C# Programmierung

Sammlung nützlicher Beispiele, Tricks, Ideen und Hinweise

Michael Koch

Version vom 27. 12. 2024

Inhaltsverzeichnis

1 Visual Studio

- **Eine Projektmappe kann mehrere Projekte enthalten**
- **Bei merkwürdigen Problemen: Erstellen / Projektmappe neu erstellen**
- **Man kann in Visual Studio immer nur ein Projekt offen haben. Alternative: Visual Studio zweimal starten**
- **Debuggen beenden, bevor man im Code ändert**

1.1 Ein Projekt kopieren

Methode 1:

Der komplette Projektordner in Eigene_Dokumente/Visual_Studio_2013/Projects/ wird kopiert und umbenannt. Dann wird das umbenannte Projekt in Visual Studio geöffnet (und zwar die *.sln Datei) und damit weitergearbeitet. Man kann dann noch im Projektmappen-Explorer den Projektnamen ändern (Rechtsklick --> umbenennen), und im Quellcode den Namespace ändern (Rechtsklick auf den Namespace --> umgestalten --> umbenennen).

Methode 2:

1. Das vorhandene Projekt wird in Visual Studio geöffnet.

2. Datei --> Vorlage exportieren --> weiter --> fertig stellen (Bei VS2019: Projekt → Vorlage exportieren)

3. Das Projekt befindet sich jetzt als zip-Datei im Ordner "My exported templates"

4. Datei --> neu --> Projekt

5. In der Auflistung, wo die verschiedenen Projektarten stehen, ganz nach unten scrollen. Dort sollte die Vorlage stehen, die wir soeben erzeugt haben. Die wird ausgewählt.

6. Bei "Name" einen neuen Projektnamen vergeben

7. Der gleiche Name wird auch für die neue Projektmappe vorgeschlagen, das ist ok. Das Häkchen bei "Projektmappenverzeichnis erstellen" sollte gesetzt sein.

8. Auf "ok" klicken.

9. Das neue Projekt ist jetzt vorhanden und kann sofort verwendet werden.

10. Wenn die zip-Datei nicht mehr benötigt wird, kann sie gelöscht werden.

Anmerkung im November 2024:

Eine Kleinigkeit hat sich geändert. Der Punkt "Vorlage exportieren" ist nicht mehr nicht unter Datei, sondern unter "Projekt" zu finden.

1.2 Tastatur-Kürzel

Alt-Tab wechselt zwischen Fenstern

F5 übersetzen + starten

F9 Haltepunkt an / aus

F10 Prozedurschritt (Methodenaufrufe werden überspringen)

F11 Einzelschritt (zeilenweise)

Control-Leertaste öffnet das Fenster mit Vorschlägen

TAB vervollständigt

Control-R Control-R Variablen umbenennen

ESC wechselt zum übergeordneten Objekt

Control-Rollrad zum Zoomen

Control-A alles markieren

Control-A Control-K Control-D Codeblock formatieren

Control-Z zurück bei falschen Vorschlägen

1.3 Reverse Engineering erschweren

-- Dotfuscator

-- regelmässig Updates veröffentlichen

-- Seriennumern von angeschlossenen Geräten abfragen

-- Spezifische Fehlermeldungen mit mehreren Wochen Verzögerung anzeigen

-- Funktionen in dll auslagern

-- siehe auch Jones/Freeman: Visual C# Recipes: Seite 30

1.4 Anwendungsweitergabe

siehe Dirk Frischalowski: Visual C# 2010, Kapitel 20

siehe Walter Doberenz, Thomas Gewinnuns: "Visual C# 2010: Grundlagen und Profiwissen" ab Seite 811

1.5 Copy & Paste mit Syntax Highlighting

Siehe Extras / Optionen / Text-Editor / Erweitert, dort das Häkchen bei "Beim Kopieren/Ausschneiden Richt Text kopieren" setzen und darunter die maximale Länge so einstellen, dass sie größer als der zu kopierende Block ist. Sonst wird bei größeren Blöcken nämlich nur Text ohne Syntax-Highlighting kopiert.

2 C# Sprachelemente

2.1 Arrays, foreach

Eindimensionale Arrays:

```
int[] zahlen1 = new int[10];        // Das eindimensionale Array wird deklariert und erzeugt
zahlen1[5] = 3;
int[] zahlen2 = {2,3,5};            // Das eindimensionale Array wird gleichzeitig deklariert, erzeugt und initialisiert
int[] zahlen3 = new int[] {4,5,6};  // Das eindimensionale Array wird gleichzeitig deklariert, erzeugt und initialisiert
int[] zahlen4;                      // Das eindimensionale Array wird zuerst deklariert
zahlen4 = new int[10];              //  und danach erzeugt

for (int i = 0; i < zahlen2.Length; i++)
    Console.WriteLine("Zahl: {0}: {1}", i, zahlen2[i]);

foreach(int zahl in zahlen3)
    Console.WriteLine("Zahl: {0}", zahl);

Console.WriteLine("Dimensionen: {0}", zahlen4.Rank); // hier 1
```

Wenn man ein bereits existierendes Array mit einer Auflistung füllen möchte, kann man z.B. den Inhalt aus einen statischen Array in das Ziel-Array kopieren:

```
byte[] b = { 0x01, 0x05, 0x86, 0x00 };               // Array wird deklariert, erzeugt und initialisiert
new byte[] { 0xff, 0x01, 0x83, 0x30 }.CopyTo(b, 0);   // Das bereits existierende Array wird mit neuen Werten gefüllt
```

Oder noch einfacher kann man das so machen:

```
byte[] b = { 0x01, 0x05, 0x86, 0x00 };          // Array wird deklariert, erzeugt und initialisiert
b = new byte[] { 0xff, 0x01, 0x83, 0x30 };      // Das bereits existierende Array wird mit neuen Werten gefüllt
```

siehe auch Buch Visual C# 2010, Seite 197

Zweidimensionale Arrays:

```
int[,] matrix1 = new int[2, 3]; // ein zweidimensionales array erzeugen
int[,] matrix2 = new int[2, 3] {{2,4,6},{1,3,5}};   // oder so
int zeilen = 3;
int spalten = 10;
int[,] matrix3 = new int[zeilen, spalten];  // oder so

for (int i = 0; i < matrix2.GetLength(0); i++)
    for (int j = 0; j < matrix2.GetLength(1); j++)
        Console.WriteLine("{0},{1}: {2}", i, j, matrix2[i,j]);

for (int dim = 0; dim < matrix2.Rank; dim++)
    Console.WriteLine("Größe der Dimension {0}: {1}", dim, matrix2.GetLength(dim));

// Ausgabe:
// 0,0: 2
// 0,1: 4
// 0,2: 6
// 1,0: 1
// 1,1: 3
// 1,2: 5
// Größe der Dimension 0: 2
// Größe der Dimension 1: 3
```

In JAVA sieht das etwas anders aus:

```
int[][] = new int[10][5];
```

2.2 Mathematische Funktionen

```
double a = 3.333;
double b;

b = Math.Sqrt(a);

// Zufall
r = new Random();
rx = r.NextDouble();  // 0..1
```

2.3 Formatierte Zahlen-Ausgabe

Formatierte Zahlen-Ausgabe mit ToString()

```
using System;
using System.Windows.Forms;
using System.Globalization;

namespace test
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            CultureInfo invC = CultureInfo.InvariantCulture;
            double a = 0.12345;
            richTextBox1.AppendText(a.ToString("1234567890\n"));                // 1234567890
            richTextBox1.AppendText(a.ToString() + "\n");                       // 0,12345
            richTextBox1.AppendText(a.ToString(invC) + "\n");                   // 0.12345
            richTextBox1.AppendText(a.ToString("F3") + "\n");                   // 0,123
            richTextBox1.AppendText(a.ToString("F3", invC) + "\n");             // 0.123
            richTextBox1.AppendText(a.ToString().PadLeft(10) + "\n");           //    0,12345
            richTextBox1.AppendText(a.ToString(invC).PadLeft(10) + "\n");       //    0.12345
            richTextBox1.AppendText(a.ToString("F3").PadLeft(10) + "\n");       //      0,123
            richTextBox1.AppendText(a.ToString("F3", invC).PadLeft(10) + "\n"); //      0.123
            richTextBox1.AppendText(a.ToString("000.000") + "\n");              // 000,123
            richTextBox1.AppendText(a.ToString("000.000", invC) + "\n");        // 000.123
            richTextBox1.AppendText(a.ToString("0.000") + "\n");                // 0,123
            richTextBox1.AppendText(a.ToString("0.000", invC) + "\n");          // 0.123
            richTextBox1.AppendText(a.ToString("##0.000") + "\n");              // 0,123
            richTextBox1.AppendText(a.ToString("##0.000", invC) + "\n");        // 0.123
            richTextBox1.AppendText(a.ToString("###.000") + "\n");              // ,123
            richTextBox1.AppendText(a.ToString("###.000", invC) + "\n");        // .123
        }
    }
}
```

Siehe Dirk Frischalowski: "Visual C# 2010", Seite 217

Merke: Im Formatstring steht 0 für eine Ziffer die immer ausgegeben wird, und # für eine Ziffer die nur ausgegeben wird wenn sie signifikant ist

Formatierte Zahlen-Ausgabe in die Konsole:

```
using System;
using System.Windows.Forms;
using System.Globalization;

namespace test
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            CultureInfo invC = CultureInfo.InvariantCulture;
            double a = 0.12345;
            Console.WriteLine("1234567890");                            // 1234567890
            Console.WriteLine(a);                                       // 0,12345
            Console.WriteLine(String.Format(invC, "{0}", a));           // 0.12345
            Console.WriteLine("{0:F3}", a);                             // 0,123
            Console.WriteLine(String.Format(invC, "{0:F3}", a));        // 0.123
            Console.WriteLine("{0,10}",a);                              //    0,12345
            Console.WriteLine(String.Format(invC, "{0,10}", a));        //    0.12345
            Console.WriteLine("{0,10:F3}", a);                          //      0,123
            Console.WriteLine(String.Format(invC,"{0,10:0.000}", a));   //      0.123
            Console.WriteLine(a.ToString("F3").PadLeft(10));            //      0,123
            Console.WriteLine(a.ToString("F3", invC).PadLeft(10));      //      0.123
            Console.WriteLine(a.ToString("000000.000"));                // 000000,123
            Console.WriteLine(a.ToString("000000.000", invC));          // 000000.123
            Console.WriteLine("{0:000000.000}", a);                     // 000000,123
            Console.WriteLine(String.Format(invC, "{0:000000.000}", a)); // 000000.123
        }
    }
}
```

Ein Vorteil von Console.WriteLine() gegenüber richTextBox.AppendText() ist, dass der Zeilenvorschub automatisch gemacht wird.

Hexadezimale Zahlen 2-stellig ausgeben: richTextBox1.AppendText(a.ToString("X2"));

2.4 Escape-Sequenzen

\' Apostroph	\0 null (z.B. Ende einer Zeichenkette)	\r Wagenrücklauf
\" Anführungszeichen	\b Rückschritt	\t Tabulator
\\ Backslash	\n Zeilenwechsel	

2.5 switch case

```
switch (x)
{
    case 0:  a = "hallo";
             break;
    case 1:  a = "huhu";
             break;
    default: a = "";
             break;
}
```

2.6 for

```
for(int zahl: zahlen)  // ist äquivalent zu:
for(int i = 0; i < zahlen.length; i++)
```

2.7 Exceptions

```
throw new Exception("Fehler 5 ist aufgetreten !");
```

```
Beispiel 1 für try/catch/finally:

String s;
try
{
    int zahl = Convert.ToInt32(s);
}
catch (Exception e)
{
    Console.WriteLine("Das ist keine Zahl");
    Console.WriteLine("Laufzeitmeldung: {0}", e.Message);
}
finally
{
    Console.WriteLine("fertig.");  // wird immer ausgeführt, auch dann wenn
                                   //  ein unbehandelter Fehler auftritt
}


Beispiel 2 für try / catch:

try
{
    int wert1 = Convert.ToInt32(TextBox1.Text);
}
catch (OverflowException ex)
{
    TextBox2.Text = "ist zu groß";
}
catch (FormatException ex)
{
    TextBox2.Text = "ist keine Zahl";
}


Beispiel 3:

try
{
}
catch (Exception ex)
{
    if (ex is OverflowException)
```

```
    {
    }
}
```

2.8 Blöcke auskommentieren

```
/**/         // wenn man hier hinter /** ein Leerzeichen einfügt /** /
             //  dann wird der gesamte Block auskommentiert
(Codeblock)

/**/
```

2.9 var

Implizit typisierte lokale Variable. Der Typ wird von Compiler automatisch festgelegt. Am besten vermeiden, man braucht das nicht wirklich.

