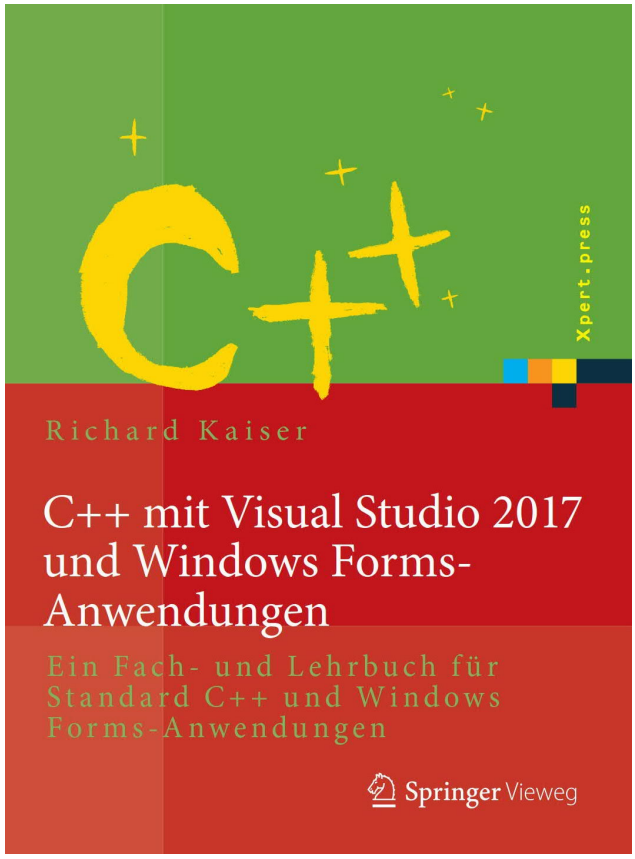
Windows Forms Projekte mit C++ in Visual Studio 2017

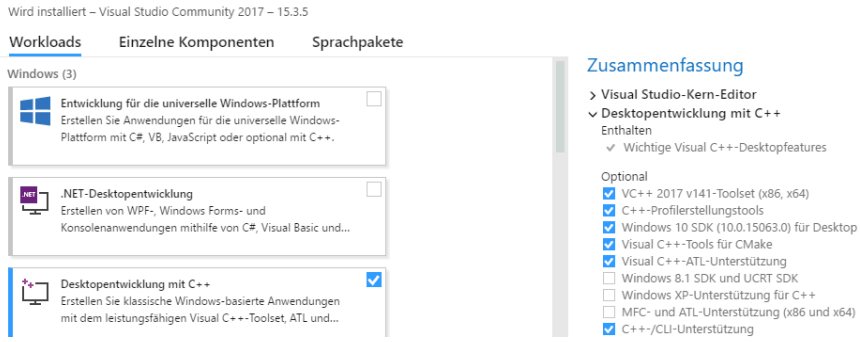
In diesem kleinen Auszug aus dem Buch



wird kurz gezeigt, wie man mit Visual Studio 2017 und früheren Versionen Windows-Programme mit einer grafischen Benutzeroberfläche entwickeln kann.

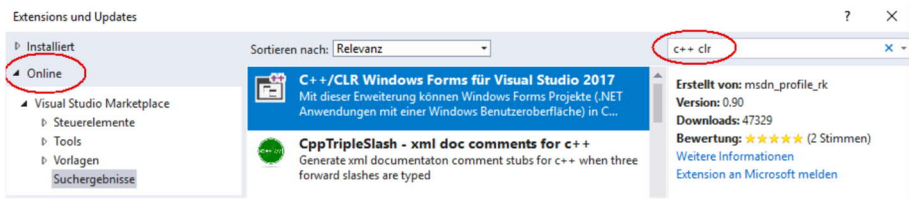
1.1 Installation von Visual Studio für Windows Forms Projekte

Damit Windows Forms Projekte in Visual Studio erstellt werden können, müssen bei der Installation von Visual Studio die Desktop-Entwicklung mit C++ und die C++/CLI-Unterstützung installiert werden. Falls das bei der Installation vergessen wurde, unter *Datei|Neu|Projekt* den Visual Studio Installer starten.



