Anweisungen

Einfache Anweisungen

Leeranweisung

```
;                                   // ; ist Terminator, nicht Separator
```

Zuweisung

```
x = 3 * y + 1;
```

Methodenaufruf

```
string s = "a,b,c";
string[] parts = s.Split(',');      // Aufruf einer Objektmethode (nicht static)

s = String.Join(" + ", parts);  // Aufruf einer Klassenmethode (static)
```

if-Anweisung


```
if ('0' <= ch && ch <= '9')
      val = ch - '0';
else if ('A' <= ch && ch <= 'Z')
      val = 10 + ch - 'A';
else {
      val = 0;
      Console.WriteLine("invalid character " + ch);
}
```

switch-Anweisung

```
switch (country) {
      case "Germany": case "Austria": case "Switzerland":
            language = "German";
            break;
      case "England": case "USA":
            language = "English";
            break;
      case null:
            Console.WriteLine("no country specified");
            break;
      default:
            Console.WriteLine("don't know language of {0}", country);
            break;
}
```

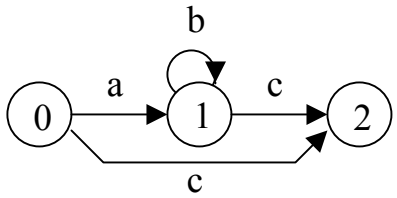
Switch-Ausdruck muß numerisch, char, enum oder string sein (null erlaubt).
Case-Marken müssen mit Switch-Ausdruck kompatibel sein.
Switch mit string-Marken wird sequentiell geprüft.

Jede Case-Anweisungsfolge muß mit break (oder return, goto, throw) enden.
Kein Fall-Through erlaubt.

Wenn keine Marke paßt, wird zu default verzweigt.
Wenn default fehlt, wird nach switch-Anweisung fortgesetzt.

switch mit Sprüngen

Z.B. zur Implementierung endlicher Automaten




```
int state = 0;
int ch = Console.Read();
switch (state) {
    case 0:  if (ch == 'a') { ch = Console.Read(); goto case 1; }
             else if (ch == 'c') goto case 2;
             else goto default;
    case 1:  if (ch == 'b') { ch = Console.Read(); goto case 1; }
             else if (ch == 'c') goto case 2;
             else goto default;
    case 2:  Console.WriteLine("input valid");
             break;
    default: Console.WriteLine("illegal character " + ch);
             break;
}
```

Schleifen

while

```
while (i < n) {
      sum += i;
      ++;
}
```

do while

```
do {
      sum += a[i];
      i--;
} while (i > 0);
```

for

```
for (int i = 0; i < n; i++)
      sum += i;
```

Kurzform für

```
int i = 0;
while (i < n) {
    sum += i;
    i++;
}
```

foreach-Anweisung

Zum Iterieren über beliebige Collections und Arrays

```
int[] a = {3, 17, 4, 8, 2, 29};
foreach (int x in a) sum += x;
```

```
string s = "Hello";
foreach (char ch in s) Console.WriteLine(ch);
```

```
Queue q = new Queue();
q.Enqueue("John"); q.Enqueue("Alice"); ...
foreach (string s in q) Console.WriteLine(s);
```

Sprünge

break;	Zum Aussprung aus Schleifen und switch-Anweisungen. Kein break mit Marke wie in Java (goto verwenden).
continue;	Kann in Schleifen verwendet werden, um an den Schleifenanfang zurückzuspringen.
goto case 3:	Kann in einer switch-Anweisung zum Ansprung einer ihrer case-Marken verwendet werden;

```
myLab:
...
goto myLab;
```

Springt zur Marke myLab.

Restriktionen:

- kein Einsprung in einen Block
- kein Aussprung aus einem finally-Block

return-Anweisung

Aussprung aus Methoden

```
void f(int x) {
    if (x == 0) return;
    ...
}
```

Aussprung aus Funktionsmethoden

```
int max(int a, int b) {
    if (a > b) return a; else return b;
}

class C {
    static int Main() {
        ...
        return errorCode;       // Rückgabe eines Fehlercodes in der Main-Methode
    }                           // eines Programms (kann in Dos-Box mittels Variable
                                // errorlevel abgeprüft werden).
}
```

Konsolen-Ausgabe

Beispiele

```
Console.Write(intVal);                  // überladen für alle Standardtypen;
Console.WriteLine(intVal);              // bei Objekten wird autom. ToString() aufgerufen

Console.Write("Hello {0}", name);       // Platzhalter
Console.WriteLine("{0} = {1}", x, y);
```

Platzhalter-Syntax

"{" n ["," width] [":" format [precision]] "}"

n	Argumentnummer (beginnend bei 0)
width	Feldbreite (wenn zu klein, wird sie überschritten) positiv = rechtsbündig, negativ = linksbündig
format	Formatierungscode (z.B. d, f, e, x, ...)
precision	Anzahl der Nachkommastellen (machmal Anzahl der Ziffern)