2.10 Bedingungsoperator ?:

```
int i = 10;
if (a > 0) i = 20;

// ist äquivalent zu:

i = (a > 0) ? 20 : 10;
```

2.11 Unterschiede zwischen JAVA und C#

	JAVA	**C#**
	boolean	bool
	System.out.println("Hallo");	System.Console.Out.WriteLine("Hallo");
	String.valueOf(variable)	variable.ToString()
Zugriffs-Modifizierer:	public	public
	private	private
	protected	protected
	Package-privat <-- default	protected internal
		internal <-- default für Klassen
	Die *.jar Datei enthält das ganze Projekt als zip-Datei	

	Auf eine static Variable innerhalb der Klasse kann über jede Instanz zugegriffen werden:
	
	public class MyClass {
	static boolean ok;
	
	public MyClass() { // Konstruktor
	}
	public static void main(String[] args) {
	MyClass objekt1 = new MyClass();
	objekt1.ok = true;
	}
	}
		Auf eine static Variable innerhalb der Klasse kann nur über den Klassennamen zugegriffen werden:
		
		class MyClass
		{
		public static bool ok;
		public MyClass() // Konstruktor
		{}
		}
		MyClass objekt1 = new MyClass();
		MyClass.ok = true; // nicht: objekt1.ok = true

2.12 Partielle Klassen

Wenn eine Klasse auf mehrere Dateien verteilt werden soll, dann müssen alle (Teil-) Klassen als partial gekennzeichnet werden. Alle Teilklassen werden vom Compiler zu einer Klasse zusammengefasst.

```
partial class Mathe
{

}
```

2.13 enum (Aufzählungstypen)

```
public enum infolaenge { il_lang, il_kurz };

public void info(infolaenge il)    // überladene Methode
{
    if (il == infolaenge.il_lang)
    {
         Console.WriteLine("Lange Info....);
    }
    else
    {
         // mache was anderes
    }
}
```

2.14 Liste / list

Dies ist eine Klasse:

```
namespace Units_Test
{
    public class PhysicalUnit
    {
        public string name;
        public int s;
        public int m;
        public int kg;
        public int A;

        public PhysicalUnit(string name, int s, int m, int kg, int A)   // Konstruktor
        {
            this.name = name;
            this.s = s;
            this.m = m;
            this.kg = kg;
            this.A = A;
        }
    }
}
```

Hier werden Elemente zu einer Liste hinzugefügt oder entfernt:

```
using System;
using System.Drawing;
using System.Windows.Forms;

namespace Units_Test
{
    public partial class Form1 : Form
    {
```

```
        System.Collections.Generic.List<PhysicalUnit> einheiten = new System.Collections.Generic.List<PhysicalUnit>();

        public Form1()
        {
            InitializeComponent();

            einheiten.Add(new PhysicalUnit("Hz",   -1, 0, 0, 0));
            einheiten.Add(new PhysicalUnit("N",    -2, 1, 1, 0));
            einheiten.Add(new PhysicalUnit("W",    -3, 2, 1, 0));
            einheiten.Add(new PhysicalUnit("J",    -2, 2, 1, 0));
            einheiten.Add(new PhysicalUnit("Pa",   -2,-1, 1, 0));
            einheiten.Add(new PhysicalUnit("C",     1, 0, 0, 1));
            einheiten.Add(new PhysicalUnit("V",    -3, 2, 1,-1));
            einheiten.Add(new PhysicalUnit("Ω",    -3, 2, 1,-2));
            einheiten.Add(new PhysicalUnit("F",     4,-2,-1, 2));
            einheiten.Add(new PhysicalUnit("H",    -2, 2, 1,-2));
            einheiten.Add(new PhysicalUnit("Wb",   -2, 2, 1,-1));
            einheiten.Add(new PhysicalUnit("T",    -2, 0, 1,-1));

            foreach (PhysicalUnit pu in einheiten)
                richTextBox1.AppendText(pu.name + " ");
            richTextBox1.AppendText("\n");

            einheiten.RemoveAll(r => r.name == "Wb");      // entferne alle Elemente mit diesem Namen

            foreach (PhysicalUnit pu in einheiten)
                richTextBox1.AppendText(pu.name + " ");
            richTextBox1.AppendText("\n");

            einheiten.RemoveAll(r => r.s == -2);           // entferne alle Elemente, bei denen s == -2 ist

            foreach (PhysicalUnit pu in einheiten)
                richTextBox1.AppendText(pu.name + " ");
            richTextBox1.AppendText("\n");
        }
    }
}
```

3 Grundsätzlicher Aufbau des Programms

-- Eine Klasse ohne Konstruktor entspricht einer C-Struktur. Es gibt in C# aber auch Strukturen, die sich von Klassen dadurch unterschieden dass sie nicht vererbbar sind.

-- Eine Klasse kann mehrere Konstruktoren haben. Sie müssen sich in

der Anzahl oder im Typ in der Parameterliste unterscheiden

-- "static" hat in C# und JAVA eine andere Bedeutung als in C.

Eigenschaften einer Klasse, die als "static" gekennzeichnet sind,

gibt es nur einmal.

Zugriff in C#: MyClass.zahl = 3; // Zugriff über den Klassen-Namen

Zugriff in JAVA: MyInstance.zahl = 3; // Zugriff ist über den Klassen-Namen oder über jede Instanz möglich

Zugriffs-Modifizierer:

private	**Zugriff nur innerhalb der Klasse**
protected	**Zugriff nur innerhalb der Klasse, und von abgeleiteten Klassen aus**
public	**Zugriff von überall aus möglich**

3.1 Accessoren, get set

```
public class MyClass
{
    private int a = 5;     // die private Variable wird kleingeschrieben
                           //
                           // für jede Eigenschaft die als private
                           //  gekennzeichnet ist, muss es Accessoren geben
                           //
                           // Rechtsklick auf "a", dann Refactor--> Encapsulate Field
                           //  erzeugt automatisch die get- und set-Methoden

    public int A           // Die öffentlich zugängliche Eigenschaft wird groß geschrieben
    {
        get
        {
            return a;
        }
        set
        {
            a = value;
        }
    }
}


public partial class Form1 : Form
{
    public Form1()
    {
        InitializeComponent();
    }

    private void button1_Click(object sender, EventArgs e)
    {
        MyClass mc = new MyClass();
        mc.A = 22;
        System.Console.WriteLine(mc.A);
    }
}
```

Es ist auch möglich ein Array von Eigenschaften zu verwenden, wie in diesem Beispiel:

```
using System;
using System.Windows.Forms;

namespace test
{
    public partial class Form1 : Form
    {
        public class MeineKlasse
        {
            private int[] a;

            public int[] A
            {
                get { return a; }
                set { a = value; }
            }

            public MeineKlasse()   // Konstruktor
            {
                a = new int[10];
            }
        }

        public Form1()
        {
            MeineKlasse instanz1 = new MeineKlasse();
            MeineKlasse instanz2 = new MeineKlasse();
            instanz1.A[3] = 33;
            instanz1.A[4] = 44;
            instanz2.A[3] = 66;
            instanz2.A[4] = 88;
            richTextBox1.AppendText(instanz1.A[3].ToString() + " " + instanz1.A[4].ToString() + "\n");
            richTextBox1.AppendText(instanz2.A[3].ToString() + " " + instanz2.A[4].ToString() + "\n");
        }
    }
}
```

3.2 Vererbung

```
public class Basisklasse    // Basisklasse, Klassen werden groß geschrieben
{
    private int breite = 5;  // die Zuweisung "=5" hätte man auch in den
                             //   Konstruktor schreiben können

    public int Breite        // Eigenschaften werden auch groß geschrieben
    {
        get { return breite; }    // Dies sind die Accessoren
        set { breite = value; }
    }
}

public class MyClass : Basisklasse   // MyClass erbt die Eigenschaften und
{                                    //  Methoden von Basisklasse
    private int hoehe;

    public int Hoehe             // Accessoren
    {
        get { return hoehe; }
        set { hoehe = value; }
    }

    public MyClass(): base()   // der Konstruktor ruft automatisch zuerst
    {                          //  den Konstruktor der Basisklasse auf
        hoehe = 13;
    }

    ~MyClass()    // optionaler Destruktor
    {
    }
}

public partial class Form1 : Form
{
    public Form1()
    {
        InitializeComponent();
    }
```

```
    private void button1_Click(object sender, EventArgs e)
    {
        MyClass mc = new MyClass();
        System.Console.WriteLine(mc.Breite);
        System.Console.WriteLine(mc.Hoehe);
    }
}
```

3.3 GUI-Anwendung ohne den Forms-Designer entwickeln

siehe Buch "Visual C# 2010" Seite 368

-- In Visual Studio eine neue Konsolen-Anwendung erzeugen

-- Rechtsklick auf den Projektnamen, das ist die Zeile unter der Projektmappe,

Eigenschaften: Ausgabetyp auf "Windows-Anwendung" setzen

-- Projekt --> Verweis hinzufügen

System.Windows.Forms und System.Drawing

```
using System;
using System.Windows.Forms;
using System.Drawing;


namespace Test_Das_erste_Fenster
{
    public static class Program
    {
        static void Main(string[] args)
        {
            Form1 fenster = new Form1();
            Application.Run(fenster);
        }
    }

    public class Form1 : Form
    {
        private Label label1;
        private Button button1;
        private void initializeComponents()
        {
            this.label1 = new Label();
            this.button1 = new Button();
```

```
            // Eigenschaften label1
            this.label1.Location = new Point(46, 40);
            this.label1.Size = new Size(110, 13);
            this.label1.Text = "Hallo, klick mich >>>";
            // Eigenschaften button1
            this.button1.Location = new Point(166, 35);
            this.button1.Size = new Size(75, 23);
            this.button1.Text = "Beenden";
            this.button1.Click += new EventHandler(this.button1_Click);
            // Eigenschaften Form1
            this.ClientSize = new Size(300, 100);
            this.Controls.Add(this.button1);
            this.Controls.Add(this.label1);
            this.Text = "Das erste Fenster";
        }

        public Form1()
        {
            initializeComponents();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            this.Close();
        }
    }
}
```

3.4 Überladene Operatoren

```
public class Messung
{
    private double w1, w2, w3;

    public Messung()
    {
        this.w1 = 0;
        this.w2 = 0;
        this.w3 = 0;
    }

    public Messung(double w1, double w2, double w3)
    {
        this.w1 = w1;
        this.w2 = w2;
        this.w3 = w3;
    }

    public static Messung operator +(Messung m1, Messung m2)
    {
        Messung ergebnis = new Messung(m1.w1 + m2.w1, m1.w2 + m2.w2, m1.w3 + m2.w3);
        return ergebnis;
    }

    public static Messung operator *(Messung m, double c)
    {
        return new Messung(m.w1 * c, m.w2 * c, m.w3 * c);
    }

    public static Messung operator *(double c, Messung m)
    {
        return m * c;
    }

    public void Print()
    {
        System.Console.Out.WriteLine(this.w1 + "  " + this.w2 + "  " + this.w3);
    }
```

```
}

private void button1_Click(object sender, EventArgs e)
{
    Messung m1 = new Messung(1, 2, 3);
    Messung m2 = new Messung(4, 5, 6);
    Messung m3 = new Messung();
    m1.Print();
    m2.Print();
    m3.Print();
    (m1 + m2).Print();
    (m1 * 2).Print();
    (2 * m1).Print();
}
```

3.5 Interfaces

```
namespace Test_Interface
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            Darstellbar[] zeichenelemente = new Darstellbar[10];
            // Der Sinn des Interface besteht hier darin, dass jetzt
            // unterschiedliche Typen (Uhrzeiger und Zifferblatt)
            // in einem gemeinsamen Array vom Typ "Darstellbar"
            // vereint werden können!
            zeichenelemente[0] = new Uhrzeiger();
            zeichenelemente[1] = new Zifferblatt();
            zeichenelemente[0].zeichnen();
            zeichenelemente[1].zeichnen();
        }
    }

    class Uhrzeiger : Darstellbar    // Die Klasse erbt das Interface "Darstellbar"
    {
        public Uhrzeiger()     // Konstruktor
        {
            Console.WriteLine("zeiger.Konstruktor");
        }

        public void zeichnen()    // Die abstrakte Methode _muss_ überschrieben werden
        {
            Console.WriteLine("zeiger.zeichnen");
        }
    }

    class Zifferblatt : Darstellbar    // Die Klasse erbt das Interface "Darstellbar"
    {
```

```
        public Zifferblatt()     // Konstruktor
        {
            Console.WriteLine("zifferblatt.Konstruktor");
        }

        public void zeichnen()    // Die abstrakte Methode _muss_ überschrieben werden
        {
            Console.WriteLine("zifferblatt.zeichnen");
        }
    }

    public interface Darstellbar
    {
        void zeichnen(); // im Interface wird eine abstrakte Methode definiert
                         // hier können zusätzlich auch konkrete Methoden drinstehen
    }
}
```

3.6 Generische Datentypen

Beispiel:

```
namespace nr9
{
    class Program
    {
        static void Main(string[] args)
        {
            // generische Datentypen
            // Rechenregeln für integer: x = a + b;
            // Rechenregeln für double: x = (a + b) * 2;

            Addierer<int> i = new Addierer<int>();
            Addierer<double> d = new Addierer<double>();

            Console.WriteLine(i.Summe(1, 1));
            Console.WriteLine(d.Summe(1, 1));

            Console.ReadLine();
        }
    }

    public class Addierer<T>
    {
        public T Summe(T wert1, T wert2)
        {
            if (wert1 is int)
            {
                dynamic w1 = wert1;
                dynamic w2 = wert1;
                return (w1 + w2);
            }
            else if (wert1 is double)
            {
                dynamic w1 = wert1;
                dynamic w2 = wert1;
                return (w1 + w2) * 2.0;
            }
            else
            {
                dynamic w = 0;
                return w;
            }
        }
    }
}
```

3.7 Delegate

Beispiel:

```
using System.IO;

namespace nr10
{
    class Program
    {
        private static String formatmeldung(String meldung)
        {
            return DateTime.Now.ToString("yyyy-MM-dd HH:mm:ss ") + meldung;
        }

        public delegate void Protokolleintrag(String meldung);

        static public void ProtokolleintragInDatei(String meldung)
        {
            try
            {
                using (StreamWriter f = File.AppendText(@"D:\BMWH89BU\log.txt"))
                {
                    f.WriteLine(formatmeldung(meldung));
                }
            }
            catch
            {
                // irgendwie anders sterben
            }
        }

        public static void ProtokolleintragViaUDP(String meldung)
        {
            // hier irgendwas sinnvolles tun
        }

        static void Main(string[] args)
        {
            // Beispiel für delegate     Delegate sind Funktionszeiger

            Protokolleintrag log = ProtokolleintragInDatei;
            log("lala");

            log = ProtokolleintragViaUDP;
            log("fufu");
        }
    }
}
```

3.8 Polymorphie: abstract, virtual, overide

siehe Dirk Frischalowski: Visual C# 2010, ab Seite 164

- **virtual: Eine Methode in der Basisklasse wird mit virtual gekennzeichnet, wenn sie in abgeleiteten Klassen überschrieben werden soll. Virtuelle Methoden haben u.U. einen hohen Speicherverbrauch wegen großer Sprungtablellen**
- **override: Eine Methode in einer abgeleiteten Klasse wird mit override gekennzeichnet, wenn sie die Methode der Basisklasse überschreiben soll.**
- **abstract: Eine Methode die als abstract deklariert ist, muss in abgeleiteten Klassen überschrieben werden.**