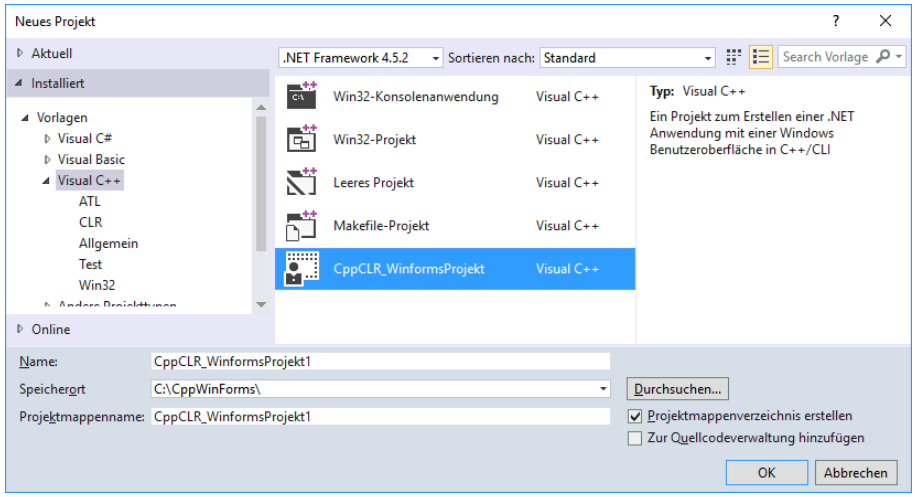
1.2 Eine Visual Studio Erweiterung für Windows Forms Projekte

In Visual Studio bis Version 2010 sind Vorlagen für Windows Forms Projekte vorhanden, aber nicht mehr ab Visual Studio 2012. Mit der unter *Extras|Optionen|Extensions und Updates*



verfügbaren VSIX-Extension wird dieser Mangel aber behoben.

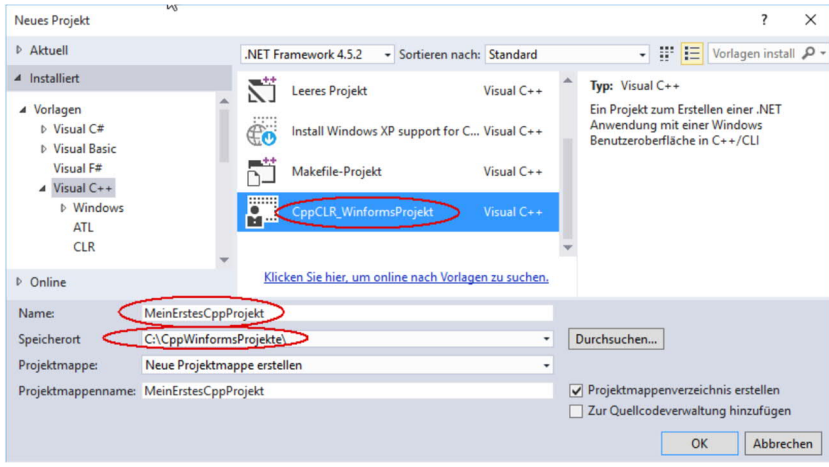
Nach der Installation dieser Extension bietet Visual Studio unter *Datei|Neu|Projekt* Windows Forms Projekte an:



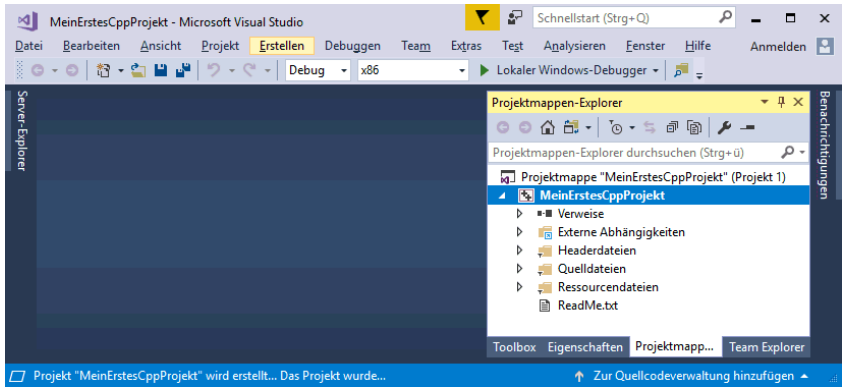
1.3 Visuelle Programmierung: Ein erstes kleines Programm

