

C++ für IT-Berufe

```
#include <iostream>
using namespace std;

int main()
{
    cout << "Informationsteil:" << endl;
    cout << "    - Standard-C++" << endl;
    cout << "    - C++ unter WinRT:" << endl;
    cout << "        ~ C++11/CX Grundlagen" << endl;
    cout << "        ~ Windows Store-Apps" << endl;
    cout << endl;
    cout << "Aufgabenpool" << endl;
    cout << endl;
    cout << "Lernsituationen" << endl;
    cout << endl;
    cout << endl;
    return 0;
}
```

2. Auflage

VERLAG EUROPA-LEHRMITTEL · Nourney, Vollmer GmbH & Co. KG
Düsselberger Str. 23 42781 Haan-Gruiten

Europa-Nr.: 85498

Verfasser:

Dirk Hardy, 46049 Oberhausen

Verlagslektorat:

Alexander Barth

2. Auflage 2013

Druck 5 4 3

Alle Drucke derselben Auflage sind parallel einsetzbar, da sie bis auf die Behebung von Druckfehlern untereinander unverändert sind.

ISBN 978-3-8085-8550-4

Alle Rechte vorbehalten. Das Werk ist urheberrechtlich geschützt. Jede Verwertung außerhalb der gesetzlich geregelten Fälle muss vom Verlag schriftlich genehmigt werden.

© 2013 by Verlag Europa-Lehrmittel, Nourney, Vollmer GmbH & Co. KG, 42781 Haan-Gruiten

<http://www.europa-lehrmittel.de>

Satz: Reemers Publishing Services GmbH, Krefeld

Umschlag: braunwerbeagentur, 42477 Radevormwald

Umschlagfotos: mathtilda-fotolia.com, Gina Sanders-fotolia.com

Druck: Medienhaus Plump, 53619 Rheinbreitbach

Vorwort

Die Entwicklung von Anwendungen geht heutzutage weit über das reine Programmieren hinaus. Die Anforderungen an den **Fachinformatiker** bzw. Informatiker mit der **Fachrichtung Anwendungs-entwicklung** sind sehr komplex geworden. Es reicht nicht mehr aus, mit einer speziellen Programmiersprache zu arbeiten. Vielmehr sind umfangreiche Kenntnisse in mehreren Programmiersprachen, in Datenbankmanagementsystemen und weiteren Gebieten wie der Client-Server-Programmierung, Web-Programmierung oder auch App-Programmierung nötig. Das Gemeinsame dieser verschiedenen Aspekte ist die Objektorientierung bzw. die Hinwendung zur Objektorientierung. Die modernen Programmiersprachen sind objektorientiert, die Datenbankmanagementsysteme werden in Zukunft verstärkt objektorientiert sein. Mischformen wie objektrelationale Datenbanken sind bereits im Einsatz. Eine gute Grundlage für diese Anforderungen ist die Erlernung der Sprache C++. Die Sprache ist plattformunabhängig und objektorientiert. Sie ist schwerer zu erlernen als andere Sprachen, bietet allerdings dafür einige Vorteile:

- Geschwindigkeit
- Weite Verbreitung
- Sehr viele Tools und Bibliotheken
- Compiler und Entwicklungswerkzeuge auf unterschiedlichen Plattformen
- Grundlage vieler weiterer Sprachen wie Java, Javascript, Perl oder auch PHP und C#

Die letzte ISO-Standardisierung der Sprache C++ fand 2011 statt. Damit verfügt C++ über viele neue Konzepte, die in anderen modernen Programmiersprachen (wie C#) bereits umgesetzt worden sind. Beispielsweise kann **C++11** (so die offizielle Bezeichnung) nun auch mit *Lambda-Ausdrücken* (einem Konzept der anonymen Funktionen) umgehen. Die vorliegende zweite Auflage dieses Buches nimmt diese Neuerungen in einem eigenen Kapitel auf. Darauf aufbauend bietet das Buch auch einen Einstieg in die komponentenbasierte Erweiterung C++/CX sowie die **App-Programmierung** unter Windows 8.