Beispiel für virtual und override:

```
using System;

namespace Schnecken
{
    class Schnecke
    {
        String name;
        int alter;

        public Schnecke(String name, int alter)
        {
            this.name = name;
            this.alter = alter;
        }

        public virtual void info()    // Diese Methode kann überschrieben werden
        {
            Console.WriteLine("Schnecke {0}: {1} Tage alt", name, alter);
        }
    }

    class Lastschnecke : Schnecke
    {
        double traglast;
```

```
        public Lastschnecke(String name, int alter, double traglast)
            : base(name, alter)
        {
            this.traglast = traglast;
        }

        public override void info()    // Diese Methode überschreibt die Methode der Basisklasse
        {
            base.info();   // Methode der Basisklasse aufrufen
            Console.WriteLine(" (Lastschnecke mit {0} g Traglast)", traglast);
        }
    }

    class Program
    {
        static void Main(string[] args)
        {
            Schnecke susi = new Schnecke("Susi", 5);
            Lastschnecke hubert = new Lastschnecke("Hubert", 3, 27);
            susi.info();
            hubert.info();
            Console.ReadLine();
        }
    }
}
```

3.9 Indexer

Siehe auch Dirk Frischalowski: Visual C#2010, Seite 228

Ein Indexer ermöglicht es, eine Klasse u.a. auch wie ein Array zu nutzen.

```
using System;
using System.Windows.Forms;

namespace test
{
    public partial class Form1 : Form
    {
        public class MeineKlasse
        {
            private int[] a;

            public int this[int index]     // Indexer
            {
                get { return a[index]; }
                set { a[index] = value; }
            }

            public MeineKlasse()   // Konstruktor
            {
                a = new int[10];
            }
        }

        public Form1()
        {
            InitializeComponent();
            MeineKlasse instanz1 = new MeineKlasse();
            MeineKlasse instanz2 = new MeineKlasse();
            instanz1[3] = 33;
            instanz1[4] = 44;
            richTextBox1.AppendText(instanz1[3].ToString() + " " + instanz1[4].ToString() + "\n");
        }
    }
}
```

4 Datei Ein- und Ausgabe

4.1 Textdateien einlesen

Beispiel 1:

```
string[] lines;
string[] words;

string[] lines = File.ReadAllLines("text.dat");
string[] coord = lines[1].Split(' ');

System.Globalization.NumberFormatInfo info = new System.Globalization.NumberFormatInfo();
info.NumberDecimalSeparator = ".";
info.NumberGroupSeparator = ",";
words = lines[0].Split(new char[] { ' ' });
d = System.Convert.ToDouble(words[0],info);
words = lines[0].Split(' ');
words = lines[i + 16].Split(new char[] { ' ' }, StringSplitOptions.RemoveEmptyEntries);
```

Beispiel 2:

```
int a = 0;
openFileDialog1.ShowDialog();
openFileDialog2.ShowDialog();

using(TextWriter outp = File.CreateText(openFileDialog2.FileName))
{
   using(TextReader inp = File.OpenText(openFileDialog1.FileName))
   {
       string line;
       using(TextWriter header = File.CreateText("Header.txt"))
       {
           for (a = 0; a < 69; a++)    // Header einlesen
           {
               line = inp.ReadLine();
               header.WriteLine(line);
           }
```

```
        }
        while ((line = inp.ReadLine()) != null)
        {
            string[] split = line.Split(new char[]{' '},StringSplitOptions.RemoveEmptyEntries);
            try
            {
                outp.Write(split[2] + " ");
            }
            catch{};
        }
    }
}
```

Beispiel 3:

```
using System.IO;

String s = File.ReadAllText(@"D:\info.txt");
String[] zeilen = s.Split('\n');
Console.WriteLine(s);

Console.WriteLine(zeilen.Length);
foreach (String z in zeilen)
    Console.WriteLine(z);

File.WriteAllText("out.txt",s);

using (StreamReader sr = new StreamReader(@"D:\info.txt"))
{
    String line;
    while ((line = sr.ReadLine()) != null)
    {
        Console.WriteLine(line);
    }
}
```

Beispiel 4: Zahlen aus einer Datei einlesen und aufsummieren, fehlerhafte Zeilen zählen, Ergebnis in eine andere Datei schreiben.

```
using System.IO;

double summe = 0;
int fehler = 0;

using (StreamReader sr = new StreamReader(@"D:\BMWH89BU\rechnung.txt"))
{
    String line;
    while ((line = sr.ReadLine()) != null)
    {
        try
        {
            summe += Convert.ToDouble(line);
        }
        catch
        {
            fehler++;
        }
    }
}

StreamWriter sw = new StreamWriter(@"D:\BMWH89BU\out.txt");
sw.WriteLine("Summe: {0:F2}", summe);
sw.WriteLine("{0} Fehler", fehler);
sw.Close();
```

Die Verwendung von "using" hätte hier den Vorteil, dass das Objekt StreamWriter sw nach Abschluss des Codeblocks nicht mehr existieren würde.

Beispiel 5: Webseite einlesen

```
using System.Net;

using (WebClient wc = new WebClient())
{
    String website = wc.DownloadString(@"http://www.spiegel.de");
    Console.WriteLine(website);
}
```

Beispiel 6: Webseite zeilenweise einlesen

```
using System.IO;
using System.Net;

WebClient wc = new WebClient();
using (Stream st = wc.OpenRead(@"https://www.heise.de/newsticker/"))
{
    using (StreamReader sr = new StreamReader(st))
    {
        String zeile;
        while ((zeile = sr.ReadLine()) != null)
        {
             Console.Write("--");
             Console.WriteLine(zeile);
        }
    }
}
```

4.2 Binärdatei erzeugen

```
FileStream fs = new FileStream("curves.acv", FileMode.OpenOrCreate);
BinaryWriter bw = new BinaryWriter(fs);
bw.Write((Byte)0);
bw.Write((Byte)4);
fs.Close();
```

4.3 Binärdatei einlesen

```
using System.IO;

BinaryReader binReader = new BinaryReader(File.Open(path, FileMode.Open));
byte[] buf = new byte[5000];
```

```
int header_size;
int intens_org_x;

binReader.BaseStream.Position = 6;
binReader.Read(buf, 0, 4);
Array.Reverse(buf, 0, 4);
header_size = BitConverter.ToInt32(buf, 0);

binReader.BaseStream.Position = 48;
intens_org_x = 256 * binReader.ReadByte();
intens_org_x += binReader.ReadByte();

binReader.Close();
```

4.4 ASCII-PGM (P2 Portable Gray Map) Datei einlesen

```
using (TextReader elev = File.OpenText("elevation.pgm"))
{
    string line = elev.ReadLine();   // P2
    line = elev.ReadLine();          // # Created by GIMP version 2.10.8 PNM plug-in
    line = elev.ReadLine();          // 264 131 width and height
    string[] width_height = line.Split(new char[] { ' ' });
    elev_width = System.Convert.ToInt32(width_height[0]);   // 264
    elev_height = System.Convert.ToInt32(width_height[1]);  // 131
    line = elev.ReadLine();          // 65536 max value
    for (int y = 0; y < elev_height; y++)
    {
        for (int x = 0; x < elev_width; x++)
        {
            elevation[x, y] = System.Convert.ToInt32(elev.ReadLine());
        }
    }
}
```

4.5 Prüfen, ob ein Verzeichnis oder eine Datei existiert

```
string path_name = "mein_Pfad";
DirectoryInfo di1 = new DirectoryInfo(path_name);
if (di1.Exists == false)
    di1.Create();            // Verzeichnis erzeugen

string file_name = "meine_Datei.txt";
FileInfo fi1 = new FileInfo(file_name);
if (fi.Exists == false)       // if the file doesn't already exist, load it from the internet
    wc.DownloadFile("https://a.tile.opentopomap.org/" + file_name, path_name + "/" + file_name);
```

5 Konsolen-Programme

5.1 Konsolen-Programm zur Anzeige der Übergabe-Parameter

```
using System;
using System.IO;

namespace Show_Parameters
{
    public class Program
    {
        static void Main(string[] args)
        {
            if (args.Length == 0)
                 Console.WriteLine("No parameters were passed");
            else
            {
                Console.WriteLine("These parameters were passed:");
                foreach (string s in args)
                    Console.WriteLine(s);
            }
            Console.Read();
        }
    }
}
```

5.2 Neues Konsolen-Projekt anlegen

Projekttyp: Konsolen-App (.NET Framework) erzeugt eine *.exe

Projekttyp: Konsolen-App (.NET Core) erzeugt eine *.dll die mit dem Konsolenbefehl dotnet *.dll gestartet werden kann.

5.3 Konsole Ein- und Ausgabe

```
Console.WriteLine("text {1} {0}", i1, i2);
// in diesem Fall wird bei {1} i2 ausgegeben und bei {0} i1.

Console.Write("Eingabe: ");
String s = Console.ReadLine();
```

6 Grafik

6.1 Bild in Panel, Autoscroll zentrieren

```
using System;
using System.Drawing;
using System.Windows.Forms;

namespace test2
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            pictureBox1.Location = new Point(0, 0);
            panel1.AutoScroll = true;
            pictureBox1.SizeMode = PictureBoxSizeMode.Normal;
            pictureBox1.Size = new Size(1280, 1024);

            Show();         //  Show(); ist notwendig weil sonst der nachfolgende Aufruf von center() nicht die gewünschte Wirkung hätte.
            center();
        }

        public void center()
        {
            pictureBox1.Load("test.png");
            Point p = new Point();
            p.X = (pictureBox1.Size.Width - panel1.ClientRectangle.Width) / 2;
            p.Y = (pictureBox1.Size.Height - panel1.ClientRectangle.Height) / 2;
            panel1.AutoScrollPosition = p;
        }

        private void button1_Click(object sender, EventArgs e)
        {
            center();
        }
    }
}
```

Panel.AutoScrollPosition hat eine sehr merkwürdige Eigenschaft, die einen in den Wahnsinn treiben kann wenn man sie nicht kennt:

- **Beim Lese-Zugriff wird die aktuelle Position mit negativem Vorzeichen zurückgegeben.**

6.2 Farbpalette aus Datei lesen

```
farbpalette = new Color[256];
int i;
spektrum = new String[256];
zeile = new String[5];
spektrum = File.ReadAllLines("spektrum.txt", System.Text.Encoding.ASCII);
for (i = 0; i < 256; i++){
    zeile = spektrum[i].Split(',');
    farbpalette[255-i] = Color.FromArgb(Int32.Parse(zeile[0]),Int32.Parse(zeile[1]),Int32.Parse(zeile[2]));
}
```

6.3 Screenshot von Form1

```
Bitmap b1 = new Bitmap(this.Width,this.Height);
this.DrawToBitmap(b1, new Rectangle(0, 0, this.Width, this.Height));
b1.Save("Screenshot.png");
```

Mögliche Probleme:

a) Inhalt von RichTextBox

b) Inhalt von PictureBox, falls das Bild mit dispose() schon wieder freigegeben wurde

Screenshot von PictureBox mit zusätzlichem Untertitel:

```
Bitmap b1 = new Bitmap(pictureBox1.Width, pictureBox1.Height + 20);
pictureBox1.DrawToBitmap(b1, new Rectangle(0, 0, pictureBox1.Width, pictureBox1.Height));
Graphics g1 = Graphics.FromImage(b1);
g1.DrawString("test",new Font("Arial", 16), Brushes.White, 0,pictureBox1.Height);
oder
g1.DrawString("Test", new Font("Arial", 16), new SolidBrush(Color.White), 0, pictureBox1.Height);
```

6.4 Exif Daten lesen

```
int exp1 = BitConverter.ToInt32(image.GetPropertyItem(33434).Value, 0);
int exp2 = BitConverter.ToInt32(image.GetPropertyItem(33434).Value, 4);
int blende1 = BitConverter.ToInt32(image.GetPropertyItem(33437).Value, 0);
int blende2 = BitConverter.ToInt32(image.GetPropertyItem(33437).Value, 4);
int exposure_program = BitConverter.ToInt16(image.GetPropertyItem(34850).Value, 0);
String exp_pro;
switch (exposure_program)
{
    case 1:
        exp_pro = "Manual";
        break;
    case 2:
        exp_pro = "Normal program";
        break;
    case 3:
        exp_pro = "Aperture priority";
        break;
    case 4:
        exp_pro = "Shutter priority";
        break;
    case 5:
        exp_pro = "Creative program";
        break;
    case 6:
        exp_pro = "Action program";
        break;
    case 7:
        exp_pro = "Landscape mode";
        break;
    default:
        exp_pro = "Not defined";
        break;
}

int iso = BitConverter.ToInt16(image.GetPropertyItem(34855).Value, 0);
String date_time = Encoding.ASCII.GetString(image.GetPropertyItem(36867).Value);

richTextBox1.AppendText(exp1.ToString() + "/" + exp2.ToString() + "\r");
richTextBox1.AppendText("F/" + (blende1/blende2).ToString() + "\r");
```

```
richTextBox1.AppendText(exp_pro + "\r");
richTextBox1.AppendText("ISO" + iso.ToString() + "\r");
richTextBox1.AppendText(date_time + "\r");

String[,] a = new String[20, 20];
a[0, 1] = exp1.ToString() + "/" + exp2.ToString();  // Belichtungszeit
a[0, 2] = "F/" + (blende1 / blende2).ToString();  // Blende
a[0, 3] = iso.ToString();
```

6.5 Bild erzeugen

```
int breite = 1000;
int hoehe = 750;
int teilung = 10;
int i;

Bitmap b1 = new Bitmap(breite,hoehe);
Graphics g1 = Graphics.FromImage(b1);
g1.Clear(Color.White);
Pen p1 = new Pen(Color.Black);
g1.DrawLine(p1, 0, 0, breite, hoehe);
g1.DrawLine(p1, breite, 0, 0, hoehe);

for (i = 0; i <= teilung; i++)
{
    g1.DrawLine(p1, breite / teilung * i - 1, 0, breite / teilung * i - 1, hoehe - 1);
    g1.DrawLine(p1, breite / teilung * i    , 0, breite / teilung * i    , hoehe - 1);
    g1.DrawLine(p1, 0, hoehe / teilung * i - 1, breite - 1, hoehe / teilung * i - 1);
    g1.DrawLine(p1, 0, hoehe / teilung * i    , breite - 1, hoehe / teilung * i    );
}
b1.Save("Test_1000_750.png");
```