Nach diesen Vorbereitungen findet man in Visual Studio unter ***Datei|Neu|-Projekt|Installiert|Visual C++*** eine Projektvorlage für Windows Anwendungen mit einer graphischen Benutzeroberfläche (C++/CLR Windows Forms Anwendungen).

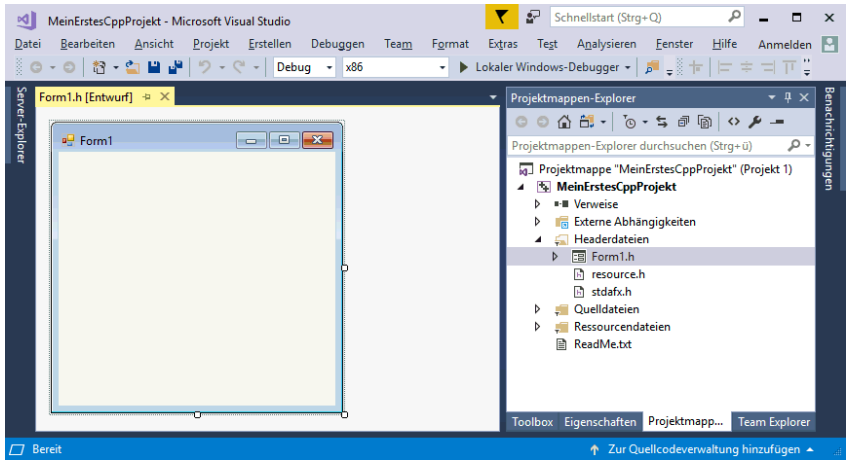
Ein Projekt für eine solche Anwendung erhält man, indem man nach *Name* einen Namen und nach *Speicherort* ein Verzeichnis für das Projekt eingibt und dann den OK-Button anklickt.



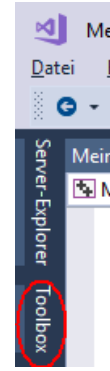
Nach dem Anklicken des OK-Buttons wird der Projektmappen-Explorer angezeigt:



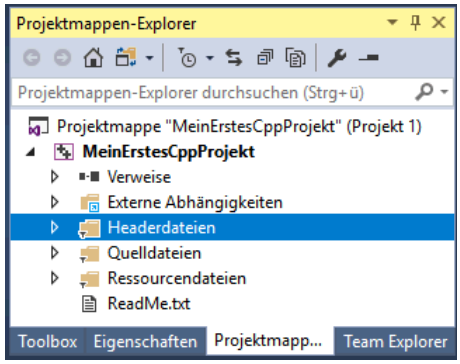
Nach dem Aufklappen der *Headerdateien* im Projektmappen-Explorer und einem Klick auf Form1.h erhält man etwa die folgende Ansicht:



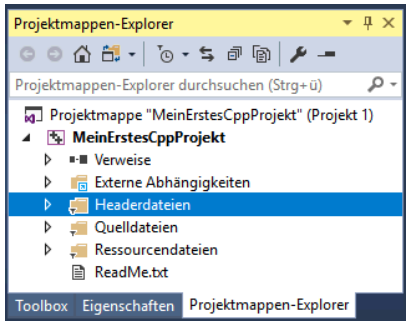
Die hier am rechten Rand angebotene **Toolbox** wird angezeigt, wenn man darauf klickt, oder mit *Ansicht|Toolbox*. In manchen Versionen von Visual Studio wird die Toolbox auch als Werkzeugkasten bezeichnet.