Den Lesern dieses Buches bleibt viel Erfolg bei der Erlernung von C++ zu wünschen und mit einem Zitat des Erfinders von C++ (Bjarne Stroustrup) zu schließen: „Die Frage *Wie schreibt man ein gutes C++ -Programm?* hat sehr viel Ähnlichkeit mit der Frage *Wie schreibt man gute englische Prosa?* Darauf gibt es zwei Antworten: Wisse, was Du sagen willst und Übe. Halte Dich an gute Vorbilder. Beide Antworten gelten für C++ ebenso wie für Englisch – und sind ebenso schwer zu befolgen.“

Für Anregungen und Kritik zu diesem Buch sind wir Ihnen dankbar (gerne auch per E-Mail).

Dirk Hardy
E-Mail: Hardy@DirkHardy.de

Im September 2013

Verlag Europa-Lehrmittel
E-Mail: Info@Europa-Lehrmittel.de

Aufbau des Buches

Das vorliegende Buch möchte die Sprache C++ möglichst anschaulich, praxis- und unterrichtsnah vermitteln. Damit verfolgt dieses Buch einen **praktischen Ansatz**. Es ist die Ansicht des Autors, dass gerade in der schulischen Ausbildung der Zugang zu den komplexen Themen der Programmierung verstärkt durch anschauliche und praktische Umsetzung vorbereitet werden muss. Anschließend können allgemeine und komplexe Aspekte der Programmierung oder auch der Softwareentwicklung besser verstanden und umgesetzt werden.

Das Buch ist in **drei Teile** getrennt. Der **erste Teil** des Buches dient als **Informationsteil** und bietet eine **systematische Einführung in die Sprache C++**. Es werden sowohl die strukturierte Programmierung als auch die objektorientierte Programmierung behandelt.

Ein ausführlicher Einstieg in die **App-Programmierung** rundet den Informationsteil ab. Dabei werden zuerst die Spracherweiterung C++11/CX für die Programmierung der Windows Runtime-API und dann die Grundlagen der Microsoft Store-Apps betrachtet.

Der **zweite Teil** des Buches ist eine **Sammlung von Übungsaufgaben**. Nach der Erarbeitung der entsprechenden Kenntnisse aus dem Informationsteil können die Aufgaben aus diesem Teil zur weiteren Auseinandersetzung mit den Themen dienen und durch verschiedene Schwierigkeitsgrade auch die Differenzierung im Unterricht ermöglichen.

Der **dritte Teil** des Buches beinhaltet **Lernsituationen** basierend auf dem Lernfeld Entwickeln und Bereitstellen von Anwendungssystemen aus dem Rahmenlehrplan für die IT-Berufe (speziell Fachinformatiker-Anwendungsentwicklung). Lernsituationen konkretisieren sich aus den Lernfeldern und sollen im Idealfall vollständige Handlungen darstellen (Planen, Durchführen, Kontrollieren). Aus diesem Grund werden die Lernsituationen so angelegt, dass neben einer Planungsphase nicht nur die Durchführung (Implementation des Programms) im Blickpunkt steht, sondern auch geeignete Testverfahren zur Kontrolle des Programms bzw. des Entwicklungsprozesses in die Betrachtung einbezogen werden. Die Lernsituationen können aber auch als **Projektideen** verstanden werden.

Das Buch ist für alle berufsbezogenen Ausbildungsgänge im IT-Bereich konzipiert. Durch die differenzierten Aufgabenstellungen kann es in allen IT-Berufen (speziell Fachinformatiker), aber auch von den informationstechnischen Assistenten genutzt werden.

Als Entwicklungswerkzeuge werden in diesem Buch die *Express-Editionen Visual Studio 2012 für Desktop sowie Visual Studio 2012 für Windows 8* von Microsoft genutzt. Diese Entwicklungsumgebungen sind kostenfrei als Download im Internet verfügbar. Die Links zu der benutzten Software und alle Quelltexte des Buches finden Sie ständig aktualisiert auf der Seite <http://www.dirkhardy.de> unter der Rubrik „Programmierbücher“.

