

Allgemeines

Titel der Übung:	GUI Programmieren
Übungsnummer:	I / 2
Übungsplatz:	T11
Datum der Übung:	22.09.2015
Klasse:	5BHEL
Übungsleiter:	Dipl. –Ing. Franz Geischläger
Schriftführer:	Thomas Gorke
Übungsteilnehmer:	Robin Mayerhofer Oliver Haschke Daniel Hogl

Inhaltsverzeichnis

Allgemeines	1
1. Microsoft Visual Studio	2
1.1. Allgemeines	2
1.2. Verwendung der Funktionen.....	2
2. Demoprogramm	3
2.1. Aufgabenstellung.....	3
2.2. Schreiben des Demoprogramms	3
2.3. Screenshot.....	3
3. Wichtige Begriffe.....	4
4. GUI Programm.....	6
4.1. Aufgabenstellung.....	6
4.2. Lösen der Aufgabe.....	6
4.2.1. GUI	6
4.2.2. Programmcode	8
5. Erkenntnisse	11
Messprotokoll.....	12
Inventarliste.....	13

1. Microsoft Visual Studio

1.1. Allgemeines

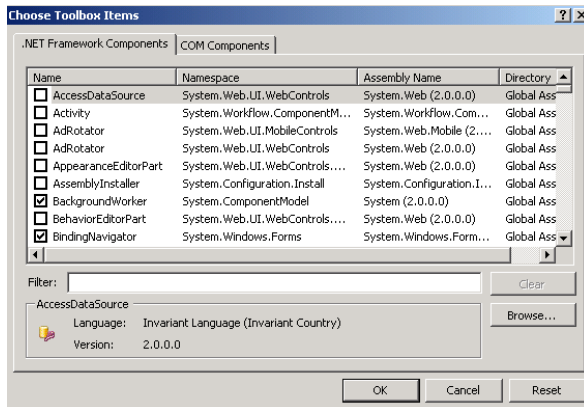
Die Entwicklungsumgebung MS Visual Studio dient zur einfachen und komfortablen Erstellung eines grafischen Userinterfaces. Der Entwickler hat die Möglichkeit, bereits vorhandene Funktionen, wie zum Beispiel Buttons, Textfelder, Eingabefelder, Listen etc. in sein Projekt einzubauen.

1.2. Verwendung der Funktionen/ Tools

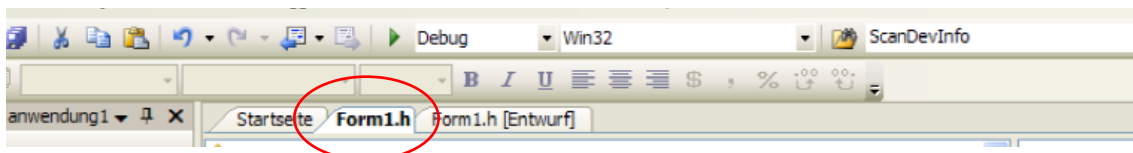
Die oben genannten Funktionen können durch einen Klick auf die jeweilige Funktion in das „UI-Fenster“ gezogen werden. Standardmäßig werden aber nicht alle verfügbaren Funktionen in der Toolbox angezeigt.

Der Entwickler kann wie folgt Features hinzufügen:

- 1) Toolbox öffnen
- 2) Rechtsklick in die Liste
- 3) „Choose Items“
- 4) Es öffnet sich folgendes Fenster:



- 5) Hier kann man auswählen, welches Tool zusätzlich heruntergeladen werden soll



Unter dem Reiter „Form.h“ kann man nun Unterprogramme, Funktionen schreiben und Variable deklarieren.

2. Demoprogramm

2.1. Aufgabenstellung

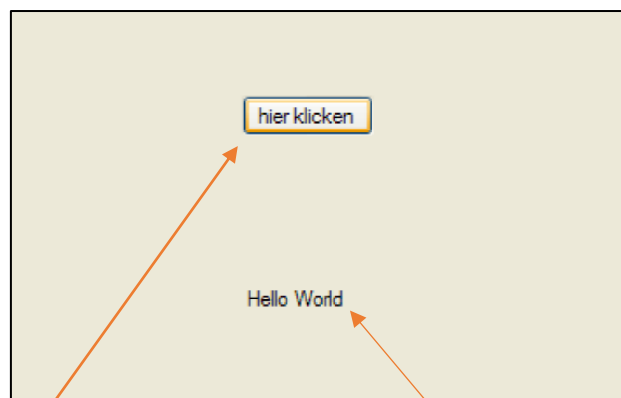
Zu Beginn der Laborübung sollte mit Hilfe der vorhandenen Anleitung (VCpp_projekt.pdf) ein einfaches Demoprogramm geschrieben werden, um sich bei der darauffolgenden Aufgabenstellung besser in der Entwicklungsumgebung (Microsoft Visual Studio / Visual C++) zurechtzufinden.