6.6 Transparenz eines Bildes einstellen

```
private Bitmap bmp;

public Form1()
{
    InitializeComponent();
    SetStyle(ControlStyles.UserPaint, true);
    SetStyle(ControlStyles.AllPaintingInWmPaint, true);
    SetStyle(ControlStyles.OptimizedDoubleBuffer, true);
    bmp = new Bitmap("C:\\Users\\mkoch\\Pictures\\M31_klein.jpg");
}

private void trackBar1_Scroll(object sender, EventArgs e)
{
    pictureBox1.Invalidate();
}

private void pictureBox1_Paint(object sender, PaintEventArgs e)
{
    ImageAttributes iattr = new ImageAttributes();
    ColorMatrix m = new ColorMatrix();
    m.Matrix33 = trackBar1.Value / 100f;
    iattr.SetColorMatrix(m);
    e.Graphics.DrawImage(bmp, new Rectangle(0, 0, bmp.Width, bmp.Height), 0, 0, bmp.Width, bmp.Height, GraphicsUnit.Pixel, iattr);
}
```

6.7 SetPixel(), DrawLine() und FillRectangle()

```
public Bitmap b1 = new Bitmap(256,256);

public Form1()
{
    InitializeComponent();
    Graphics g1 = Graphics.FromImage(b1);
    g1.Clear(Color.White);
    b1.SetPixel(100,100,Color.Red);
    pictureBox1.Refresh();   // bewirkt dass pictureBox_Paint aufgerufen wird
}

private void pictureBox1_Paint(object sender, PaintEventArgs e);
{
    Graphics g = e.Graphics;
    g.DrawImage(b1,0,0);
    Pen p = new Pen(Color.Black);
    g.DrawLine(p,0,0,50,50);
    SolidBrush brush = new SolidBrush(Color.DarkGray);
    g.FillRectangle(brush, 16 * x, 16 * y, 16, 16);
}
```

Eine andere Methode:

```
Brush aBrush = (Brush)Brushes.Black;
Graphics g = this.CreateGraphics();
g.FillRectangle(aBrush, x, y, 1, 1);
```

Es ist wichtig, dass man innerhalb der pictureBox_Paint Routine das Graphics Objekt so holt, wie oben gezeigt, also mit

Graphics g = e.Graphics;

und nicht mit

Graphics g = Graphics.FraomHwnd(pictureBox1.handle);

6.8 FillPolygon()

```
using System;
using System.Drawing;
using System.Windows.Forms;

namespace FillPolygon
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            Bitmap b1 = new Bitmap(pictureBox1.Width, pictureBox1.Height);
            Graphics g1 = Graphics.FromImage(b1);
            pictureBox1.Image = b1;

            SolidBrush blueBrush = new SolidBrush(Color.Blue);

            Point point1 = new Point(50, 50);
            Point point2 = new Point(100, 25);
            Point point3 = new Point(200, 5);
            Point point4 = new Point(250, 50);
            Point point5 = new Point(220, 100);
            Point point6 = new Point(180, 200);
            Point point7 = new Point(140, 250);
            Point[] curvePoints = { point1, point2, point3, point4, point5, point6, point7 };

            g1.FillPolygon(blueBrush, curvePoints);
        }
    }
}
```

6.9 Das Graphics Objekt

Ein Graphics Objekt kann so erzeugt werden:

-- aus einer Bitmap: g1 = Graphics.FromImage(bitmap1);

-- von irgendeinem Windows Form Objekt: `Graphics g1 = Graphics.FromHwnd(panel1.Handle);`

-- oder mittels Übergabeparameter:

```
private void pictureBox1_Paint(object sender, PaintEventArgs e);
{
    Graphics g = e.Graphics;
    Pen p = new Pen(Color.Black);
    g.DrawLine(p,0,0,50,50);
}
```

Es stellt u.a. diese Methoden zur Verfügung:

-- g1.Clear(Color.White);

-- g1.DrawImage(bitmap1,50,50);

-- g1.DrawLine(pen1, 0, 0, 99, 99);

-- g1.DrawString(s.ToString(), font1, new SolidBrush(Color.Gray), 20, 30);

-- g1.FillRectangle(brush1, x, y, 16, 16);

6.10 Das Bitmap Objekt

Ein Bitmap Objekt kann so erzeugt werden:

-- Bitmap b1 = new Bitmap(breite,hoehe);

-- Bitmap b1 = new Bitmap("IMG_4266.JPG");

-- Bitmap b1 = new Bitmap(this.Width,this.Height); (Größe dieses Formulars)

Es stellt u.a. diese Methoden zur Verfügung:

-- b1.Save("Test.png");

-- b1.Save("Test.png", System.Drawing.Imaging.ImageFormat.Png);

-- b1.SetPixel(100,100,Color.Red);

6.11 Beispiel

```
Bitmap b1 = new Bitmap(breite,hoehe);
Graphics g1 = Graphics.FromImage(b1);
g1.Clear(Color.White);
Pen p1 = new Pen(Color.Black);
p1.Width = 4;
g1.DrawLine(p1, 0, 0, 99, 99);
b1.Save("Test.png");
// b1.Save("Test.bmp, System.Drawing.Imaging.ImageFormat.Bmp);

Font font1 = new Font("Arial",9,FontStyle.Bold);
g.DrawString(s.ToString(), font1, new SolidBrush(Color.Gray), 20, 30);

g.DrawImage(bitmap1,50,50);

button1.BackColor = System.Drawing.SystemColors.Control;
button1.BackColor = Color.Tomato;
```

```
private void ApplyGain(Bitmap b, double gain)
{
    Rectangle rect = new Rectangle(0, 0, b.Width, b.Height);   // Lock the bitmap's bits
    System.Drawing.Imaging.BitmapData bmpData =
        b.LockBits(rect, System.Drawing.Imaging.ImageLockMode.ReadWrite, b.PixelFormat);

    IntPtr ptr = bmpData.Scan0;   // Get the address of the first line
    int bytes = Math.Abs(bmpData.Stride) * b.Height;   // Declare an array to hold the bytes of the bitmap
    byte[] rgbValues = new byte[bytes];
    System.Runtime.InteropServices.Marshal.Copy(ptr, rgbValues, 0, bytes); // Copy the RGB values into the array

    double d;
    for (int counter = 0; counter < rgbValues.Length; counter++)
    {
        d = gain * rgbValues[counter];
        if (d > 255) d = 255;
        rgbValues[counter] = System.Convert.ToByte(d);
    }
    System.Runtime.InteropServices.Marshal.Copy(rgbValues, 0, ptr, bytes);  // Copy the RGB values back to the bitmap
    b.UnlockBits(bmpData);   // Unlock the bits
}

Bitmap image = new Bitmap("IMG_4266.JPG");
pictureBox1.SizeMode = PictureBoxSizeMode.Zoom;
image.SetPixel(x, y, Color.Gray);
pictureBox1.Image = image;
pictureBox1.Refresh();

brightness = image.GetPixel(x,y).GetBrightness();


ApplyGain(image,1.5);
pictureBox1.Image = image;   // Draw the modified image

image.Dispose();   // wenn man's nicht mehr braucht
```

7 Windows Forms

Ist veraltet und wird durch WPF ersetzt!

7.1 Parameterübergabe an eine Windows Forms Anwendung

Es gibt zwei Möglichkeiten wie man das machen kann. Die einfachere Möglichkeit ist der Zugriff über Environment.GetCommandLineArgs()

```
private void Form1_Shown(object sender, EventArgs e)
{
    string filename;

    string[] arguments = Environment.GetCommandLineArgs();
    if (arguments.Length == 2)
        filename = arguments[1];
}
```

Die etwas kompliziertere Möglichkeit geht wie folgt:

Program.cs

```
using System;
using System.Windows.Forms;

namespace test
{
    static class Program
    {
        /// <summary>
        /// Der Haupteinstiegspunkt für die Anwendung.
        /// </summary>
        [STAThread]
        static void Main(string[] args)            // Diese Zeile muss modifiziert werden
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Form1(args));      // Diese Zeile muss modifiziert werden
        }
    }
}
```

Form1.cs

```
using System;
using System.Windows.Forms;

namespace test
{
    public partial class Form1 : Form
    {
        string[] arguments;            // globale Variable

        public Form1(string[] args)    // Diese Zeile muss modifiziert werden
        {
            InitializeComponent();
            arguments = args;          // Argumente in die globale Variable übertragen
        }

        private void Form1_Shown(object sender, EventArgs e)
        {
            foreach (string s in arguments)
                richTextBox1.AppendText(s + "\n");
        }
    }
}
```

oder alternativ so:

```
string[] arguments;

public Form1(string[] arguments)    // Diese Zeile muss modifiziert werden
{
    InitializeComponent();
    this.arguments = arguments;     // Argumente in die globale Variable übertragen
}
```

7.2 InitializeComponent()

Vor "InitializeComponent();" sollte man keinen Code schreiben, insbesondere keine Klasse. Das kann dazu führen, dass die Designer-Ansicht nicht mehr funktioniert.

Nach "InitializeComponent();" kann Code eingefügt werden, der nach dem Start des Programms einmalig ausgeführt werden soll. Falls auf irgendwelche Windows Forms Komponenten zugegriffen werden soll, kann es erforderlich sein vorher noch Show() aufzurufen.

Alternativ kann man auch die Ereignisbehandlungsroutine Form1_Shown() verwenden.

7.3 Form beim Programmstart maximieren

Das muss in Form1.Shown gemacht werden. Wenn man es direkt hinter InitializeComponent() machen würde, dann würde this.Size.Width eine falsche (zu kleine) Breite ergeben. Die Form wird erst dann maximiert, wenn Form1_Shown() aufgerufen wird.

```
private void Form1_Shown(object sender, EventArgs e)
{
    this.WindowState = FormWindowState.Maximized;
    size_x = this.Size.Width;
}
```

7.4 RichTextBox mit farbigen Texten

```
private void AddText(RichTextBox rtb, string txt, Color col)
{
    rtb.SelectionColor = col;
    rtb.SelectionBackColor = Color.Yellow;
    rtb.SelectedText = txt;
    rtb.SelectionColor = rtb.ForeColor;
    rtb.SelectionBackColor = rtb.BackColor;
}
```

In der RichTextBox zum Ende des Textes scrollen:

```
richTextBox1.ScrollToCaret();
```

7.5 RichTextBox mit Exponenten

```
richTextBox1.SelectedRtf = @"{\rtf1\ansi This is an exponent\super 2\nosupersub . This is normal text.}";
richTextBox1.SelectedText = "\n";
richTextBox1.SelectedRtf = "{\\rtf1\\ansi This is an exponent\\super 2\\nosupersub . This is normal text.}";
richTextBox1.SelectedText = "\n";
int a = 2;
richTextBox1.SelectedRtf = "{\\rtf1\\ansi This is an exponent\\super " + a.ToString() + "\\nosupersub . This is normal text.}";
richTextBox1.SelectedText = "\n";
```

Liste mit allen RTF Steuer-Worten:

https://learn.microsoft.com/de-de/previous-versions/office/developer/office2000/aa140302(v=office.10)

7.6 RichTextBox mit griechischen Buchstaben

Lowercase Greek letters		Uppercase Greek letters	
Greek letter	RTF code	Greek letter	RTF code
alpha	\u0945?	ALPHA	\u0913?
beta	\u0946?	BETA	\u0914?
gamma	\u0947?	GAMMA	\u0915?
delta	\u0948?	DELTA	\u0916?
epsilon	\u0949?	EPSILON	\u0917?
zeta	\u0950?	ZETA	\u0918?
eta	\u0951?	ETA	\u0919?
theta	\u0952?	THETA	\u0920?
iota	\u0953?	IOTA	\u0921?
kappa	\u0954?	KAPPA	\u0922?
lambda	\u0955?	LAMBDA	\u0923?
mu	\u0956?	MU	\u0924?
nu	\u0957?	NU	\u0925?
xi	\u0958?	XI	\u0926?
omicron	\u0959?	OMICRON	\u0927?
pi	\u0960?	PI	\u0928?
rho	\u0961?	RHO	\u0929?
sigma	\u0963?	SIGMA	\u0931?
tau	\u0964?	TAU	\u0932?
upsilon	\u0965?	UPSILON	\u0933?
phi	\u0966?	PHI	\u0934?
chi	\u0967?	CHI	\u0935?
psi	\u0968?	PSI	\u0936?
omega	\u0969?	OMEGA	\u0937?

Multiplikations-Symbol: \u8901?