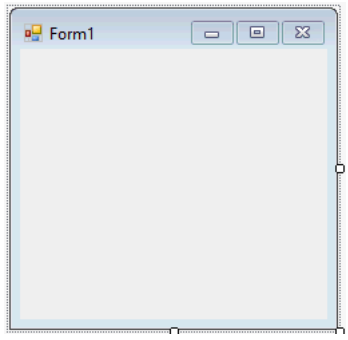
Damit das Formular und später auch der Editor etwas mehr Platz haben, ziehen Sie die Toolbox auf das Frame mit dem Projektmappen-Explorer (die linke Maustaste auf der Titelzeile des Eigenschaftenfensters drücken, dann mit gedrückter Maustaste auf die Titelzeile des Projektmappen-Explorers fahren und die Maustaste loslassen). Dann sieht dieses etwa folgendermaßen aus:



Wir werden hier zunächst nur die Toolbox, den Projektmappen-Explorer und das Eigenschaftenfenster benötigen. Falls hier noch weitere Fenster angezeigt werden (z.B. Klassenansicht, Eigenschaften-Manager, Team Explorer usw.), kann man diese auch löschen:



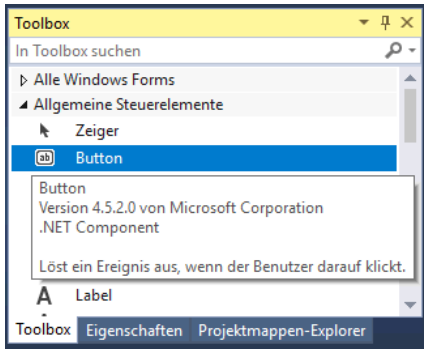
Das **Formular** (hier *Form1*) ist der Ausgangspunkt für alle Windows Forms Anwendungen. Es entspricht dem Fenster, das beim Start des Programms angezeigt wird:



Auf ein Formular kann man **Steuerelemente** (**Controls**) aus der **Toolbox** setzen. Die Toolbox wird nur angezeigt, wenn das Formular angezeigt wird. Er enthält praktisch alle unter Windows üblichen Steuerelemente. Diese sind auf verschiedene Gruppen verteilt (z.B. *Allgemeine Steuerelemente*, *Container* usw.), die auf- und zugeklappt werden können. Die

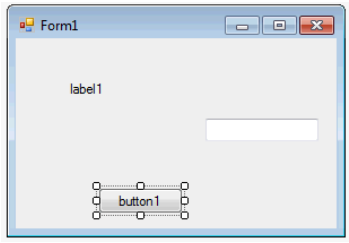
meisten dieser Steuerelemente (wie z.B. ein Button) werden im laufenden Programm auf dem Formular angezeigt.

Falls Ihnen die Namen und die kleinen Icons nicht allzu viel sagen, lassen Sie einfach den Mauszeiger kurz auf einer Zeile der Toolbox stehen: Dann erscheint ein kleines Hinweisfenster mit einer kurzen Beschreibung.



Um ein **Element** aus der Toolbox **auf das Formular** zu **setzen**, zieht man es einfach von der Toolbox auf das Formular. Oder man klickt mit der Maus zuerst auf die Toolbox-Zeile (sie wird dann als markiert dargestellt) und dann auf die Stelle im Formular, an die die linke obere Ecke kommen soll.

Beispiel: Nachdem man ein Label (Zeile sieben in *Allgemeine Steuerelemente*, mit dem großen *A*), eine TextBox (vierte Zeile von unten, Aufschrift *ab*) und einen Button (zweite Zeile mit der Aufschrift *ab*) auf das Formular gesetzt hat, sieht es etwa folgendermaßen aus:



Durch diese Spielereien haben Sie **schon ein richtiges Windows-Programm** erstellt – zwar kein besonders nützliches, aber immerhin. Sie können es folgendermaßen starten:

– mit *Debuggen|Debugging starten* von der Menüleiste, oder
– mit *F5* von einem beliebigen Fenster in Visual Studio oder
– durch den Aufruf der vom Compiler erzeugten Exe-Datei.

Dieses Programm hat schon viele Eigenschaften, die man von einem Windows-Programm erwartet: Man kann es mit der Maus verschieben, vergrößern, verkleinern und schließen.

