Die Wiederholung (Iteration)/Schleifen

Im Verlauf eines Algorithmus müssen manchmal eine oder mehrere Anweisungen wiederholt ausgeführt werden. Solche Wiederholungen nennt man auch **Schleifen**. Es gibt mehrere Arten von Schleifen:

- Die Anzahl der Schleifendurchläufe ist nicht von vornherein bekannt und wird während des Programmauslaufs durch eine **Bedingung** gesteuert, die bei jedem Schleifendurchlauf **am Schleifenanfang abgefragt** wird (in C#: **while-Schleife**).
- Die Anzahl der Schleifendurchläufe ist nicht von vornherein bekannt und wird während des Programmlaufs durch eine **Bedingung** gesteuert, die am **Schleifenende abgefragt** wird (in C#: **do-while-Schleife**).
- Die **Anzahl der Schleifendurchläufe** ist beim Erstellen des Programms von vornherein **bekannt (**in C#: **for-Schleife)**.
- In C# gibt es noch weitere Arten von Wiederholungen, u.a. die for-each-Schleife oder über LINQ. Damit man beides sinnvoll nutzen kann, muss man sich jedoch ein wenig intensiver mit objektorientierter Programmierung beschäftigen als es im Moment notwendig ist.

Wir werden uns zunächst theoretisch mit dem Konzept von Schleifen beschäftigen und uns erst danach um die Umsetzung in C# kümmern.

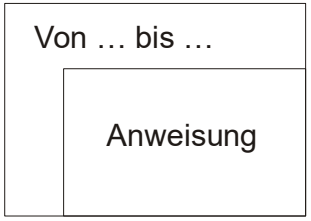
Aufgaben:

1. In einem Text sollen alle doppelten Leerstellen entfernt werden. Wie geht das? Was setzt du bei dem voraus, der dein Verfahren ausführt?
2. Gesucht sind alle Primzahlen zwischen 1 und n. Für die Lösung der Aufgabe bietet sich das „Sieb des Eratosthenes“ an. Die Idee in der Umgangssprache lautet:
   Schreibe die Zahlen 1 bis n nieder. Streiche daraus die Vielfachen von 2, 3, 5 usw. bis zur Quadratwurzel aus n. Die nicht gestrichenen Zahlen sind die Primzahlen.
   a) Warum sprechen wir hier nur von einer Idee für den Algorithmus?
   b) Führe den Algorithmus für ein Beispiel (z. B. n = 36) aus.
   c) Stelle den Algorithmus mit Hilfe eines Struktogrammes dar.
3. In der Informatik spielen Iterationsverfahren eine große Rolle. Ein besonders einfaches und altes Verfahren ist das Heron-Verfahren zur Bestimmung der Quadratwurzel einer Zahl a.
   Für einen Schritt gilt: $x_{n+1} = \frac{1}{2}\left(x_n + \frac{a}{x_n}\right)$ Führe das Verfahren für a = 2 aus, bis sich x_n und x_{n+1} um 0,001 unterscheiden und zeichne ein Struktogramm.

a) Die Zählschleife (for-Schleife)

Wenn bereits bei der Erstellung eines Programms die Anzahl der notwendigen Schleifendurchläufe bekannt ist, kann man eine for-Schleife benutzen. Deshalb wird die for-Schleife auch als Zählschleife bezeichnet.

Das Struktogramm für eine Zählschleife ist:



Leider ist der Aufbau in C# wesentlich komplizierter und kann durch das Struktogramm nicht vollständig erfasst werden.

Formulierung in C#:

```
for(Initialisierung; Ausdruck; Zählanweisung)
  Anweisung;
```

Ist die Anweisung aus mehreren Einzelanweisungen zusammengesetzt, so müssen diese durch { und } zusammengefasst werden, wie bei Verzweigungen.

Betrachten wir nun die for-Schleife etwas genauer:

- Im Teil `Initialisierung` wird die Zählvariable initialisiert, d.h. auf den Startwert gesetzt. Dieser Teil wird nur einmal durchlaufen.
  Außerdem wird häufig sogar die Zählvariable deklariert. Dadurch ist sie nur innerhalb der Schleife verfügbar und kann außerhalb nicht verwendet werden.
- Der Teil `Ausdruck` steht ein logischer Ausdruck, der entweder `true` oder `false` ist. Solange er true ist, wird die Schleife durchlaufen. Man spricht in diesem Zusammenhang auch gerne von Abbruchbedingung oder Schleifenbedingung.
- Im Teil `Zählanweisung` wird der Wert der Zählvariable geändert.

Beispiel:

Da diese allgemeine Erläuterungen recht kompliziert erscheinen, schauen wir uns am bestens einige Code-Beispiele an:

```
void Button1Click(object sender, System.EventArgs e)
{
     int i;    // Zählvariable deklarieren

     // eigentliche for-Schleife
     for(i=1;i<4;i++)
     {
        // Was passiert jetzt hier?
        MessageBox.Show(i.ToString());
     }
}
```

Die for-Schleife beginnt mit 1 (Initialisierung: i=1). Bei jedem Schleifendurchgang wird i um 1 erhöht (Zählanweisung: i++). Die Schleife wird solange wiederholt, bis i nicht mehr kleiner als 4 ist, d.h. i größer oder gleich 4 ist (Ausdruck: i<4). Beim Ausprobieren des Programms erscheint eine MessageBox mit 1, mit 2 und mit 3. Zählvariablen werden in der Regel mit i, j, k usw. bezeichnet. i steht für index.