7.7 TableLayoutPanel in einem Panel, für Differenz-Darstellung

Panel: AutoScroll = true

TableLayoutPanel: Dock = left

Alle Coulumns auf feste Pixel-Breite setzen

Beide Rows auf 50% Höhe setzen

PictureBox: SizeMode = zoom

Dock = fill

ColumnSpan = 2

Image = ... Bild auswählen

Man kann alternativ auch das Panel weglassen, aber das hat den Nachteil dass dann zur Laufzeit auch ein vertikaler Scrollbalken erscheint. In diesem Fall muss man setzen:

TableLayoutPanel: Dock = none

AutoScroll = true

Zur Laufzeit die Breite einer Spalte des TableLayoutPanels verändern:

```
tableLayoutPanel1.ColumnStyles[X].Width = 100;
```

Zur Laufzeit die Breite aller Spalten an die Höhe des TableLayoutPanels anpassen:

```
private void tableLayoutPanel1_SizeChanged(object sender, EventArgs e)
{
    float h = tableLayoutPanel1.Height;
    for (int i = 0; i < tableLayoutPanel1.ColumnCount; i++)
```

```
        tableLayoutPanel1.ColumnStyles[i].Width = h * 0.35F;
}
```

7.8 Listview

```
listView1.Clear();
listView1.GridLines = true;
listView1.Columns.Add("Bildnummer", 100, HorizontalAlignment.Right);
listView1.Columns.Add("Aufnahmezeitpunkt", 100, HorizontalAlignment.Right);
```

7.9 Ordner auswählen

```
if (folderBrowserDialog1.ShowDialog() == DialogResult.OK)
    textBox1.Text = folderBrowserDialog1.SelectedPath;

folderBrowserDialog1.RootFolder = Environment.SpecialFolder.MyPictures;
folderBrowserDialog1.ShowDialog();
DirectoryInfo myDir = new DirectoryInfo(folderBrowserDialog1.SelectedPath);
FileInfo[] myFiles = myDir.GetFiles("*.jpg");
Array.Sort<FileSystemInfo>(myFiles, delegate(FileSystemInfo a, FileSystemInfo b)  // Sortieren
{
    // return a.CreationTimeUtc.CompareTo(b.CreationTimeUtc);
    return a.Name.CompareTo(b.Name);
});
```

7.10 Datei auswählen

```
openFileDialog1.InitialDirectory = textBox1.Text;
if (openFileDialog1.ShowDialog() == DialogResult.OK)
    textBox2.Text = Path.GetFileName(openFileDialog1.FileName);
```

7.11 Folderbrowser Dialog

```
folderBrowserDialog1.ShowDialog();
fileSystemWatcher1.Path = folderBrowserDialog1.SelectedPath;
textBox2.Text = folderBrowserDialog1.SelectedPath;
```

7.12 Filesystemwatcher

Problem: Das Event fileSystemWatcher_Created kommt sofort, wenn eine neue Datei erzeugt wird. Es ist möglich, dass man zu diesem Zeitpunkt noch nicht auf die Datei zugreifen kann, weil sich noch nicht fertig geschrieben wurde. Insbesondere dann, wenn die Datei über ein Netzwerk geladen wird.

Lösung:

```
private static bool CanWriteFile(string fileName)
{
    try
    {
        using (FileStream fs = new FileStream(fileName, FileMode.Open))
        {
            return fs.CanWrite;
        }
    }
    catch
    {
        return false;
    }
}

private void fileSystemWatcher1_Created(object sender, System.IO.FileSystemEventArgs e)
{
    for ( ; CanWriteFile(e.FullPath) == false ; )
        System.Threading.Thread.Sleep(1);           // Delay
    BinaryReader binReader = new BinaryReader(File.Open(e.FullPath, FileMode.Open));
}
```

7.13 Panel / PictureBox

Ein Bild mit Scrollbalken anzeigen:

Panel: AutoScroll = true;

darin liegt eine PictureBox: Dock = None

Image = ...

Location = 0;0

SizeMode = Normal

Scrollbalken per Code setzen:

```
panel1.AutoScrollPosition = new Point(100, 100);  // das geht
panel1.AutoScrollPosition.X = 100;                // das geht nicht, weil der Datentyp
                                                  //  Point ein Wertetype (value type)
                                                  //  ist. Man muss also immer den
                                                  //  gesamten Datentyp als Variable
                                                  //  vorhalten (einlesen, dann Eigenschaft
                                                  //  verändern, neu zuweisen):
panel1.AutoScrollPosition = new Point(100, panel1.AutoScrollPosition.Y);  // so geht es
```

Scrollbalken auslesen:

```
int x = panel1.AutoScrollPosition.X;
```

7.14 MessageBox

```
MessageBox.Show("Hallo");
```

7.15 DataGridView

Mit Sortierung nach mehreren Spalten, indem man auf die Einträge in der Kopfzeile klickt:

```
using System;
using System.Windows.Forms;

namespace DataGridView
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();

            dataGridView1.ColumnCount = 3;
            dataGridView1.Columns[0].Name = "Available";
            dataGridView1.Columns[1].Name = "Radius";
            dataGridView1.Columns[2].Name = "Cavity";

            dataGridView1.Rows.Add(new string[] { "no", "150", "333" });
            dataGridView1.Rows.Add(new string[] { "yes", "240", "444" });
            dataGridView1.Rows.Add(new string[] { "yes", "330", "333" });
            dataGridView1.Rows.Add(new string[] { "yes", "150", "770" });
            dataGridView1.Rows.Add(new string[] { "no", "240", "555" });

            dataGridView1.Columns["Available"].SortMode = DataGridViewColumnSortMode.Programmatic;
            dataGridView1.Columns["Radius"].SortMode = DataGridViewColumnSortMode.Programmatic;
            dataGridView1.Columns["Cavity"].SortMode = DataGridViewColumnSortMode.Programmatic;

            dataGridView1.AllowUserToAddRows = false;

            // Alle Zeilen bis auf die erste löschen:
            // dataGridView1.RowCount = 1;

            // Eine Zelle markieren:
            // dataGridView1.CurrentCell = dataGridView1[1, 2];

            dataGridView1.SelectionMode = DataGridViewSelectionMode.FullRowSelect;
            // dataGridView1.ClearSelection();
```

```
        // In die Zelle in Spalte 1 und Zeile 2 schreiben:
        // dataGridView1[1, 2].Value = "12345";

        // Die Zelle in Spalte 1 und Zeile 2 lesen:
        // string s = dataGridView1[1, 2].Value.ToString();
        // richTextBox1.AppendText(s + "\n");

        // Sortiere nach Spalte 1 aufsteigend
        // dataGridView1.Sort(dataGridView1.Columns[1], ListSortDirection.Ascending);
    }

    private void button1_Click(object sender, EventArgs e)
    {
        dataGridView1.Sort(new RowComparer(1));
    }

    private class RowComparer : System.Collections.IComparer
    {
        private static int c = 0;

        public RowComparer(int column)
        {
            c = column;
        }

        public int Compare(object x, object y)
        {
            DataGridViewRow DataGridViewRow1 = (DataGridViewRow)x;
            DataGridViewRow DataGridViewRow2 = (DataGridViewRow)y;

            // Try to sort based on column 0
            int CompareResult = -1 * System.String.Compare(
                DataGridViewRow1.Cells[0].Value.ToString(),
                DataGridViewRow2.Cells[0].Value.ToString());

            // If the fields in column 0 are equal, sort based on columns 1 or 2
            if (CompareResult == 0)
            {
                CompareResult = System.String.Compare(
                    DataGridViewRow1.Cells[c].Value.ToString(),
                    DataGridViewRow2.Cells[c].Value.ToString());
```

```
                }
                return CompareResult;
            }
        }

        private void dataGridView1_ColumnHeaderMouseClick_1(object sender, DataGridViewCellMouseEventArgs e)
        {
            if ((e.ColumnIndex >= 1) && (e.ColumnIndex <= 2))   // Wenn in die Kopfzeile der Spalten 1 oder 2 geklickt wurde
            {
                dataGridView1.Sort(new RowComparer(e.ColumnIndex));   // Übergebe die Nummer der Spalte nach der sortiert werden soll
                for (int i = 0; i < dataGridView1.ColumnCount; i++)   // Lösche alle Sortier-Symbole in der Kopfzeile
                    dataGridView1.Columns[i].HeaderCell.SortGlyphDirection = SortOrder.None;
                dataGridView1.Columns[e.ColumnIndex].HeaderCell.SortGlyphDirection = SortOrder.Ascending;  // Setze Sortier-Symbol auf aufsteigend
            }
        }

    }
}
```

7.16 NumericUpDown

Man kann in einem NumericUpDown auch eine Einheit (z.B. %) mit anzeigen, indem man eine abgeleitete Klasse (hier: NumericUpDownUnit) erzeugt, die von der Klasse NumericUpDown erbt. In dieser neuen Klasse wird die Methode "UpdateEditText" überschrieben:

```
public class NumericUpDownUnit : NumericUpDown
{
    public NumericUpDownUnit()
    {
    }

    protected override void UpdateEditText()
    {
        this.Text = this.Value.ToString() + " %";
    }
}
```

Wichtig ist, dass man die neue Klasse entweder in eine neue Datei schreibt, oder wenn man sie in die Datei "Form1.cs" schreibt, dann muss sie unbedingt hinter der Klasse "Form1" stehen. Denn die Klasse "Form1" muss in der Datei "Form1.cs" immer die erste Klasse sein!

7.17 ComboBox

Wenn die Einträge nicht editierbar sein sollen, muss man die Eigenschaft "DropDownStyle" auf "DropDownList" setzen.

Dann hat man allerdings das Problem, dass die ComboBox nach dem Programmstart erst mal leer ist.

Dies kann man verhindern, indem man z.B. in Form1.Shown einfügt: comboBox1.SelectedIndex = 0;

8 WPF

WPF ist der Nachfolger von Windows Forms.

Vorteile: Gute grafische Effekte

Nachteile: Es gibt standardmäßig nicht so viele Elemente in der Toolbox wie bei Windows Forms. Abhilfe: Externe Toolkits hinzufügen, z.B. "Extended WPF Toolkit". WPF ist komplizierter als Windows Forms.

In der *.xaml Datei stehen die Eigenschaften der Tools.

In der *.cs Datei stehen die Aktionen (der Code)

Image ersetzt PictureBox

Grid ersetzt TableLayout

Timer wird ersetzt durch einen Thread, in dem Thread.Sleep() aufgerufen wird. Alternative: Timer Objekt per Code erzeugen

SerialPort direkt im Code erzeugen (in Windows Forms nachschauen wie der Code aussehen muss)

Im Gegensatz zu Windows Forms ist es bei WPF durchaus sinnvoll, direkt im xaml-Code zu editieren.

<tag> </tag> kann abgekürzt werden zu: <tag />

ScrollViewer

StackPanel

Button

Objekte im Designer verschieben: Am Rahmen anpacken!

Windows Forms Objekte können in einem WindowsFormsHost verwendet werden

(keine Ahnung ob das Sinn macht)

-- siehe auch Jones/Freeman: Visual C# Recipes: ab Seite 789

8.1 WPF ComboBox

Im Unterschied zu Windows Forms enthält die Combobox bei WPF keine Strings, sondern Objekte.
Die Abfrage, welches Element ausgewählt ist, geht so:

```
ComboBoxItem cbi = ComboBox1.SelectedItem as ComboBoxItem;
oder so:
ComboBoxItem cbi = (ComboBoxItem) cbRechenoperation.SelectedItem;

switch (cbi.Content)    // Fallunterscheidung
{
   case "+":
   ergebnis = wert1 + wert2;
   break;
}



/*   Unterschied zwischen "foo=baa as klasse" und "foo=(klasse) baa"
 *
 *   Falls die Objekte aus unterschiedlichen Klassen sind:
 *   () casting löst Exception aus
 *   as liefert Null-Objekt zurück (ohne Exception)
 *
 *   Wenn es einen Konvertierungsvorgang vonm baa nach klasse gibt,
 *   dann konvertiert () während "as" ein Null-Objekt zurück gibt.
```

```
*
*   int32 = int16 as int32;  Null
*   int32 = (int32) int16;  Konvertierung
*
*   "as" ist meist besser
*
*   Alternativ kann man die Items mit Namen versehen, und dann anhand
*    des Namens die Fallunterscheidung machen:
*/
   Control ctrl = (Control) ComboBox1.SelecedItem;
   switch (ctrl.name)
   {
      case "Name1":
         {
         }
         break;
   }
```

8.2 WPF Grafik erzeugen (Linie malen)

```
private void button_Click(object sender, RoutedEventArgs e)
{
    PixelFormat pf = PixelFormats.Rgb24;

    int w = (int)image.Width;
    int h = (int)image.Height;
    int zeilengroesse = (w * pf.BitsPerPixel + 7) / 8;
    byte[] pixelData = new byte[zeilengroesse * h];

    Color c = Color.FromRgb(255, 255, 255);

    for (int i = 0; i < 100; i++)
    {
        int xIndex = i/*x*/ * 3;
        int yIndex = i/*y*/ * zeilengroesse;
        pixelData[xIndex + yIndex] = c.R;
        pixelData[xIndex + yIndex + 1] = c.G;
        pixelData[xIndex + yIndex + 2] = c.B;
    }

    BitmapSource bitmap = BitmapSource.Create(w, h, 96, 96, pf, null, pixelData, zeilengroesse);

    WriteableBitmap writeableBitmap = new WriteableBitmap(
                    (int)this.Width,
                    (int)this.Height,
                    96,
                    96,
                    PixelFormats.Bgr32,
                    null);

    image.Source = bitmap;
}
```

8.3 Eine externe Toolbox hinzufügen (Extended WPF Toolkit)

1. Projekt --> NuGet --> Pakete verwalten

2. Links auf Online klicken

3. Oben rechts im Suchfenster "WPF" eingeben

4. Die gewünschte Toolbox auswählen und auf "installieren" klicken.

5. Jetzt muss man suchen wohin die Toolbox installiert wurde, in diesem Fall:

Eigene_Dokumente / Visual_Studio_2013 / Projects / Projektname / packages /

Extended.Wpf.Tollkit.3.3.0 / lib / net40 / Xceed.Wpf.Toolkit.dll

6. Im Werkzeugkasten Rechtsklick --> Registerkarte hinzufügen und einen Namen eingeben

7. Die *.dll Datei mit der Maus auf den neuen Namen ziehen. Alle anderen

Dateien in dem Ordner werden aber auch benötigt.

8. Nun stehen die neuen Tools zur Verfügung.

8.4 In der WPF Anwendung ein zweites Fenster öffnen

WPF Projekt erzeugen

(Rechtsklick auf das Projekt) --> hinzufügen --> Fenster --> Fenster(WPF)

Name: Window1.xaml

Im Hauptfenster Button hinzufügen, wenn der angeklickt wird soll das zweite Fenster aufgehen, aber immer nur einmal.