Vorwort	3
Aufbau des Buches.....	4
Teil 1 Einführung in C++	11
1 Einführung in C++	13
1.1 Historische Entwicklung der Sprache C++.....	13
1.1.1 Von C zu C++.....	13
1.1.2 Prozedurale, strukturierte und objektorientierte Programmierung.....	13
1.1.3 Kleiner Stammbaum der Programmiersprachen	15
1.2 Bestandteile eines C++-Programms.....	15
1.2.1 Was ist ein Programm?.....	15
1.2.2 C++-Quellcode.....	15
1.2.3 Grundsätzlicher Aufbau eines C++-Programmes	16
1.3 Compiler, Linker und Bibliotheken	17
1.3.1 Der Compiler	17
1.3.2 Der Linker	17
1.3.3 Bibliotheken	18
1.3.4 Schematischer Ablauf einer Programmerstellung.....	18
2 Das erste C++-Programm.....	20
2.1 Ausgabe auf dem Bildschirm	20
2.1.1 Ein C++-Projekt in Visual Studio 2012 anlegen.....	20
2.1.2 Bibliothek für die Ausgabe.....	23
2.1.3 Das erste Programm.....	24
2.1.4 Erste Ausgabe auf dem Bildschirm.....	24
2.2 Grundlegende Konventionen in C++	25
2.2.1 Schlüsselworte in C++	25
2.2.2 Bezeichner (Namen) in C++	25
2.2.3 Trennzeichen	26
2.2.4 Kommentare	26
2.3 Datentypen und Variablen	27
2.3.1 Variablen in C++	27
2.3.2 Elementare Datentypen.....	28
2.3.3 Operationen auf den elementaren Datentypen.....	30
2.3.4 Anwendungsbeispiel von Variablen	31
3 Ein- und Ausgabe in C++	32
3.1 Ausgabe mit cout.....	32
3.1.1 Das Objekt cout	32
3.1.2 Ausgabe von Sonderzeichen.....	33
3.1.3 Manipulatoren	34
3.2 Eingabe mit cin	35
3.2.1 Das Objekt cin	35
3.2.2 Der Streamstatus.....	36
4 Operatoren in C++	38
4.1 Arithmetische Operatoren	38
4.1.1 Elementare Datentypen und ihre arithmetischen Operatoren	38
4.1.2 Der Modulo-Operator	39
4.1.3 Inkrement- und Dekrementoperatoren.....	39
4.2 Relationale und logische Operatoren.....	40
4.2.1 Relationale Operatoren.....	40
4.2.2 Logische Operatoren.....	41
4.3 Bit-Operatoren und weitere Operatoren.....	42
4.3.1 Bit-Operatoren	42
4.3.2 Die Bit-Schiebeoperatoren << und >>.....	43
4.3.3 Typumwandlung mit cast-Operatoren.....	44
4.3.4 Der sizeof-Operator	44
4.3.5 Zuweisung und gekoppelte Zuweisung.....	45

4.4	Prioritäten von Operatoren	45
4.4.1	Rang von Operatoren	45
5	Selektion und Iteration	48
5.1	Die Selektion	48
5.1.1	Darstellung der Selektion mit einem Programmablaufplan	48
5.1.2	Die einseitige Selektion mit der if-Anweisung	49
5.1.3	Die zweiseitige Selektion mit der if-else-Anweisung	49
5.1.4	Verschachtelte Selektionen mit if und if-else	50
5.1.5	Mehrfachselektion mit switch.....	51
5.2	Kopf-, fuß- und zählergesteuerte Iterationen	53
5.2.1	Die do-while-Schleife.....	54
5.2.2	Die while-Schleife.....	55
5.2.3	Die for-Schleife.....	55
5.2.4	Abbruch und Sprung in einer Schleife	57
6	Funktionen in C++	58
6.1	Entwicklung des Funktionsbegriffs.....	58
6.1.1	Wiederkehrende Programmabschnitte.....	58
6.1.2	Übergabe von Werten.....	59
6.1.3	Rückgabe eines Wertes.....	60
6.1.4	Funktionen in Funktionen aufrufen.....	61
6.1.5	Zusammenfassung der Aspekte aus 6.1	62
6.2	Aufbau der Funktionen in C++	62
6.2.1	Deklaration einer Funktion	62
6.2.2	Definition einer Funktion.....	63
6.2.3	Lokale und globale Variablen.....	65
6.2.4	Call by value	66
6.2.5	Überladen von Funktionen	67
6.2.6	Default-Argumente für Funktionen	68
6.2.7	Rekursive Funktionen.....	68
6.3	Modularer Programmaufbau	70
6.3.1	Schnittstelle und Implementation.....	71
6.3.2	Umsetzung in C++	71
6.3.3	Namensräume	72
6.3.4	Der Präprozessor	74
6.3.5	Regeln zur modularen Programmgestaltung.....	76
7	Arrays	77
7.1	Ein- und mehrdimensionale Arrays	78
7.1.1	Eindimensionale Arrays.....	78
7.1.2	Mehrdimensionale Arrays	79
7.1.3	Übergabe von Arrays an Funktionen.....	81
7.2	Zeichenketten in C++	83
7.2.1	Arrays vom Typ char	83
7.2.2	Funktionen zur Zeichenkettenbearbeitung	84
7.3	Sortieren von Arrays	85
7.3.1	Sortieren durch Auswahl	86
7.3.2	Der Bubblesort.....	88
8	Zeiger	91
8.1	Zeigervariablen	91
8.1.1	Deklaration eines Zeigers.....	91
8.1.2	Der Adressoperator	91
8.1.3	Der Dereferenzierungsoperator	92
8.2	Anwendungen von Zeigervariablen.....	93
8.2.1	Der call by reference	93
8.2.2	Zeiger und Arrays.....	94
8.2.3	Zeigerarithmetik.....	95
8.2.4	Zeiger auf Funktionen	97
8.2.5	Arrays von Zeigern	99
8.2.6	Dynamische Speicherreservierung	99

