeninin değeri kullanılır daha sonra bir artırılır.
`--`	predecrement	`--b`	Önce **b** değişkeni 1 azaltılır daha sonra yeni değeri kullanılır.
`--`	postdecrement	`b--`	Önce **b** değişkeninin değeri kullanılır daha sonra bir azaltılır.

Fig. 4.13 The increment and decrement operators.

```
1    // Fig. 4.14: Increment.cs
2    // Preincrementing and postincrementing
3
4    using System;
5
6    class Increment
7    {
8       static void Main(string[] args)
9       {
10         int c;
11
12         c = 5;
13         Console.WriteLine( c );   // print 5
14         Console.WriteLine( c++ ); // print 5 then postincrement
15         Console.WriteLine( c );   // print 6
16
17         Console.WriteLine();      // skip a line
18
19         c = 5;
20         Console.WriteLine( c );   // print 5
21         Console.WriteLine( ++c ); // preincr
22         Console.WriteLine( c );   // print 6
23
24      } // end of method Main
25
26   } // end of class Increment
```

c değişkeni tanımlanır

c değişkenine 5 atandı

Ekrana c (6) olarak yazılır

c ye 5 atandı

Ekrana c (5) olarak yazıldı

1 eklendi ve ekrana c (6) olarak yazıldı

Ekrana c (6) olarak yazıldı

```
5
5
6

5
6
6
```