Der Quellcode für das Hauptfenster sieht so aus:

```
public partial class MainWindow : Window
{
    public Window1 zf = null;

    public MainWindow()
    {
        InitializeComponent();
    }

    private void Button_Click(object sender, RoutedEventArgs e)
    {
        if (zf == null)
        {
            zf = new Window1();
            zf.meinhauptfenster = this;
        }
        zf.Show();
    }
}
```

Der Quellcode für das zweite Fenster sieht so aus:

```
public partial class Window1 : Window
{
    public MainWindow meinhauptfenster = null;

    public Window1()
```

```
    {
        InitializeComponent();
    }

    private void Window1Closed(object sender, EventArgs e)
    {
        meinhauptfenster.zf = null;
    }
}
```

9 Sound-Dateien

9.1 Sound-Datei einlesen

```
public class WaveFile
{
    string format;
    int audioFormat;
    int numChannels;
    int sampleRate;
    int byteRate;
    int blockAlign;
    int bitsPerSample;
    int dataSize;
    Single[][] data;

    public int NumChannels    // liefert die Anzahl der Kanäle zurück
    {
        get { return numChannels; }
    }

    public int Length()
    {
         return(dataSize / (numChannels * bitsPerSample / 8));
    }

    public Single[] getChannel(int channel)   // liefert einen Kanal als
    {                                         //  eindimensionales Array zurück
        return(data[channel]);
    }

    public void LoadWave(string path)    // liest eine Wave Datei ein
    {
        System.IO.FileStream fs = System.IO.File.OpenRead(path); // zu lesende Wave Datei öffnen
        System.Text.ASCIIEncoding Encoder = new ASCIIEncoding();
        byte[] temp = new byte[44];

        fs.Read(temp, 0, 44);
```

```
        format = Encoder.GetString(temp, 8, 4);
        if (format != "WAVE")
            throw new Exception("Keine WAVE Datei !");
        audioFormat = System.BitConverter.ToInt16(temp, 20);
        numChannels = System.BitConverter.ToInt16(temp, 22);
        sampleRate = System.BitConverter.ToInt32(temp, 24);
        byteRate = System.BitConverter.ToInt32(temp, 28);
        blockAlign = System.BitConverter.ToInt16(temp, 32);
        bitsPerSample = System.BitConverter.ToInt16(temp, 34);
        dataSize = System.BitConverter.ToInt32(temp, 40);

        // der 1. Index von Data spezifiziert den Audiokanal, der 2. das Sample
        data = new Single[numChannels][];

        // für jeden Kanal das Data Array auf die Anzahl der Samples dimensionalisieren
        for (int i = 0; i < numChannels; i++)
            data[i] = new Single[dataSize / (numChannels * bitsPerSample / 8)];
        int bytesPerSample = blockAlign / numChannels;

        temp[0] = 0;
        temp[1] = 0;
        int offset = 4 - bytesPerSample;
        for (int i = 0; i < data[0].Length; i++)     // nacheinander alle Samples durchgehen
        {
            for (int j = 0; j < numChannels; j++)   // alle Audiokanäle pro Sample durchgehen
            {
                if (fs.Read(temp, offset, bytesPerSample) > 0)
                    data[j][i] = (Single)System.BitConverter.ToInt32(temp, 0);
                else
                    throw new Exception("Datei ist zu kurz !");
            }
        }
    }
}
```

9.2 Sound-Datei erzeugen, abspielen und speichern

```
using System.IO;

public static void PlayBeep(UInt16 frequency, int msDuration, double volume)
{
    var mStrm = new MemoryStream();
    BinaryWriter writer = new BinaryWriter(mStrm);

    const double TAU = 2 * Math.PI;
    int formatChunkSize = 16;
    int headerSize = 8;
    short formatType = 1;
    short tracks = 2;
    int samplesPerSecond = 44100;
    short bitsPerSample = 16;
    short frameSize = (short)(tracks * ((bitsPerSample + 7) / 8));
    int bytesPerSecond = samplesPerSecond * frameSize;
    int waveSize = 4;
    int samples = (int)((decimal)samplesPerSecond * msDuration / 1000);
    int dataChunkSize = samples * frameSize;
    int fileSize = waveSize + headerSize + formatChunkSize + headerSize + dataChunkSize;
    writer.Write(0x46464952); // = encoding.GetBytes("RIFF")
    writer.Write(fileSize);
    writer.Write(0x45564157); // = encoding.GetBytes("WAVE")
    writer.Write(0x20746D66); // = encoding.GetBytes("fmt ")
    writer.Write(formatChunkSize);
    writer.Write(formatType);
    writer.Write(tracks);
    writer.Write(samplesPerSecond);
    writer.Write(bytesPerSecond);
    writer.Write(frameSize);
    writer.Write(bitsPerSample);
    writer.Write(0x61746164); // = encoding.GetBytes("data")
    writer.Write(dataChunkSize);
    double theta = frequency * TAU / (double)samplesPerSecond;
    Random ra = new Random();
    for (int step = 0; step < samples; step++)
    {
        short s = (short)(32767 * volume * Math.Sin(theta * (double)step));   // Kanal 1
        writer.Write(s);
```

```
        s = (short)(ra.Next(65535));                                    // Kanal 2
        writer.Write(s);
    }

    mStrm.Seek(0, SeekOrigin.Begin);
    new System.Media.SoundPlayer(mStrm).Play();

    FileStream fs = new FileStream("out.wav", FileMode.Create);
    mStrm.WriteTo(fs);
    fs.Close();

    writer.Close();
    mStrm.Close();
}
```

Alternativ kann man eine Sound-Datei auch so abspielen:

```
Process.Start("c:/ffmpeg/ffplay.exe", "-vn -nodisp -autoexit out.wav");
```

9.3 Synthesizer

Problem: "SoundPlayer" scheint Sound-Dateien zu verschlucken, wenn der Sound kürzer als ca. 45ms ist.

Lösung: Es ist ein Hardware-Problem. Möglicherweise werden die Lautsprecher im Notebook zu früh abgeschaltet, so dass am Ende ein Stück vom Sound weggeschnitten wird. Das Problem tritt nicht so deutlich auf, wenn man die Lautsprecher ausschaltet indem man einen Klinkenstecker in den Audio-Ausgang steckt. Workaround: Am Ende des Streams 50ms Stille anfügen.

Bemerkenswert ist, dass man bei einem 1kHz Ton einen deutlichen Unterschied hören kann, je nachdem ob die Abtastrate 44.1kHz oder 48kHz ist.

```
using System;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.IO;

namespace Sound_Test
{
    public partial class Form1 : Form
    {
        double[] sound;
        double[] envelope;
        int length;
        int samplesPerSecond;

        public Form1()
        {
            InitializeComponent();
            sound = new double[480000];
            envelope = new double[480000];
            calculate();
        }

        public void calculate()
        {
            double tDelay = 0.001 * System.Convert.ToDouble(numericUpDown10.Value);
            double tAttack = 0.001 * System.Convert.ToDouble(numericUpDown1.Value);
            double nlAttack = Math.Pow(10, 0.1 * System.Convert.ToDouble(numericUpDown7.Value));
            double tHold = 0.001 * System.Convert.ToDouble(numericUpDown2.Value);
            double tDecay = 0.001 * System.Convert.ToDouble(numericUpDown3.Value);
            double nlDecay = Math.Pow(10, 0.1 * System.Convert.ToDouble(numericUpDown8.Value));
            double tSustain = 0.001 * System.Convert.ToDouble(numericUpDown4.Value);
            double sustain = 0.01 * System.Convert.ToDouble(numericUpDown5.Value);
            double tRelease = 0.001 * System.Convert.ToDouble(numericUpDown6.Value);
```

```
            double nlRelease = Math.Pow(10, 0.1 * System.Convert.ToDouble(numericUpDown9.Value));
            double tEndDelay = 0.001 * System.Convert.ToDouble(numericUpDown12.Value);

            double duration = tDelay + tAttack + tHold + tDecay + tSustain + tRelease + tEndDelay;

            double frequency = System.Convert.ToDouble(numericUpDown11.Value);
            samplesPerSecond = System.Convert.ToInt32(comboBox1.Text);
            length = (int)(samplesPerSecond * duration);

            double theta = frequency * 2 * Math.PI;
            double t;

            for (int i = 0; i < length; i++)
            {
                t = (double)i / samplesPerSecond;

                if (t < tDelay)
                    envelope[i] = 0;
                else
                {
                    t -= tDelay;
                    if (t < tAttack)
                        envelope[i] = Math.Pow(t / tAttack, nlAttack);
                    else
                    {
                        t -= tAttack;
                        if (t < tHold)
                            envelope[i] = 1;
                        else
                        {
                            t -= tHold;
                            if (t < tDecay)
                                envelope[i] = sustain + (1 - sustain) * Math.Pow((tDecay - t) / tDecay, nlDecay);
                            else
                            {
                                t -= tDecay;
                                if (t < tSustain)
                                    envelope[i] = sustain;
                                else
                                {
                                    t -= tSustain;
                                    if (t < tRelease)
                                        envelope[i] = sustain * Math.Pow((tRelease - t) / tRelease, nlRelease);
                                    else
                                        envelope[i] = 0;
                                }
                            }
                        }
                    }
```

```
            }

            t = (double)i / samplesPerSecond;
            sound[i] = envelope[i];
            if (t > 0)
                sound[i] *= Math.Sin(theta * t);
        }

        pictureBox1.Refresh();
        play();
    }

    private void pictureBox1_Paint(object sender, PaintEventArgs e)
    {
        Graphics g = e.Graphics;
        Pen p1 = new Pen(Color.Black);
        Pen p2 = new Pen(Color.Red);
        g.Clear(Color.Yellow);
        Point[] curvePoints = new Point[length];
        Point[] envelopePoints = new Point[length];

        for (int i = 0; i < length; i++)
        {
            curvePoints[i] = new Point((int)(i * pictureBox1.Width / length), (int)((-0.5 * (sound[i] - 1.0)) * pictureBox1.Height));
            envelopePoints[i] = new Point((int)(i * pictureBox1.Width / length), (int)((0.999 - envelope[i]) * pictureBox1.Height));
        }
        g.DrawLines(p1, curvePoints);
        g.DrawLines(p2, envelopePoints);
    }

    public void play()
    {
        MemoryStream mStrm = new MemoryStream();
        BinaryWriter writer = new BinaryWriter(mStrm);

        int formatChunkSize = 16;
        int headerSize = 8;
        short formatType = 1;
        short tracks = 1;
        short bitsPerSample = 16;
        short frameSize = (short)(tracks * ((bitsPerSample + 7) / 8));
        int bytesPerSecond = samplesPerSecond * frameSize;
        int waveSize = 4;
        int dataChunkSize = length * frameSize;
        int fileSize = waveSize + headerSize + formatChunkSize + headerSize + dataChunkSize;
        writer.Write(0x46464952); // = encoding.GetBytes("RIFF")
        writer.Write(fileSize);
        writer.Write(0x45564157); // = encoding.GetBytes("WAVE")
        writer.Write(0x20746D66); // = encoding.GetBytes("fmt ")
```

```
            writer.Write(formatChunkSize);
            writer.Write(formatType);
            writer.Write(tracks);
            writer.Write(samplesPerSecond);
            writer.Write(bytesPerSecond);
            writer.Write(frameSize);
            writer.Write(bitsPerSample);
            writer.Write(0x61746164); // = encoding.GetBytes("data")
            writer.Write(dataChunkSize);

            for (int i = 0; i < length; i++)
            {
                writer.Write((short)(32767 * sound[i]));
            }

            writer.Flush();
            mStrm.Flush();
            mStrm.Seek(0, SeekOrigin.Begin);
            new System.Media.SoundPlayer(mStrm).Play();

            /*
            FileStream fs = new FileStream("out.wav", FileMode.Create);
            mStrm.WriteTo(fs);
            fs.Close();
            */

            writer.Close();
            mStrm.Close();
        }


        private void numericUpDown1_ValueChanged(object sender, EventArgs e)
        {
            calculate();
        }

        private void numericUpDown2_ValueChanged(object sender, EventArgs e)
        {
            calculate();
        }

        private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
        {
            calculate();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            play();
```

```
        }

        private void numericUpDown2_ValueChanged_1(object sender, EventArgs e)
        {
            calculate();
        }

        private void numericUpDown3_ValueChanged(object sender, EventArgs e)
        {
            calculate();
        }

        private void numericUpDown4_ValueChanged(object sender, EventArgs e)
        {
            calculate();
        }

        private void numericUpDown5_ValueChanged(object sender, EventArgs e)
        {
            calculate();
        }

        private void numericUpDown6_ValueChanged(object sender, EventArgs e)
        {
            calculate();
        }

        private void numericUpDown7_ValueChanged(object sender, EventArgs e)
        {
            calculate();
        }

        private void numericUpDown8_ValueChanged(object sender, EventArgs e)
        {
            calculate();
        }

        private void numericUpDown9_ValueChanged(object sender, EventArgs e)
        {
            calculate();
        }

        private void numericUpDown10_ValueChanged(object sender, EventArgs e)
        {
            calculate();
        }

        private void numericUpDown12_ValueChanged(object sender, EventArgs e)
        {
```

```
            calculate();
        }

    }
}
```

10 Midi

Midi-dot-net Bibliothek: https://code.google.com/archive/p/midi-dot-net/

Die Datei Midi.dll wird benötigt und als Referenz zu dem Projekt hinzugefügt.

Es gibt noch eine midi-dot-net2 Bibliothek, aber die ist problematisch weil sie .NET 4.7 erfordert und weil sie nicht als *.dll verfügbar ist und weil sie schlecht dokumentiert ist.