8.3	Die Referenz	103
8.3.1	Der Referenzoperator	103
8.3.2	Anwendung des Referenzoperators	104
9	Strukturen	105
9.1	Die Struktur in C++	105
9.1.1	Deklaration einer Struktur	105
9.1.2	Zugriff mit Operatoren	106
9.1.3	Strukturen in Strukturen	107
9.1.4	Arrays von Strukturen	108
9.2	Höhere Datenstrukturen	109
9.2.1	Die doppelt verkettete Liste	110
9.2.2	Der Binärbaum	112
10	Das Klassenkonzept in C++	115
10.1	Die Klasse in C++	117
10.1.1	Aufbau einer Klasse in C++	117
10.1.2	Die Konstruktoren einer Klasse	119
10.1.3	Der Destruktor einer Klasse	122
10.1.4	Get- und Set-Methoden	123
10.2	Dynamische Speicherreservierung in Klassen	125
10.2.1	Die Klasse CKette	125
10.2.2	Call by value und der Copy-Konstruktor	127
10.3	Weitere Elemente einer Klasse	128
10.3.1	Der this-Zeiger	128
10.3.2	Statische Klassenelemente	130
10.4	Deklaration und Implementation bei Klassen	131
10.4.1	Header- und cpp-Datei	131
11	Dateioperationen	132
11.1	Ein- und Ausgabeströme	133
11.1.1	Eine Datei im Textmodus öffnen	133
11.1.2	Fehler bei Dateioperationen	135
11.1.3	Methoden der FileStream-Klassen	136
11.1.4	Eine Datei im Binärmodus öffnen	139
11.2	Wahlfreier Zugriff in Dateien	140
11.2.1	Positionieren des Dateizeigers	141
11.2.2	Lesen der Dateizeiger-Position	142
12	Das Überladen von Operatoren	143
12.1	Globale überladene Operator-Funktion	143
12.1.1	Die globale Operator-Funktion	143
12.1.2	Addition von Zeichenketten	144
12.1.3	Weitere Beispiele für globale Operator-Funktionen	146
12.2	Überladene Operatorfunktion als Methode	147
12.2.1	Überladener Operator als Methode	147
12.2.2	Addition von Zeichenketten	148
12.2.3	Weitere Beispiele für überladene Operatoren als Methoden	148
12.3	Überladen der Ein- und Ausgabeoperatoren	149
12.3.1	Überladene Operatoren der iostream-Klassen	149
12.3.2	Überladen des Eingabeoperators für eigene Klassen	150
12.3.3	Überladen des Ausgabeoperators für eigene Klassen	151
13	Vererbungskonzept in C++	152
13.1	Die einfache Vererbung	152
13.1.1	Umsetzung einer einfachen Vererbung in C++	153
13.1.2	Attribute als protected deklarieren	154
13.1.3	Aufruf der Basisklassenkonstruktoren	154
13.1.4	Weitere Formen der Vererbung	155
13.2	Die Mehrfachvererbung	157
13.2.1	Umsetzung der Mehrfachvererbung in C++	157