Beispiel: {0,10:f2}

Formatierungscode für Zahlen

d, D	Dezimalformat (ganze Zahl mit führenden Nullen) precision = Anz. Ziffern	-xxxxx
f, F	Fixpunktformat precision = Anz. Nachkommastellen (Default = 2)	-xxxxx.xx
n, N	Nummernformat (mit Tausender-Trennstrich) precision = Anz. Nachkommastellen (Default = 2)	-xx,xxx.xx
e, E	Exponentialformat (groß/klein signifikant) precision = Anz.Nachkommastellen	-x.xxxE+xxx
c, C	Currency-Format precision = Anz. Nachkommastellen (Default = 2) Negative Werte werden in Klammern gesetzt	$xx,xxx.xx ($xx,xxx.xx)
x, X	Hexadezimalformat (groß/klein signifikant) precision = Anzahl Hex-Ziffern (evtl. führende 0)	xxx
g, G	General (kompaktestes Format für gegebenen Wert; Default)	

Beispiele


```
int x = 17;

Console.WriteLine("{0}", x);          17
Console.WriteLine("{0,5}", x);           17

Console.WriteLine("{0:d}", x);        17
Console.WriteLine("{0:d5}", x);       00017

Console.WriteLine("{0:f}", x);        17.00
Console.WriteLine("{0:f1}", x);       17.0

Console.WriteLine("{0:E}", x);        1.700000E+001
Console.WriteLine("{0:E1}", x);       1.7E+001

Console.WriteLine("{0:x}", x);        11
Console.WriteLine("{0:x4}", x);       0011
```

Allgemeine Stringformatierung

Mittels *ToString*, für numerische Standardtypen (int, long, short, ...):

```
string s;
int i = 12;
s = i.ToString();              // "12"
s = i.ToString("x4");          // "000c"
s = i.ToString("f");           // "12.00"
```

Mittels *String.Format*, für beliebige Typen

```
s = String.Format("{0} = {1,6:x4}", name, i);      // "val =    000c"
```

Länderspezifische Formatierung

```
s = i.ToString("c");                                // "$12.00"
s = i.ToString("c", new CultureInfo("en-GB"));      // "£12.00"
s = i.ToString("c", new CultureInfo("de-AT"));      // "öS 12.00"
```

Formatierte Ausgabe auf Datei


```
using System;
using System.IO;

class Test {

    static void Main() {
        FileStream s = new FileStream("output.txt", FileMode.Create);
        StreamWriter w = new StreamWriter(s);

        w.WriteLine("Table of sqares:");
        for (int i = 0; i < 10; i++)
            w.WriteLine("{0,3}: {1,5}", i, i*i);

        w.Close();
    }

}
```

Es können nicht mehrere *StreamWriter* gleichzeitig an verschiedene Positionen eines Stream gesetzt werden.

Tastatur-Eingabe

int ch = Console.Read();

Liefert nächstes Zeichen.

Wartet, bis Benutzer Zeile mit CR,LF abgeschlossen hat.
Z.B. Eingabe: "abc" + Return-Taste.
Read liefert der Reihe nach: 'a', 'b', 'c', '\r', '\n'.

Nach letztem Zeichen (Strg-Z + CR + LF) liefert Read() den Wert -1

string line = Console.ReadLine();

Liefert nächste Zeile (nach Strg-Z+CR+LF liefert es null).

Wartet, bis Benutzer Zeile mit CR,LF abgeschlossen hat.
Liefert dann diese Zeile ohne CR, LF.

Es gibt keinen Tokenizer zur formatierten Eingabe wie in Java.

Eingabe von einer Datei


```
using System;
using System.IO;

class Test {

    static void Main() {
        FileStream s = new FileStream("input.txt", FileMode.Open);
        StreamReader r = new StreamReader(s);

        string line = r.ReadLine();
        while (line != null) {
            ...
            line = r.ReadLine();
        }

        r.Close();
    }

}
```

Es können nicht mehrere *StreamReader* gleichzeitig an verschiedene Positionen eines Stream gesetzt werden.

Lesen der Kommandozeilenparameter


```
using System;

class Test {

      static void Main(string[] arg) {                    // Aufruf: Test value = 3
            for (int i = 0; i < arg.Length; i++)
                  Console.WriteLine("{0}: {1}", i, arg[i]);   // Ausgabe (Tokentrennung bei Leerzeichen):
                                                              //   0: value
                                                              //   1: =
                                                              //   2: 3
            foreach (string s in arg)
                  Console.WriteLine(s);                       // Ähnliche Ausgabe wie oben
      }
}
```