Bemerkenswert an diesem Programm ist vor allem der im Vergleich zu einem nichtvisuellen Entwicklungssystem **geringe Aufwand**, mit dem es erstellt wurde. So braucht Petzold in

seinem Klassiker „Programmierung unter Windows“ (Petzold 1992, S. 33) ca. 80 Zeilen nichttriviale C-Anweisungen, um den Text „Hello Windows“ wie in einem Label in ein Fenster zu schreiben. Und in jeder dieser 80 Zeilen kann man einiges falsch machen.

Vergessen Sie nicht, Ihr **Programm** zu **beenden**, bevor Sie es weiterbearbeiten. Solange das Programm noch läuft können Sie weder den Compiler erneut starten noch das Formular verändern.

Diese Art der Programmierung bezeichnet man als **visuelle Programmierung**. Während man bei der konventionellen Programmierung ein Programm ausschließlich durch das Schreiben von Anweisungen (Text) in einer Programmiersprache entwickelt, wird es bei der visuellen Programmierung ganz oder teilweise aus vorgefertigten grafischen Steuerelementen zusammengesetzt.

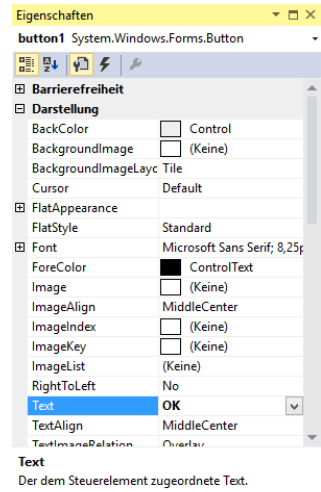
Mit Visual Studio kann die Benutzeroberfläche eines Programms visuell gestaltet werden. Damit sieht man bereits beim Entwurf des Programms, wie es später zur Laufzeit aussehen wird. Die Anweisungen, die als Reaktionen auf Benutzereingaben (Mausklicks usw.) erfolgen sollen, werden dagegen konventionell in einer Programmiersprache (z.B. C++**)** geschrieben.

1.4 Das Eigenschaftenfenster

Das zuletzt auf einem Formular (bzw. im Pull-down-Menü des Eigenschaftenfensters) angeklickte Steuerelement wird als das **aktuell ausgewählte Steuerelement** bezeichnet. Man erkennt es an den kleinen Quadraten an seinem Rand, den sogenannten **Ziehquadraten**. An ihnen kann man mit der Maus ziehen und so die Größe des Steuerelements verändern. Ein **Formular** wird dadurch zum aktuell ausgewählten Steuerelement, dass man mit der Maus eine freie Stelle im Formular anklickt.

Beispiel: Im letzten Beispiel ist *button1* das aktuell ausgewählte Steuerelement.

Im **Eigenschaftenfenster** (Kontextmenü des Steuerelements auf dem Formular, oder *Ansicht|Eigenschaftenfenster* – nicht mit *Ansicht|Eigenschaftenseiten* verwechseln) werden die Eigenschaften des aktuell ausgewählten Steuerelements angezeigt. In der linken Spalte stehen die **Namen** und in der rechten die **Werte** der Eigenschaften. Mit der Taste *F1* erhält man eine Beschreibung der Eigenschaft.



Ein Fenster (z.B. das Eigenschaftenfenster) kann man aus Visual Studio lösen, indem man seine Verankerung über das Kontextmenü (rechte Maustaste) der Titelzeile aufhebt:

Dann erhält man ein eigenständiges Fenster wie in der Abbildung rechts oben.

Den Wert einer Eigenschaft kann man über die rechte Spalte verändern. Bei manchen Eigenschaften kann man den neuen Wert über die Tastatur eintippen. Bei anderen wird nach dem Anklicken der rechten Spalte ein kleines Dreieck für ein Pull-down-Menü angezeigt, über das ein Wert ausgewählt werden kann. Oder es wird ein Symbol mit drei Punkten „…“ angezeigt, über das man Werte eingeben kann.