2.2. Schreiben des Demoprogramms

Wie verlangt, wurde ein kleines Programm geschrieben, welches bei dem Klick auf einen Button den Text „Hello World“ ausgibt.

2.3. Screenshot

Nach dem das Programm gestartet wurde, war folgendes Fenster zu sehen:



Beim Klick auf den Button „hier klicken“ wurde der Text „Hello World“ ausgegeben.

3. Wichtige Begriffe

Während an dem Demoprogramm gearbeitet wurde, wurden uns verschiedenste Begriffe erklärt. Zur besseren Erklärung ist hier die Header-Datei des Demoprogrammes angeführt:

```
#pragma once

namespace first_programm { //Klassen die zusammen gehören werden zusammengefasst

    using namespace System;
    using namespace System::ComponentModel;
    using namespace System::Collections;
    using namespace System::Windows::Forms;
    using namespace System::Data;
    using namespace System::Drawing;

    /// <summary>
    /// Summary for Form1
    ///
    /// WARNING: If you change the name of this class, you will need to change the
    ///           'Resource File Name' property for the managed resource compiler tool
    ///           associated with all .resx files this class depends on. Otherwise,
    ///           the designers will not be able to interact properly with localized
    ///           resources associated with this form.
    /// </summary>
    public ref class Form1 : public System::Windows::Forms::Form //Form1 erbt von
    System::Windows::Forms::Form ab
    {
    public:
        Form1(void) //Constructor
        {
            InitializeComponent();

            //Hier können Variable initialisiert werden

            //
            //TODO: Add the constructor code here
            //
        }

    protected:
        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        ~Form1() //Destructor
        {
            //Hier kann der verwendete Speicher freigegeben werden
            if (components)
            {
                delete components;
            }
        }

    private: System::Windows::Forms::Button^ button1; //Trackinghandle (Speicher, verschieben,...)
    protected:
    private: System::Windows::Forms::Label^ label1;

    private:
        /// <summary>
        /// Required designer variable.
        /// </summary>
        System::ComponentModel::Container ^components;
```

```

#pragma region Windows Form Designer generated code
/// <summary>
/// Required method for Designer support - do not modify
/// the contents of this method with the code editor.
/// </summary>
void InitializeComponent(void)
{
    this->button1 = (gcnew System::Windows::Forms::Button());
    //gc ... GarbageCollector löscht Obj wieder
    this->label1 = (gcnew System::Windows::Forms::Label());
    this->SuspendLayout();
    //
    // button1 //GUI
    //
    //Design/ Verhalten/ Texte usw. festlegen:
    this->button1->BackColor = System::Drawing::SystemColors::MenuHighlight;
    this->button1->Location = System::Drawing::Point(82, 57);
    this->button1->Name = L"button1";
    this->button1->Size = System::Drawing::Size(147, 23);
    this->button1->TabIndex = 0;
    this->button1->Text = L"Hier klicken!";
    this->button1->UseVisualStyleBackColor = false;
    this->button1->Click += gcnew System::EventHandler(this,
    &Form1::button1_Click);
    //
    // label1
    //
    this->label1->AutoSize = true;
    this->label1->Location = System::Drawing::Point(79, 157);
    this->label1->Name = L"label1";
    this->label1->Size = System::Drawing::Size(35, 13);
    this->label1->TabIndex = 1;
    this->label1->Text = L"label1";
    //
    // Form1
    //
    this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
    this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
    this->ClientSize = System::Drawing::Size(292, 273);
    this->Controls->Add(this->label1);
    this->Controls->Add(this->button1);
    this->Name = L"Form1";
    this->Text = L"Form1";
    this->Load += gcnew System::EventHandler(this, &Form1::Form1_Load);
    this->ResumeLayout(false);
    this->PerformLayout();

}
#pragma endregion
private: System::Void Form1_Load(System::Object^ sender, System::EventArgs^ e) {
}
//EventHandler:
private: System::Void button1_Click(System::Object^ sender, System::EventArgs^ e) {
    label1->Text="Hallo Welt";
}
};
}

```

Anmerkung: Die zusätzlichen Kommentare zur Erklärung wurden in pink geschrieben.

Der Eventhandler wird aufgerufen, sobald der Button gedrückt wird. In diesem Fall wird der Text von „label1“ in „Hallo Welt“ geändert.