Aufgabe:

4. a) Ändere das Beispielprogramm so ab, dass die MessageBox fünfmal ausgegeben wird.
   b) Ändere das Programm aus a), so dass die Zählung bei 3 beginnt.
5. Schreibe ein **neues** Programm, in dem alle ganzen Zahlen bis 10 ausgegeben werden.
6. Schreibe ein Programm zur Aufg. 3. Es soll das n-te Folgenglied berechnet werden.

Es ist empfehlenswert, die Zählvariable erst innerhalb der for-Schleife zu deklarieren. Dann ist sie nur innerhalb des nachfolgendes Blockes zwischen { und } vorhanden. Das Beispiel kann man folgendermaßen umschreiben:

```
void Button2Click(object sender, System.EventArgs e)
{
      // eigentliche for-Schleife
      for(int i=1;i<4;i++)
      {
        MessageBox.Show(i.ToString());
      }
}
```

Beispiel: „Aufaddieren“

Das folgende Programm addiert die Zahlen 1 bis 10 und gibt das Ergebnis in label2 aus:

```
void Button1Click(object sender, System.EventArgs e)
{
  // Variable definieren und auf sinnvollen Anfangswert
  // für eine Addition setzen
  int summe=0;

  // eigentliches Addieren:
  for(int i=1;i<=10;i++)
  {
    // zum Wert von summe wird der aktuelle i addiert
    summe=summe+i;
  }

  // Ergebnis ausgeben
  label2.Text=summe.ToString();
}
```

Aufgaben:

7. Erstelle ein Programm, das mit Hilfe einer for-Schleife die Fakultät von x berechnet. Bsp.: 5! = 1*2*3*4*5 = 120
8. Erstelle ein Programm, das die Summe aller ganzen Zahlen zwischen einer Anfangszahl und einer Endzahl mit Hilfe einer for-Schleife berechnet.
9. Erstelle ein Programm, das eine einzugebende ganzzahlige Potenz mit Hilfe einer Schleife berechnet („a hoch b“). Verwende nicht die in C# vorhandene Methode Math.Power. Du kannst die Methode aber zum Vergleich verwenden!
10. Erstelle ein Programm, das nach Eingabe einer natürlichen Zahl n die Fibonacci-Zahl von n mit Hilfe einer for-Schleife berechnet. (*Vorsicht: kompliziert!*)

    Definition Fibonacci-Zahlen: Fib (1) = 1
    Fib (2) = 1
    Fib (n+2) = Fib (n+1) + Fib (n)

    d.h. Fib (3) = 2 Fib (4) = 3 Fib (5) = 5 Fib (6) = 8 Fib (7) = 13

Beispiel: Regelmäßiges Sechseck mit der Turtle-Komponente (aus den InfoKursTools)

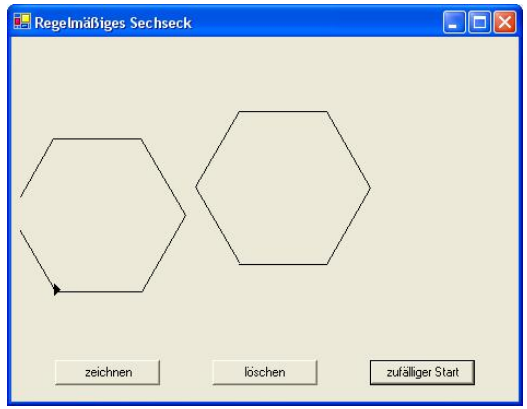
```
void Button1Click(object sender, System.EventArgs e) // Button "zeichnen"
{
  // Startposition setzen
  turtle1.MoveTurtleTo(200,200);
  // regelmäßiges Sechseck zeichnen
  for(int i=1;i<=6;i++)
  {
    // zeichne eine Seite
    turtle1.DrawTurtle(80);
    // Innenwinkel im 6-Eck ist 120°
    // der Außenwinkel ist hier
    // 180° - 120° = 60 °
    // Dieser muss für die Drehung
    an// gegeben werden
    turtle1.TurnLeft(60);
  }
}

void Button2Click(object sender, Sys-
tem.EventArgs e)  // Button "löschen"
{
  turtle1.Clear();
}

void Button3Click(object sender, System.EventArgs e)
// Ereignisbehandlung für Button "zufälliger Start"
{
  // Obergrenze fuer Zufallszahlen bestimmen
  randomNumbers1.Max=turtle1.Height;

  // Koordinaten zufällig bestimmen
  int startX=randomNumbers1.Next();
  int startY=randomNumbers1.Next();

  // Startposition setzen
  turtle1.MoveTurtleTo(startX,startY);
  // regelmäßiges Sechseck zeichnen
  for(int i=1;i<=6;i++)
  {
    // zeichne eine Seite
    turtle1.DrawTurtle(80);
    // Innenwinkel im 6-Eck ist 120 Grad
    // der Außenwinkel ist hier 180 Grad - 120 Grad = 60 Grad
    // Dieser muss fuer die Drehung angegeben werden
    turtle1.TurnLeft(60);
  }
}
```



In diesem Beispiel wird gezeigt, wie man mit Hilfe der Turtle-Komponente regelmäßige Sechsecke zeichnen kann. Die Ereignisbehandlungen `Button1Click` und `Button3Click` sind recht ähnlich. In `Button3Click` wird mit Hilfe der Komponente RandomNumbers der Startpunkt zufällig erwähnt.

Aufgaben:

11. Schreibe ein Programm, mit dem regelmäßige Achtecke zeichnen kann. Orientiere dich am Beispiel.
12. Verallgemeinere das Programm aus Aufgabe 10, so dass jetzt die Anzahl der Ecken n sowie die Seitenlänge vom Benutzer eingelesen wird und dann ein regelmäßiges n-Eck gezeichnet wird.