Beispiel: Bei der Eigenschaft ***Text*** kann man mit der Tastatur einen Text eingeben. Bei einem Button ist dieser Text die Aufschrift auf dem Button (z.B. „OK“), und bei einem Formular die Titelzeile (z.B. „Mein erstes C++-Programm“).
Bei der Eigenschaft ***BackColor*** (z.B. bei einem Button) kann man über ein Pull-down-Menü die **Hintergrundfarbe** auswählen.
Klickt man die rechte Spalte der Eigenschaft ***Font*** und danach das Symbol „…“ an, kann man die **Schriftart** der Eigenschaft *Text* auswählen.

Ein Steuerelement auf dem Formular wird nicht nur an seine Eigenschaften im Eigenschaftenfenster angepasst, sondern auch umgekehrt: Wenn man die Größe durch Ziehen an den Ziehquadraten auf dem Formular verändert, werden die Werte der entsprechenden Eigenschaften (*Location* und *Size* im Abschnitt *Layout*) im Eigenschaftenfenster automatisch aktualisiert.

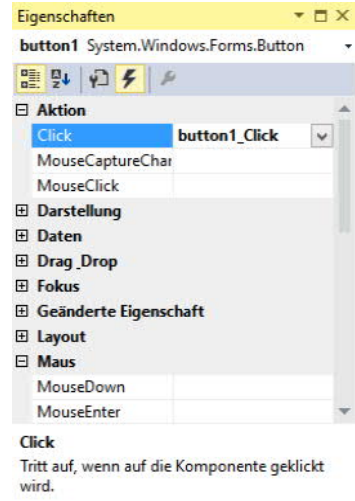
1.5 Erste Schritte in C++

Als nächstes soll das Programm aus dem letzten Abschnitt so erweitert werden, dass als Reaktion auf Benutzereingaben (z.B. beim Anklicken eines Buttons) Anweisungen ausgeführt werden.

Windows-Programme können Benutzereingaben in Form von Mausklicks oder Tastatureingaben entgegennehmen. Im Unterschied zu einfachen Konsolen-Programmen (z.B. DOS-Programmen) muss man in einem Windows-Programm aber keine speziellen Funktionen (wie z.B. *scanf* in C) aufrufen, die auf solche Eingaben warten. Stattdessen werden alle Eingaben von Windows zentral entgegengenommen und als sogenannte Botschaften (Messages) an das entsprechende Programm weitergegeben. Dadurch wird in diesem Programm ein sogenanntes **Ereignis** ausgelöst.

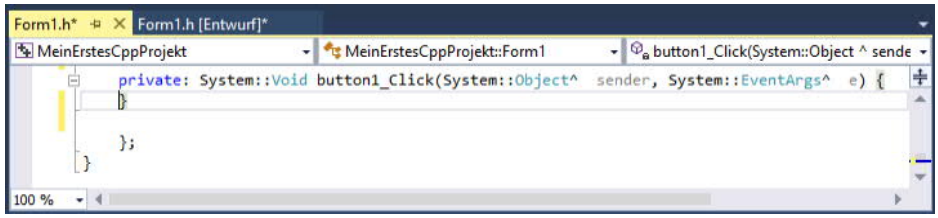
Die Ereignisse, die für das aktuell ausgewählte Steuerelement eintreten können, werden nach dem Anklicken des Symbols für die **Ereignisse** im Eigenschaftenfenster angezeigt.

Die Abbildung rechts zeigt einige Ereignisse für einen Button. Dabei steht *Click* für das Ereignis, das beim Anklicken des Buttons eintritt. Klappt man die Gruppen auf, sieht man, dass ein Button nicht nur auf das Anklicken reagieren kann, sondern auch noch auf zahlreiche andere Ereignisse.



Einem solchen Ereignis kann eine Funktion zugeordnet werden, die dann aufgerufen wird, wenn das Ereignis eintritt. Diese Funktion wird auch als **Ereignisbehandlungsroutine** (engl. **event handler**) bezeichnet. Sie wird von Visual Studio durch einen Doppelklick auf die Zeile des Ereignisses erzeugt und im **Quelltexteditor** angezeigt. Der Cursor steht dann am Anfang der Funktion.