4. GUI Programm

4.1. Aufgabenstellung

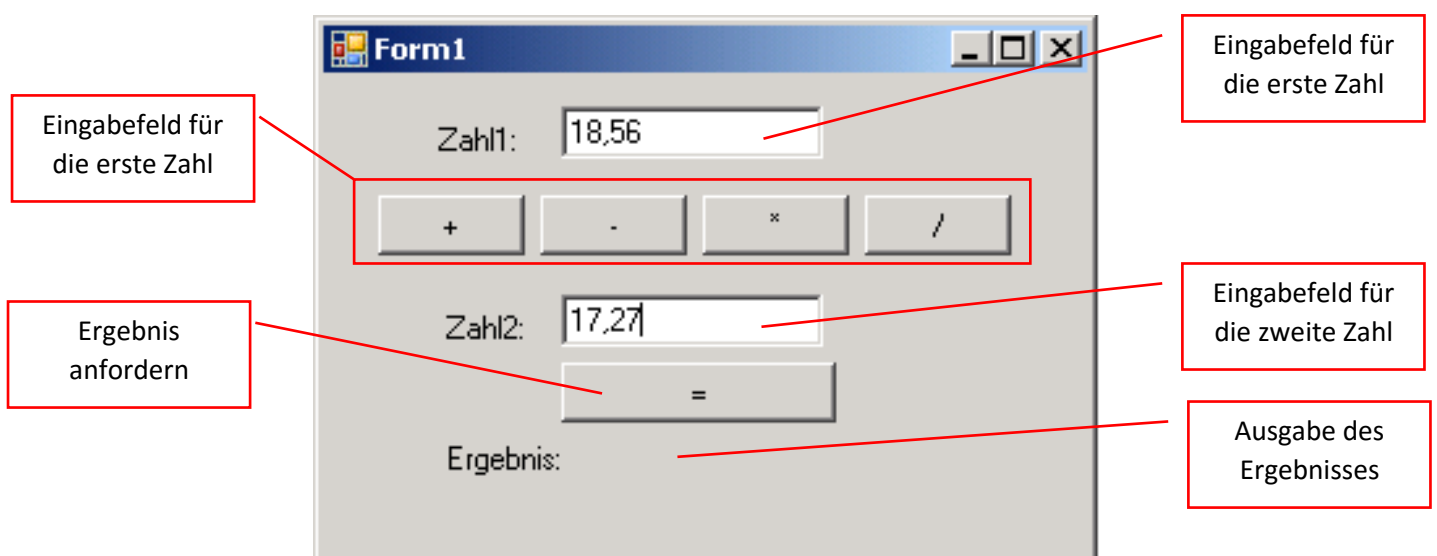
Es soll ein beliebiges Programm (z.B.: Taschenrechner, ...) in Visual C++ geschrieben werden. Dabei sollen mehrere Funktionen und Features verwendet bzw. erstellt werden.

4.2. Lösen der Aufgabe

Dieses Mal wurde ein Taschenrechner mit den Funktionen: Addieren, Subtrahieren, Dividieren und Multiplizieren geschrieben. Dabei wurde wie folgt vorgegangen:

4.2.1. GUI

Zuerst wurde das grafische Userinterface mit den Eingabeflächen und der Auswahl der Rechenoperation geschrieben.



Geplante Funktion:

Der User soll in den zwei Eingabefeldern je eine Zahl eingeben. Anschließend muss per Mausklick die Rechenoperation ausgewählt werden. Mit einem Klick auf „=" soll das Ergebnis berechnet und ausgegeben werden.

Zusätzlich soll eine Fehlermeldung ausgegeben werden, wenn der User keine Rechenzeichen ausgewählt oder nicht beide Eingabefelder ausgefüllt hat.