Man findet über Nuget auch noch weitere Midi-Bibliotheken, aber überzeugt hat mich keine davon.

```
using System;
using System.Threading.Tasks;
using System.Windows.Forms;
using Midi;

namespace Midi_Test
{
    public partial class Form1 : Form
    {
        OutputDevice outputDevice;

        public Form1()
        {
            InitializeComponent();

            string myInstrument = "MatrixBrute";      // "MatrixBrute" connected via USB (this is better than a USB/Midi adapter)
                                                      // "MicroBrute" connected via USB
                                                      // "USB2.0-MIDI" USB/MIDI adapter
                                                      // "USB A" another USB/MIDI adapter
                                                      // "Microsoft GS Wavetable Synth" a simple software synthesizer in Windows

            foreach (OutputDevice device in OutputDevice.InstalledDevices)
            {
                if (device.Name == myInstrument)
                    outputDevice = device;
            }
            outputDevice.Open();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            if (outputDevice != null)
            {
                outputDevice.SendControlChange(Channel.Channel1, Midi.Control.Volume, 127);    // Volume

                for (int i = 0; i < 100; i++)
                {
                    double f1 = 400 + 1 * i;     // rising frequency
```

```
                double f2 = 1000 - 5 * i;     // falling frequency

                double midiNumber1 = (70 + 12 * Math.Log(f1 / 440) / Math.Log(2));    // Midi number as floating point
                double midiNumber2 = (70 + 12 * Math.Log(f2 / 440) / Math.Log(2));    // Midi number as floating point

                outputDevice.SilenceAllNotes();

                if (false)     // MatrixBrute in Duo-Split mode, on Midi channels 1 and 2:
                {
                    outputDevice.SendNoteOn(Channel.Channel1, (Pitch)midiNumber1, 80);                                  // Channel 1 is VCO1
                    outputDevice.SendControlChange(Channel.Channel1, (Midi.Control)65, (int)(64 * (midiNumber1 % 1)));  // VCO1 Fine
                    outputDevice.SendControlChange(Channel.Channel1, (Midi.Control)18, (int)127);                       // Mixer VCO1 (Volume)

                    outputDevice.SendNoteOn(Channel.Channel2, (Pitch)midiNumber2, 80);                                  // Channel 2 is VCO3
                    outputDevice.SendControlChange(Channel.Channel2, (Midi.Control)20, (int)127);                       // Mixer VCO3 (Volume)
                }

                if (false)     // MatrixBrute in Polyphonic mode, on Midi channel 1:
                {
                    outputDevice.SendNoteOn(Channel.Channel1, (Pitch)midiNumber1, 80);                                  // First note is played by VCO1
                    outputDevice.SendControlChange(Channel.Channel1, (Midi.Control)65, (int)(64 * (midiNumber1 % 1)));  // VCO1 Fine
                    outputDevice.SendControlChange(Channel.Channel1, (Midi.Control)18, (int)127);                       // Mixer VCO1 (Volume)

                    outputDevice.SendNoteOn(Channel.Channel1, (Pitch)midiNumber2, 80);                                  // Second note is played by VCO2
                    outputDevice.SendControlChange(Channel.Channel1, (Midi.Control)71, (int)(64 * (midiNumber1 % 1)));  // VCO2 Fine
                    outputDevice.SendControlChange(Channel.Channel1, (Midi.Control)19, (int)127);                       // Mixer VCO2 (Volume)

                    outputDevice.SendNoteOn(Channel.Channel1, (Pitch)midiNumber3, 80);                                  // Third note is played by VCO3
                    outputDevice.SendControlChange(Channel.Channel1, (Midi.Control)20, (int)127);                       // Mixer VCO2 (Volume)
                }

                //  outputDevice.SendNoteOn(Channel.Channel2, (Pitch)80, 80);
                //  outputDevice.SendControlChange(Channel.Channel2, Midi.Control.RegisteredParameterNumberMSB, (int)0);   // VCO3 with fine adjustment?
                //  outputDevice.SendControlChange(Channel.Channel2, Midi.Control.RegisteredParameterNumberLSB, (int)22);
                //  outputDevice.SendControlChange(Channel.Channel2, (Midi.Control)22, (int)(midiNumber2));  // MSB
                //  outputDevice.SendControlChange(Channel.Channel2, (Midi.Control)54, (int)i);              // LSB    But this didn't work with MatrixBrute


                if (true)     // SubSequent37 on Midi channel 3:
                {
                    outputDevice.SendNoteOn(Channel.Channel3, (Pitch)midiNumber1, 80);                                  // Note is played by OSC2
                    outputDevice.SendControlChange(Channel.Channel3, (Midi.Control)12, (int)(8 * (midiNumber1 % 1)));   // OSC2 Freq (fine adjustment)
                    outputDevice.SendControlChange(Channel.Channel3, (Midi.Control)116, (int)127);                      // OSC2 Level (Volume)
                }

                System.Threading.Thread.Sleep(200);
            }
            outputDevice.SilenceAllNotes();
        }
    }

    private void button2_Click(object sender, EventArgs e)
    {
        outputDevice.SilenceAllNotes();             // Silence button
    }
}
```

```
}
```

Bei Midi gibt es 16 Kanäle von 1 bis 16, die aber intern als 0 bis 15 dargestellt werden. Diese beiden Zeilen bewirken das gleiche:

```
outputDevice.SendNoteOn((Channel)2, (Pitch)midiNumber, 127);
outputDevice.SendNoteOn(Channel.Channel3, (Pitch)midiNumber, 127);
```

Beispiel für Midi Steuerung des Arturia MicroFreak. Das Problem ist, dass im MicroFreak viele Kommandos gar nicht implementiert sind:

```
using System;
using System.Windows.Forms;
using Midi;                  // midi.dll from https://code.google.com/archive/p/midi-dot-net/

namespace Midi_Sound_Generator
{
    public partial class Form1 : Form
    {
        OutputDevice outputDevice;
        Random r;

        public Form1()
        {
            InitializeComponent();
            r = new Random();

            string myInstrument = "USB2.0-MIDI";

            foreach (OutputDevice device in OutputDevice.InstalledDevices)
            {
                if (device.Name == myInstrument)
                    outputDevice = device;
            }
            outputDevice.Open();
        }

        private void button1_Click(object sender, EventArgs e)    // Start button
        {
            timer1.Interval = 10;  // 10ms
            timer1.Start();
        }

        private void button2_Click(object sender, EventArgs e)   // Stop button
        {
            timer1.Stop();
            outputDevice.SilenceAllNotes();                                                 // MicroFreak doesn't support this command
            outputDevice.SendControlChange(Channel.Channel3, (Midi.Control)123, (int)127);   // all notes off, MicroFreak doesn't support this command
            outputDevice.SendControlChange(Channel.Channel3, (Midi.Control)127, (int)127);   // reset, MicroFreak doesn't support this command
            outputDevice.SendControlChange(Channel.Channel3, (Midi.Control)7, (int)0);       // set volume to 0, MicroFreak doesn't support this command

            for (int n=0; n<128; n++)     // all notes off, this does work
            {
                outputDevice.SendNoteOff(Channel.Channel3, (Pitch)n, 0);
```

```
                System.Threading.Thread.Sleep(1);
            }
        }

        private void timer1_Tick(object sender, EventArgs e)
        {
            int i;
            if (outputDevice != null)
            {
                i = 70 + (int)(10 * r.NextDouble());
                outputDevice.SendNoteOn(Channel.Channel3, (Pitch)i, 80);
            }
        }

        private void button3_Click(object sender, EventArgs e)
        {
            outputDevice.SendControlChange(Channel.Channel3, (Midi.Control)127, 0);   // Polyphonic on, MicroFreak doesn't support this command
            outputDevice.SendControlChange(Channel.Channel3, (Midi.Control)126, 0);   // Polyphonic off, MicroFreak doesn't support this command
            outputDevice.SendNoteOn(Channel.Channel3, (Pitch)70, 80);
            outputDevice.SendNoteOn(Channel.Channel3, (Pitch)72, 80);
            outputDevice.SendNoteOn(Channel.Channel3, (Pitch)74, 80);
            outputDevice.SendNoteOn(Channel.Channel3, (Pitch)76, 80);
        }
    }
}
```

11 Json

JSON ist ein Dateiformat zum Austauschen von beliebigen Daten.

11.1 Json mit Newtonsoft.Json

In Visual Studio über Nuget das Paket "Newtonsoft.Json" installieren.

Beispiel-Datei "regallager.json"

```
{
  "etage": 1,
  "turm": 2,
  "ausfuehrung": {
    "boden": "holz",
    "hoehe": 200,
    "tragkraft": 150,
    "plaetze": 4
  },
  "inhalt": [
    {
      "referenznummer": "x001",
      "platz": 3,
      "name": "weihnachtsmänner",
      "kunde": {
        "name": "edeka",
        "plz": "12345"
      },
      "gewicht": 20
    },
    {
      "referenznummer": "x031",
      "platz": 2,
      "name": "osterhasen",
```

```
      "kunde": {
        "name": "edeka",
        "plz": "12345"
      },
      "gewicht": 10
    }
  ]
}
```

Beispiel 1:

In diesem Beispiel kann man die Klassen Regalplatz, Ausführung, Inhalt und Kunde automatisch so in den Quellcode einfügen:

In der Json Datei alles markieren und in die Zwischenablage übernehmen, dann bearbeiten --> Inhalte einfügen --> Json als Klassen einfügen

```
using Newtonsoft.Json;

namespace nr8
{
    class Program
    {
        static void Main(string[] args)
        {
            using (StreamReader sr = new StreamReader(@"R:\regallager.json"))
            {
                String json = sr.ReadToEnd();
                Regalplatz rp = JsonConvert.DeserializeObject<Regalplatz>(json);
                foreach(Inhalt inhalt in rp.inhalt)
                {
                    Console.WriteLine(inhalt.gewicht);   // alle Gewichte ausgeben
                }
            }
            Console.ReadLine();
        }
    }

    public class Regalplatz
    {
        public int etage { get; set; }
        public int turm { get; set; }
        public Ausfuehrung ausfuehrung { get; set; }
        public Inhalt[] inhalt { get; set; }
    }
```

```
    public class Ausfuehrung
    {
        public string boden { get; set; }
        public int hoehe { get; set; }
        public int tragkraft { get; set; }
        public int plaetze { get; set; }
    }

    public class Inhalt
    {
        public string referenznummer { get; set; }
        public int platz { get; set; }
        public string name { get; set; }
        public Kunde kunde { get; set; }
        public int gewicht { get; set; }
    }

    public class Kunde
    {
        public string name { get; set; }
        public string plz { get; set; }
    }
}
```

Beispiel 2:

Gesamtgewicht in einem Regalplatz ermitteln. In diesem Beispiel wird auch eine Json Datei erzeugt

```
using Newtonsoft.Json;

namespace nr8
{
    class Program
    {
        static void Main(string[] args)
        {
            using (StreamReader sr = new StreamReader(@"R:\regallager.json"))
            {
                String json = sr.ReadToEnd();
                Regalplatz rp = JsonConvert.DeserializeObject<Regalplatz>(json);
                foreach(Inhalt inhalt in rp.inhalt)
```

```
            {
                Console.WriteLine(inhalt.gewicht);
            }
            String s = JsonConvert.SerializeObject(rp);   // Json Datei erzeugen
            // optional mit Formatierung, jeder Wert in einer neuen Zeile:
            //  JsonConvert.SerializeObject(rp, Formatting.Indented);
            Console.WriteLine(s);
        }
        Console.ReadLine();
    }
}

public class Regalplatz
{
    public int etage { get; set; }
    public int turm { get; set; }
    public Ausfuehrung ausfuehrung { get; set; }
    public Inhalt[] inhalt { get; set; }
    public int gesamtgewicht
    {
        get
        {
            int summe = 0;
            foreach (Inhalt i in inhalt)
            {
                summe += i.gewicht;
            }
            return summe;
        }
        set
        {
            // schreibender Zugriff macht hier keinen Sinn
        }
    }
}

public class Ausfuehrung
{
    public string boden { get; set; }
    public int hoehe { get; set; }
    public int tragkraft { get; set; }
    public int plaetze { get; set; }
}
```

```
    public class Inhalt
    {
        public string referenznummer { get; set; }
        public int platz { get; set; }
        public string name { get; set; }
        public Kunde kunde { get; set; }
        public int gewicht { get; set; }
    }

    public class Kunde
    {
        public string name { get; set; }
        public string plz { get; set; }
    }
}
```

11.2 Json ohne externe Pakete

-- siehe auch Jones/Freeman: Visual C# Recipes: Seite 90

Nachteil: Kann nicht so serialisieren, dass jeder Wert in einer neuen Zeile steht.

Im Projektmappen-Explorer unter "Verweise" auf "Verweis hinzufügen" gehen und dann bei System.Runtime.Serialization das Häkchen setzen.

```
using System;
using System.Collections.Generic;
using System.Runtime.Serialization.Json;
using System.IO;
using System.Text;

MemoryStream ms = new MemoryStream();
DataContractJsonSerializer dcjs = new DataContractJsonSerializer(liste.GetType());
dcjs.WriteObject(ms, liste);
s = Encoding.Default.GetString(ms.ToArray());
```

12 Zeit

```
TimeSpan ts = DateTime.UtcNow - new DateTime(2000, 1, 1, 12, 01, 06, DateTimeKind.Utc);
double jh = Convert.ToDouble(ts.TotalDays) / 36525;    /* Zeit in Jahrhunderten */

DateTime dt = myFile.CreationTime.ToUniversalTime();
dt = dt.AddHours(zeitkorrektur);
```

12.1 Zeitverzögerung Delay

```
System.Threading.Thread.Sleep(1);    // in Millisekunden
```

13 Serielle Schnittstelle

13.1 Mitutoyo Mikrometerschraube seriell einlesen

```
serialPort1.PortName = "COM11";
serialPort1.Open();          // Mitutoyo
serialPort1.NewLine = "\r";  // Newline Character for Mitutoyo
serialPort1.DtrEnable = true;
serialPort1.RtsEnable = true;
serialPort1.ReadTimeout = 500;

double z;
serialPort1.DiscardInBuffer();
serialPort1.WriteLine("1");
try
{
    z = Convert.ToDouble(serialPort1.ReadLine().Substring(3,9),System.Globalization.NumberFormatInfo.InvariantInfo);
}
catch
{
    z = 9999;
}
textBox4.Text = z.ToString("F3");
```

13.2 Relais-Ansteuerung

Es gibt 12V Reed-Relais, die so wenig Strom brauchen dass man sie direkt mit dem RTS-Pin der seriellen Schnittstelle ansteuern kann, ohne dass man eine zusätzliche Stromversorgung braucht:

```
serialPort2.PortName = "COM1";
serialPort2.RtsEnable = true;  // Remote Control Relay
serialPort2.Open();
System.Threading.Thread.Sleep(500);
serialPort2.RtsEnable = false;
serialPort2.Close();
```

13.3 CO2-Sensoren MH-Z14A und MH-Z19B seriell ansteuern und auslesen

Der Sensor kann direkt an einen FTDI Adapter angeschlossen werden und wird über diesen mit Spannung versorgt. Der Adapter muss auf 5V Betriebsspannung eingestellt werden (intern den roten Draht an +5V anlöten).