13.2.2	Virtuelle Vererbung	158
13.2.3	Aufruf der Basisklassenkonstruktoren	158
14	Polymorphismus und virtuelle Methoden	159
14.1	Zuweisungen innerhalb einer Vererbungshierarchie.....	159
14.1.1	Zuweisung von Objekten	159
14.1.2	Zeiger auf Basisklassen	160
14.2	Polymorphismus	160
14.2.1	Virtuelle Methoden.....	161
14.2.2	Regeln im Umgang mit virtuellen Methoden.....	162
14.2.3	Arrays von Basisklassenzeigern.....	162
14.2.4	Abstrakte Basisklassen.....	163
15	Fortgeschrittene Programmierung in C++	164
15.1	Templates	164
15.1.1	Funktionen-Templates.....	164
15.1.2	Klassen-Templates.....	165
15.2	Ausnahmen – Exceptions.....	166
15.2.1	Versuchen und Werfen – try und throw.....	167
15.2.2	Auffangen – catch	167
15.3	Die C++-Standardbibliothek	170
15.3.1	Die Klasse string	170
15.3.2	Die Klasse map.....	173
16	C++ und Windows Store Apps	175
16.1	Applikationen unter Windows 8	175
16.1.1	Neue Windows-Konzeption	175
16.1.2	Windows Runtime.....	175
16.1.3	Aufbau einer Windows Store App	176
16.2	C++11.....	178
16.2.1	Der neue Standard C++11	178
16.2.2	Der auto-Typ.....	178
16.2.3	Intelligente Zeiger	178
16.2.4	Lambda-Ausdrücke.....	180
16.2.5	Die neue for-Schleife	182
16.3	Die Komponenten-Erweiterung C++/CX.....	183
16.3.1	C++ und die Windows Runtime.....	183
16.3.2	Die Erweiterungen von C++ im Überblick.....	183
16.3.3	Neue Datentypen in C++/CX	185
16.3.4	Verweisklassen in C++/CX	185
16.3.5	Wertklassen in C++/CX.....	187
16.3.6	Boxing und Unboxing.....	187
16.3.7	Properties in C++/CX	188
16.3.8	Arrays in C++/CX	189
16.3.9	Vererbung in C++/CX.....	190
16.3.10	Schnittstellen in C++/CX	192
16.4	XAML.....	193
16.4.1	Extensible Application Markup Language XAML	193
16.4.2	XAML-Steuerlemente	194
16.4.3	Layout-Container	195
16.4.4	Eigenschaften von XAML im Überblick	196
17	Windows Store App-Entwicklung	197
17.1	Die erste Windows Store App	197
17.1.1	Ein leeres App-Projekt anlegen.....	197
17.1.2	Den Werkzeugkasten nutzen	200
17.1.3	Auf Ereignisse reagieren	201
17.2	Einfache Steuerelemente	202
17.2.1	Textboxen, Buttons, RadioButtons und Checkboxes	202
17.2.2	Listboxen und Kombinationsboxen	204
17.2.3	Flexibles Steuerelement FlipView	206

17.3	Eine geteilte Windows Store App.....	208
17.3.1	Ein geteiltes App-Projekt anlegen	208
17.3.2	Das Datenmodell der App	209
17.3.3	Ein Beispiel einer geteilten App	210
17.4	Eine Windows Store App mit DirectX und XAML	211
17.4.1	DirectX und Windows Store Apps.....	211
17.4.2	Eine Direct2D-App (XAML) mit Textausgabe	212
17.4.3	Eigenen Text mit einer Direct2D-App (XAML) ausgeben	216
17.5	Weitere Konzepte	217
17.5.1	Weitergeleitete Ereignisse (Routed Events).....	217
17.5.2	Datenbindungen.....	218
17.5.3	Zugriff auf Ressourcen mit asynchronen Methoden	219