Vorläufig soll unser Programm allerdings nur auf das Anklicken eines Buttons reagieren. Die bei diesem Ereignis aufgerufene Funktion erhält man am einfachsten durch einen Doppelklick auf den Button im Formular. Dadurch erzeugt Visual Studio die folgende Funktion und zeigt sie im **Editor** an:

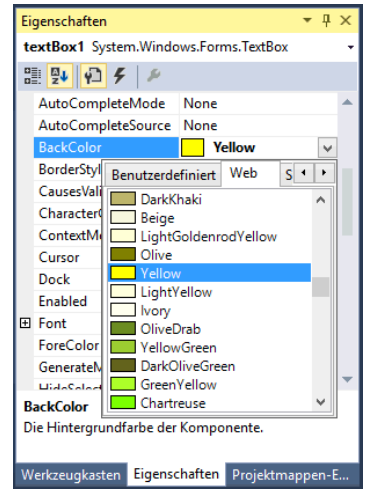


Zwischen die geschweiften Klammern „{“ und „}“ schreibt man dann die **Anweisungen**, die ausgeführt werden sollen, wenn das Ereignis *Click* eintritt.

Welche Anweisungen hier möglich sind und wie diese aufgebaut werden müssen, ist der Hauptgegenstand dieses Buches und wird ab Kapitel 3 ausführlich beschrieben. Im Rahmen dieses einführenden Kapitels sollen nur einige wenige Anweisungen vorgestellt werden und diese auch nur so weit, wie das zum Grundverständnis von Visual Studio notwendig ist. Falls Ihnen Begriffe wie „Variablen“ usw. neu sind, lesen Sie trotzdem weiter – aus dem Zusammenhang erhalten Sie sicherlich eine intuitive Vorstellung, die zunächst ausreicht. Später werden diese Begriffe dann genauer erklärt.

Eine beim Programmieren häufig verwendete Anweisung ist die **Zuweisung** (mit dem Operator „=“), mit der man einer Variablen einen Wert zuweisen kann. Als Variablen sollen zunächst nur solche Eigenschaften von Steuerelementen verwendet werden, die auch im Eigenschaftenfenster angezeigt werden. Diesen Variablen können dann die Werte zugewiesen werden, die auch im Eigenschaftenfenster in der rechten Spalte der Eigenschaften angeboten werden.

In der Abbildung rechts sieht man einige zulässige Werte für die Eigenschaft *BackColor*. Sie werden nach dem Aufklappen des Pull-down-Menüs angezeigt.



Schreibt man jetzt die Anweisung

```
textBox1->BackColor = Color::Yellow;
```

zwischen die geschweiften Klammern

```
private: System::Void
button1_Click(System::Object^  sender, System::EventArgs^  e)
{
  textBox1->BackColor = Color::Yellow;
}
```

erhält die Eigenschaft *BackColor* von *textBox1* beim Anklicken von *button1* während der Ausführung des Programms den Wert *Color::Yellow*, der für die Farbe Gelb steht. Wenn Sie das Programm jetzt mit *F5* starten und dann *button1* anklicken, erhält die TextBox tatsächlich die Hintergrundfarbe Gelb.

Auch wenn dieses Programm noch nicht viel sinnvoller ist als das erste, haben Sie doch gesehen, wie mit Visual Studio Anwendungen für Windows entwickelt werden. Dieser **Entwicklungsprozess** besteht immer aus den folgenden Aktivitäten:

1. Man gestaltet die Benutzeroberfläche, indem man Steuerelemente aus der Toolbox auf das Formular setzt (drag and drop) und ihre Eigenschaften im Eigenschaftenfenster oder das Layout mit der Maus anpasst (visuelle Programmierung).
2. Man schreibt in C++ die Anweisungen, die als Reaktion auf Benutzereingaben erfolgen sollen (nichtvisuelle Programmierung).
3. Man startet das Programm und testet, ob es sich auch wirklich so verhält, wie es sich verhalten soll.

Der Zeitraum der Programmentwicklung (Aktivitäten 1. und 2.) wird auch als **Entwurfszeit** bezeichnet. Im Unterschied dazu bezeichnet man die Zeit, während der ein Programm läuft, als **Laufzeit** eines Programms.