Verdrahtung:

Pin am MH-Z19B	Funktion	Pin am FTDI Adapter
1 braun	V_o Analoge Ausgangsspannung	
2 weiss	n.c.	
3 schwarz	GND	GND schwarz
4 rot	+5V	+5V rot
5 blau	RXD (Eingang am HM-Z19B)	TXD grün
6 grün	TXD (Ausgang am HM-Z19B)	RXD weiss
7 gelb	n.c.	

Datenblatt vom Hersteller: https://www.winsen-sensor.com/d/files/MH-Z19B.pdf

Ein Arduino-Projekt mit Bibliotheken für diesen Sensor: https://libraries.io/platformio/MH-Z19

Hacker-Seite: https://revspace.nl/MH-Z19B

Die Kalibrierung muss bei einem CO2 Gehalt von 400ppm durchgeführt werden. Standardmäßig ist die automatische Kalibrierung aktiviert, die nach 24 Stunden den minimalen CO2 Gehalt, der innerhalb von 24 Stunden vorgekommen ist, auf 400ppm setzt. Dies ist nur sinnvoll wenn sich der Sensor an einer Stelle befindet, die regelmäßig gelüftet wird. Im Gewächshaus macht das keinen Sinn. Man kann die automatische Kalibrierung über ein Kommando deaktivieren.

Der Meßbereich geht wahlweise bis 2000ppm, 5000ppm oder 10000ppm, oder auch andere Werte dazwischen. Im Lieferzustand ist 5000ppm eingestellt. 400ppm entspricht 0.04%, das entspricht im Jahr 2020 dem normalen CO2 Gehalt in der Atmosphäre.

Wenn man den CO2-Sensor auf den Bereich 3000ppm einstellt und den DAC-Bereich auf 0 bis 3000 setzt, dann zeigt das Voltmeter direkt den CO2 Gehalt von 400ppm bis 3000ppm an.

Der MH-Z14A kann mit der gleichen Software ausgelesen werden, aber es dauert nach dem Einschalten einige Minuten bevor man den Messwert abfragen kann.

```
using System;
using System.Data;
using System.Windows.Forms;

namespace MH_Z19B_CO2_Sensor
{
    public partial class Form1 : Form
    {
        int[] reply = new int[9];

        public Form1()
        {
            InitializeComponent();
            serialPort1.PortName = "COM3";
            serialPort1.BaudRate = 9600;
            serialPort1.StopBits = System.IO.Ports.StopBits.One;
            serialPort1.Handshake = System.IO.Ports.Handshake.None;
            serialPort1.ReadTimeout = 500;
            serialPort1.Open();
        }

        private void Read_CO2_Click(object sender, EventArgs e)
        {
            serialPort1.DiscardInBuffer();
            serialPort1.Write(new byte[] {0xff,0x01,0x86,0x00,0x00,0x00,0x00,0x00,0x79}, 0, 9);
            for (int i = 0; i < 9; i++)
            {
                reply[i] = serialPort1.ReadByte();
                // richTextBox1.AppendText(reply[i].ToString("X2") + " ");
            }
            // richTextBox1.AppendText("\n");
            label1.Text = "CO2: " + (256 * reply[2] + reply[3]).ToString() + " ppm";
        }

        private void Self_Calibration_ON_Click(object sender, EventArgs e)
        {
            serialPort1.Write(new byte[] { 0xff, 0x01, 0x79, 0xA0, 0x00, 0x00, 0x00, 0x00, 0xE6 }, 0, 9);
        }

        private void Self_Calibration_OFF_Click(object sender, EventArgs e)
        {
            serialPort1.Write(new byte[] { 0xff, 0x01, 0x79, 0x00, 0x00, 0x00, 0x00, 0x00, 0x86 }, 0, 9);
        }

        private void Detection_Range_2000ppm_Click(object sender, EventArgs e)
        {
            serialPort1.Write(new byte[] { 0xff, 0x01, 0x99, 0x00, 0x00, 0x00, 0x07, 0xD0, 0x8F }, 0, 9);
```

```
        }

        private void Detection_Range_3000ppm_Click(object sender, EventArgs e)
        {
            serialPort1.Write(new byte[] { 0xff, 0x01, 0x99, 0x00, 0x00, 0x00, 0x0B, 0xB8, 0xA3 }, 0, 9);
        }

        private void Detection_Range_5000ppm_Click(object sender, EventArgs e)
        {
            serialPort1.Write(new byte[] { 0xff, 0x01, 0x99, 0x00, 0x00, 0x00, 0x13, 0x88, 0xCB }, 0, 9);
        }

        private void Detection_Range_10000ppm_Click(object sender, EventArgs e)
        {
            serialPort1.Write(new byte[] { 0xff, 0x01, 0x99, 0x00, 0x00, 0x00, 0x27, 0x10, 0x2F }, 0, 9);
        }

        private void Zero_Calibration_400ppm_Click(object sender, EventArgs e)
        {
            serialPort1.Write(new byte[] { 0xff, 0x01, 0x87, 0x00, 0x00, 0x00, 0x00, 0x00, 0x78 }, 0, 9);
        }

        private void Read_DAC_Bounds_Click(object sender, EventArgs e)
        {
            serialPort1.DiscardInBuffer();
            serialPort1.Write(new byte[] { 0xff, 0x01, 0xA5, 0x00, 0x00, 0x00, 0x00, 0x00, 0x5A }, 0, 9);
            for (int i = 0; i < 9; i++)
            {
                reply[i] = serialPort1.ReadByte();
                richTextBox1.AppendText(reply[i].ToString("X2") + " ");
            }
            richTextBox1.AppendText("\n");
        }

        private void Set_DAC_400_2000_Click(object sender, EventArgs e)
        {
            serialPort1.Write(new byte[] { 0xff, 0x01, 0xA4, 0x01, 0x90, 0x07, 0xD0, 0x00, 0xF3 }, 0, 9);
        }

        private void Set_DAC_0_3000_Click(object sender, EventArgs e)
        {
            serialPort1.Write(new byte[] { 0xff, 0x01, 0xA4, 0x00, 0x00, 0x0B, 0xB8, 0x00, 0x98 }, 0, 9);
        }
    }
}
```

14 Simulierter F1 oder ctrl-T Tastendruck

Beispiel 1:

```
using System.Runtime.InteropServices;

private void button1_Click(object sender, EventArgs e)             // Wenn ich auf diesen Button drücke
{
    Process[] allNotepads = Process.GetProcessesByName("notepad"); // dann werden alle Notepad Anwendungen gesucht
                                                                   // oder "MetroPro" für MetroPro_9
                                                                   // oder "MainUI" für MetroPro_X
    if (allNotepads.Count() == 0)                                  // Fehlermeldung falls kein Notepad gefunden wurde...
    {
        MessageBox.Show("Kein Notepad gefunden!");
        return;
    }
    IntPtr notepadHandle = allNotepads.First().MainWindowHandle;   // das erste Notepad aus der Liste wird genommen...
    const uint WM_KEYDOWN = 0x100;                                 // simuliere einen "F1" Tastendruck
    const uint WM_KEYUP = 0x101;
    WinAPI.PostMessage(notepadHandle, WM_KEYDOWN, (IntPtr)Keys.F1, IntPtr.Zero);   // für MetroPro_X: Keys.F12
    WinAPI.PostMessage(notepadHandle, WM_KEYUP, (IntPtr)Keys.F1, IntPtr.Zero);     // für MetroPro_X: Keys.F12
}

public class WinAPI
{
    [DllImport("user32.dll")]
    public static extern IntPtr PostMessage(IntPtr hWnd, uint Msg, IntPtr wParam, IntPtr lParam);
}
```

Beispiel 2:

```
using System.Runtime.InteropServices;

Process[] allProcesses = Process.GetProcessesByName("4Sight");  // search all "4Sight" processes
if (allProcesses.Count() == 0)                                  // error message if none was found
{
    MessageBox.Show("Didn't find 4Sight!");
    return;
}
IntPtr processHandle = allProcesses.First().MainWindowHandle;   // use the first process
```

```
const uint WM_KEYDOWN = 0x100;                  // now send a simulated a "CTRL-T" keypress to 4Sight
const uint WM_KEYUP = 0x101;
const int VK_CONTROL = 0x11;
WinAPI.keybd_event(VK_CONTROL, 0x9d, 0, 0);                                  // hold CTRL button down
WinAPI.PostMessage(processHandle, WM_KEYDOWN, (IntPtr)Keys.T, IntPtr.Zero); // press T button
System.Threading.Thread.Sleep(100);
WinAPI.PostMessage(processHandle, WM_KEYUP, (IntPtr)Keys.T, IntPtr.Zero);   // release T button
WinAPI.keybd_event(VK_CONTROL, 0x9d, 0x02, 0);                               // release CTRL button

public class WinAPI
{
    [DllImport("user32.dll")]
    public static extern IntPtr PostMessage(IntPtr hWnd, uint Msg, IntPtr wParam, IntPtr lParam);

    [DllImport("user32.dll")]
    public static extern void keybd_event(byte bVk, byte bScan, uint dwFlags, int dwExtraInfo);
}
```

15 E-Mail

In diesem Beispiel wird der lokal zugewiesene E-Mail Client gestartet um eine E-Mail zu erzeugen, in welcher der Empfänger, der Betreff und der Inhalt schon vor-ausgefüllt sind:

```
using System.Diagnostics;
var url = "mailto:meine_email@gmx.de?subject=Test&body=Dies ist ein Test";
Process.Start(url);
```

16 Bibliotheken

CSharpFITS http://skyservice.pha.jhu.edu/develop/vo/CSharpFITS/

Theoretisch kann man externe DLL's in die *.exe Datei einbinden, aber das ist sehr kompliziert. Erst mal Finger weg davon!
Es könnte mit dem Tool ILMerge funktionieren, aber das ist mir zu kompliziert zu installieren:
https://it-stack.de/11/02/2010/mehrere-projektdateien-wie-exe-und-dll-in-eine-exe-mergen/
https://github.com/dotnet/ILMerge/blob/master/README.md

17 Näherungen

sqrt(x^2 + y^2) kann angenähert werden durch max(abs(x),abs(y)) + 13/32 * min(abs(x),abs(y))
Der Fehler ist im Bereich -0.6% bis +7.9%

18 Diverse unsortierte Dinge

http://www.astro-electronic.de/Topographie-Anleitung.txt

```
namespace meinprogramm
{
   public partial class Form1 : Form      // Der Name der Klasse wird groß geschrieben!
   {
```

```
public int[] x = new int[20];   // Array anlegen
string[] lines;
public Bitmap bitmap1 = new Bitmap(301,301);
public Double[,] map = new Double[301,301];  // zweidimensionales Array anlegen
Boolean taking_pictures = false;

public Form1()
{
   InitializeComponent();
}

private void pictureBox1_Paint(object sender, PaintEventArgs e)
{
   Graphics g = e.Graphics;
   Pen pen1 = new Pen(Color.Black, 2);
   g.DrawLine(pen1, 10, 10, 99, 99);
   Pen pen2 = new Pen(Color.OrangeRed);
   pen2.DashStyle = System.Drawing.Drawing2D.DashStyle.Dash;
   pen2.DashPattern = new float[] {5,5};
}

private void Form1_Shown(object sender, EventArgs e)
{
   richTextBox1.AppendText("Hallo\n");
   int x = System.Convert.ToInt16(coord[2 * i]);
```

```
    this.Refresh();
    System.Threading.Thread.Sleep(4000);  // Programm beendet sich selbst
    this.Close();
}

private void pictureBox1_MouseMove(object sender, MouseEventArgs e)
{
    if (e.Button == MouseButtons.Left)
    {
        mouse_move(e);
    }
    if (e.Button == MouseButtons.Right)
    {
        mouse_rotate(e);
    }
}

private void pictureBox1_MouseDown(object sender, MouseEventArgs e)
{
    if (e.Button == MouseButtons.Left)
    {
        mouse_move(e);
    }
    if (e.Button == MouseButtons.Right)
    {
        int xx = e.X - 200;
```

```
            int yy = 200 - e.Y;
            startwinkel = (int)numericUpDown9.Value;
            mausw = Math.Atan2(xx, yy);
        }
    }

    private void mouse_move(MouseEventArgs e)
    {
        int xx = e.X - 200;
        int yy = 200 - e.Y;
    }

    private void mouse_rotate(MouseEventArgs e)
    {
        int xx = e.X - 200;
        int yy = 200 - e.Y;
    }

  }
}
```

********** The End **********

```
using BitMiracle.LibTiff.Classic;

using (Tiff tiff = Tiff.Open(filePath, "r"))
{
    int height = tiff.GetField(TiffTag.IMAGELENGTH)[0].ToInt();
    FieldValue[] modelPixelScaleTag = tiff.GetField((TiffTag)33550);
    FieldValue[] modelTiepointTag = tiff.GetField((TiffTag)33922);

    byte[] modelPixelScale = modelPixelScaleTag[1].GetBytes();
    double pixelSizeX = BitConverter.ToDouble(modelPixelScale, 0);
    double pixelSizeY = BitConverter.ToDouble(modelPixelScale, 8) * -1;

    byte[] modelTransformation = modelTiepointTag[1].GetBytes();
    double originLon = BitConverter.ToDouble(modelTransformation, 24);
    double originLat = BitConverter.ToDouble(modelTransformation, 32);

    double startLat = originLat + (pixelSizeY / 2.0);
    double startLon = originLon + (pixelSizeX / 2.0);

    var scanline = new byte[tiff.ScanlineSize()];

    //TODO: Check if band is stored in 1 byte or 2 bytes.
    //If 2, the following code would be required
    //var scanline16Bit = new ushort[tiff.ScanlineSize() / 2];
    //Buffer.BlockCopy(scanline, 0, scanline16Bit, 0, scanline.Length);
```

```
    double currentLat = startLat;
    double currentLon = startLon;

    for (int i = 0; i < height; i++)
    {
        tiff.ReadScanline(scanline, i); //Loading ith Line

        var latitude = currentLat + (pixelSizeY * i);
        for (var j = 0; j < scanline.Length; j++)
        {
            var longitude = currentLon + (pixelSizeX * j);
            geodata.Points[0] = new[] { new PointXY(longitude, latitude) };
            object value = scanline[j];

            //... process each data item

            yield return dataItem;
        }
    }
}
```

Stichwortverzeichnis