4.2.2. Programmcode

```
public ref class Form1 : public System::Windows::Forms::Form
{
public:
    Form1(void)
    {
        InitializeComponent();
        //Variable initialisieren:
        op=0;
        //
        //TODO: Add the constructor code here
        //
    }

protected:
    /// <summary>
    /// Clean up any resources being used.
    /// </summary>
    ~Form1()
    {
        if (components)
        {
            delete components;
        }
    }

private: System::Windows::Forms::TextBox^    textBox1;
protected:
private: System::Windows::Forms::TextBox^    textBox2;
private: System::Windows::Forms::Label^    label1;
private: System::Windows::Forms::Label^    label2;
private: System::Windows::Forms::Label^    label3;
private: System::Windows::Forms::Label^    label4;
private: System::Windows::Forms::Button^    plus;

private: System::Windows::Forms::Button^    min;

private: System::Windows::Forms::Button^    mal;
private: System::Windows::Forms::Button^    div;
private: System::Windows::Forms::Button^    erg;

    //Variablen definieren:
private: double zahl1,zahl2;
private: int op;

private:
    /// <summary>
    /// Required designer variable.
    /// </summary>
    System::ComponentModel::Container ^components;
```

```

#pragma region Windows Form Designer generated code
    /// <summary>
    /// Required method for Designer support - do not modify
    /// the contents of this method with the code editor.
    /// </summary>

    void InitializeComponent(void)
    {
        //Initialisieren der Eingabefelder, Labels, Buttons etc.
        this->textBox1 = (gcnew System::Windows::Forms::TextBox());
        this->textBox2 = (gcnew System::Windows::Forms::TextBox());
        this->label1 = (gcnew System::Windows::Forms::Label());
        this->label2 = (gcnew System::Windows::Forms::Label());
        this->label3 = (gcnew System::Windows::Forms::Label());
        this->label4 = (gcnew System::Windows::Forms::Label());
        this->plus = (gcnew System::Windows::Forms::Button());
        this->min = (gcnew System::Windows::Forms::Button());
        this->mal = (gcnew System::Windows::Forms::Button());
        this->div = (gcnew System::Windows::Forms::Button());
        this->erg = (gcnew System::Windows::Forms::Button());
        this->SuspendLayout();
        //
        // textBox1
        //
        this->textBox1->Location = System::Drawing::Point(91, 12);
        this->textBox1->Name = L"textBox1";
        .
        .
        . //Programm code wurde gekürzt, da dieser Code automatisch generiert wird
        .
        this->Controls->Add(this->textBox1);
        this->Name = L"Form1";
        this->Text = L"Form1";
        this->ResumeLayout(false);
        this->PerformLayout();
    }
#pragma endregion
//Eventhandler:
private: System::Void plus_Click(System::Object^ sender, System::EventArgs^ e) {
    op=1;
}
private: System::Void min_Click(System::Object^ sender, System::EventArgs^ e) {
    op=2;
}
private: System::Void mal_Click(System::Object^ sender, System::EventArgs^ e) {
    op=3;
}
private: System::Void div_Click(System::Object^ sender, System::EventArgs^ e) {
    op=4;
}
private: System::Void erg_Click(System::Object^ sender, System::EventArgs^ e) {
    setZahlen(); //Unterprogramm wird aufgerufen
}

```

```
//Unterprogramm:
private: void setZahlen() {

    double erg; //Variable wird definiert
    //Eingabefelder berprfen ob diese leer sind
    if((textBox1->Text!="") || (textBox2->Text!="")){

        zahl1= Convert::ToDouble(textBox1->Text);
        zahl2= Convert::ToDouble(textBox2->Text);
        //Überprüfen ob eine Rechenoperation ausgewählt wurde
        if(op!=0){
            switch (op){
                //Ergebnis berechnen
                case 1:
                    erg= zahl1+zahl2;
                    break;

                case 2:
                    erg= zahl1-zahl2;
                    break;

                case 3:
                    erg= zahl1*zahl2;
                    break;

                case 4:
                    erg= zahl1/zahl2;
                    break;
            }
            //Double to String konvertieren und ausgeben
            String^ ergebnis = Convert::ToString(erg);
            label4->Text=ergebnis;

            //Eingabefelder leeren:
            textBox1->Text="";
            textBox2->Text="";
            op=0;

        } else { //Ausgabe wenn keine Rechenoperation ausgewählt wurde
            label4->Text="kein Rechenoperation gewählt";
        }
    } else //Eingabefelder sind leer
        label4->Text="kein Zahlen eingegeben";
}

};
}
```

Programmbeschreibung:

Zu Beginn werden die Variablen „zahl1“, „zahl2“ und „op“. Definiert. In „zahl1“ und „zahl2“ werden die zwei eingegeben Zahlen des Users abgespeichert. In der Variable „op“ wird das Rechenzeichen festgelegt:

- op = 1: addieren
- op = 2: subtrahieren
- op = 3: multiplizieren
- op = 4: dividieren

Mit einem Mausklick auf ein Rechenzeichen-Symbol wird der Variable der entsprechende Wert zugewiesen. Wird auf das „= - Feld“ gedrückt wird mittels der „op“ –Variable die entsprechende Rechenoperation ausgewählt.

Wurde jedoch keine Rechenoperation gewählt, ist die Variable „op“ NULL. Daher wird statt einem Ergebnis der Text „keine Rechenoperation gewählt“ ausgegeben. Zusätzlich wird überprüft, ob die Eingabefelder leer sind. Ist dies der Fall wird der Text „keine Zahlen eingegeben“ ausgegeben.

5. Erkenntnisse

- Bei dieser Übung wurde erlernt, wie man schnell und einfach ein grafisches Userinterface mit den verschiedensten Funktionen erstellen kann.
- Weiters wurden Kenntnisse zur objektorientierten Programmierung aufgefrischt und erweitert.
- Zusätzlich wurden einige Begriffe wie zum Beispiel „namespace“, „constructor“, usw. wiederholt und erklärt.

Messprotokoll

Inventarliste