Weitere Aufgaben zu Schleifen, Umgang mit Komponenten

Bei der Arbeit mit Schleifen ist eine neue grafische Komponente praktisch: die Listbox. In ihr können mehrere Elemente gleichzeitig angezeigt werden.

Beispiel (hier mit einer For-Schleife):

```
void Button1Click(object sender, System.EventArgs e)
{
      // Schleife von 0 bis 10
      for(int i=0;i<=10;i++)
{
        // fügt einen Eintrag in listbox1 ein
        listBox1.Items.Add(i);
      }
}

void Button2Click(object sender,
                  System.EventArgs e)
{
      // löscht alle Einträge in der Listbox
      // listbox1
      listBox1.Items.Clear();
}
```

Aufgaben:

1. Schreibe ein C#-Programm, das eine Wertetabelle für den Sinus in einer Listbox ausgibt. Den Sinus einer Zahl x erhält man durch `Math.Sin(x)`. x muss man dabei im Bogemaß angeben.
2. Der Computer soll nach Eingabe einer natürlichen Zahl kleiner als 20 (Kontrolle durch das Programm!) die nächsten 15 natürlichen Zahlen zusammen mit ihrem Quadrat in einer Listbox ausgeben. Erstelle ein Struktogramm und ein C#-Programm!
3. Schreibe ein C#-Programm, das die Teiler aller Zahlen bis zu einer vorgegebenen Zahl n bestimmt und in einer Listbox ausgibt.
   Ein ganze Zahl t teilt die ganze Zahl k, wenn der Rest der (ganzzahligen) Division k:t null ist. Die Bestimmung des Rests nennt man „modulo“ (kurz: mod). In C# berechnet man den Rest durch %.
4. Die Strahlung eines radioaktiven Stoffes nimmt stündlich um 5 % ab. Die Intensität besteht am Anfang 2000 Strahlungseinheiten. Berechne die Radioaktivität nach 1, 2, 3, usw. Stunden. Lies die gewünschte Stundenzahl ein.
5. Michael hat 380 € geschenkt bekommen. Er möchte das Geld anlegen. Nach wie viel Jahren hat sich sein Guthaben verdoppelt, wenn die Bank ihm 2,1 % Zinsen anbietet?
   a) Wie muss man an dieses Problem mathematisch herangehen? Was gilt?
   b) Schreibe ein C#-Programm, das das Guthaben bis zur Verdoppelungszeit ausgibt.
   c) Verallgemeinere das Programm!
6. Schreibe ein Programm, mit dem durch reines Probieren nachgeprüft werden kann, ob drei natürliche Zahlen a, b, c die Gleichung $a^2 + b^2 = c^2$ erfüllen. a, b, c nennt man in diesem Fall „Pythagoräisches Tripel“. Das Programm soll eine Obergrenze einlesen und alle ganzen Zahlen a, b, c bis zu dieser Grenze überprüfen.
7. Ändere Aufgabe 9 (Fibonacci-Zahlen) vom letzten Blatt, so dass jetzt jede Fibonacci-Zahl bis zur gewünschten Platznummer ausgegeben wird.

8. Ändere Aufgabe 10 bzw. 11 (n-Eck mit der Turtle), so dass jetzt die Koordinaten der Eckpunkte in einer Listbox ausgegeben werden.

Nützliche Methoden und Eigenschaften der Listbox:

Wir betrachten in der Übersicht eine ListBox mit dem Namen listBox1:

Methode/Eigenschaft	Funktion
`listBox1.Items.Add(Objekt);`	Objekt der Listbox am Ende hinzufügen
`listBox1.Items.Insert(nr,Objekt)`	Objekt an Position nr (ab 0) in ListBox einfügen
`listBox1.Items.Clear();`	Alle Einträge in der ListBox löschen
`int nr=listBox1.SelectedIndex;`	Position (ab 0) des ausgewählten Eintrags in der ListBox oder -1, wenn nichts ausgewählt wurde
`Object o=listBox1.SelectedItem;`	Liefert das Objekt des ausgewählten Eintrags zurück oder `null`, wenn nichts ausgewählt wurde
`listBox1.SelectedItem=null;`	Alle Einträge werden in der ListBox abgewählt.
`listBox1.Items.RemoveAt(nr)`	Element an Position nr (ab 0) löschen
`int anz=listBox1.Items.Count;`	Anzahl der Elemente in ListBox
`listBox1.Items.Contains(Objekt);`	Prüft, ob ListBox Objekt enthält (kann aber je nach Klasse des Objekts Probleme verursachen)
`Int nr=listBox1.Items.IndexOf(Objekt);`	Liefert die Position des Objektes zurück (kann aber je nach Klasse des Objekts Probleme verursachen)
`Object o=listBox1.Items[nr];`	Liefert das Objekt an der Position nr aus der ListBox zurück

Die meisten der genannten Methoden und Eigenschaften werden in einem Beispielprogramm auf der Kursseite oder in den Online-Demos angewendet.

Wenn man die Einträge schon beim Design anlegen möchte, trägt man sie direkt bei der Eigenschaft `Items` im Eigenschaften-Fenster ein.

Wenn man schon beim Programmstart einen Eintrag in einer ListBox markieren möchte, muss man dazu das Formular auswählen, auf die Ereignisseite (Blitz) des Eigenschaften-Fensters klicken und das FormLoad Ereignis auswählen bzw. erzeugen. Dann kann man folgenden Code ergänzen:

```
void MainFormLoad(object sender, EventArgs e)
{
      listBox1.SelectedIndex=0;  // Alternativ: SelectedItem setzen
}
```

Alternative Combobox

Wenn man wenig Platz auf dem Formular hat oder einige Auswahlmöglichkeiten anbieten muss, bietet sich als Alternative die Combobox an. Comboxen sind eine Kombination aus Textfeld (TextBox) und ListBox.

Entscheidend für das Verhalten der Combobox ist die Eigenschaft `DropDownStyle`. Wenn der DropDownStyle auf `DropDownList` gesetzt ist, kann man nichts im Textfeld eingeben, sondern nur Einträge aus der aufklappbaren Liste auswählen. Damit verhält sich die Combobox wie eine Listbox. Zusätzlich kann aber den ausgewählten Eintrag mit der Eigenschaft `Text` bestimmt werden.

Wenn man den DropDownStyle auf `DropDown` setzt, kann man entweder die Einträge aus der Liste auswählen oder einen eigenen Text eingeben. Die gewünschte Eingabe kann man auf jedem Fall aus der Eigenschaft Text auslesen.

Zwei Beispiele zur Arbeit mit Comboboxen befinden sich ebenfalls auf der Kursseite.