Teil 2 Aufgabenpool 222

1	Aufgaben zum Umfeld der Sprache C++	223
2	Aufgaben zum ersten Programm in C++.....	223
3	Aufgaben zur Ein- und Ausgabe	224
4	Aufgaben zu Operatoren.....	225
5	Aufgaben zur Selektion und Iteration	227
6	Aufgaben zu Funktionen in C++	231
7	Aufgaben zu Arrays in C++.....	233
8	Aufgaben zu Zeigern.....	236
9	Aufgaben zu Strukturen	239
10	Aufgaben zu Klassen in C++	242
11	Aufgaben zu Dateioperationen mit C++	245
12	Aufgaben zur Überladung von Operatoren.....	247
13	Aufgaben zur Vererbung in C++	251
14	Aufgaben zu virtuellen Methoden	253
15	Aufgaben zur fortgeschrittenen Programmierung.....	254
16	Aufgaben zu C++ und Windows Store Apps	256
17	Aufgaben zu Windows Store App-Entwicklung	258

Teil 3 Lernsituationen 262

Lernsituation 1:	Erstellen einer Präsentation mit Hintergrundinformationen zu der Sprache C++ (in Deutsch oder Englisch).....	263
Lernsituation 2:	Anfertigen einer Kundendokumentation für den Einsatz einer Entwicklungsumgebung in C++ (in Deutsch oder Englisch)	264
Lernsituation 3:	Entwicklung eines Verschlüsselungsverfahrens für ein internes Memo-System der Support-Abteilung einer Netzwerk-Firma.....	266
Lernsituation 4:	Entwicklung eines Informationstools für die Anzeige von Umgebungsvariablen.....	267
Lernsituation 5:	Planung, Implementierung und Auswertung eines elektronischen Fragebogens.....	269
Lernsituation 6:	Realisierung einer Klasse zur Speicherung von Messwerten	272
Lernsituation 7:	Implementierung einer Klasse zur Simulation der echten Bruchrechnung.....	274
Lernsituation 8:	Entwicklung einer Grafik-App für einfache Skizzen	276

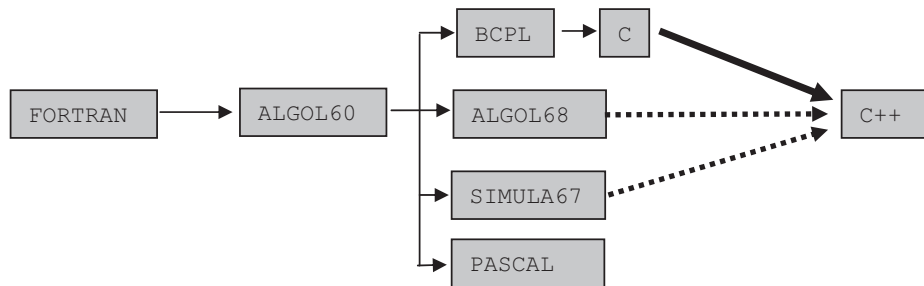
Anhang A: Strukturierte Dokumentationstechniken..... 279

Der Programmablaufplan (PAP)	279
Beispiel eines Programmablaufplanes	280
Das Struktogramm (auch Nassi-Shneiderman-Diagramm).....	281
Beispiel eines Struktogrammes	282

Anhang B: Index 283

1.1.3 Kleiner Stammbaum der Programmiersprachen

1950.....1960.....1970.....1980



C++ im Überblick:

- ▶ Entwickelt von Bjarne Stroustrup in den 80er-Jahren.
- ▶ Basiert auf der Sprache C und hat Konzepte der Sprachen SIMULA67 und ALGOL68 übernommen.
- ▶ C++ ist eine objektorientierte Sprache.
- ▶ Sie ist plattformunabhängig, systemnah und sehr schnell.
- ▶ Viele Betriebssysteme wurden in C++ programmiert.
- ▶ Die Sprache ist sehr weit verbreitet und kommt in technischen, kaufmännischen und wissenschaftlichen Anwendungen zum Einsatz.

1.2 Bestandteile eines C++-Programms

1.2.1 Was ist ein Programm?

Ein Programm besteht aus einer Folge von Anweisungen an den Computer. Diese Anweisungen (Befehle) werden in einer bestimmten Form gegeben – und zwar in einer speziellen Sprache (Programmiersprache).

Diese Programmiersprache wird vom Computer nicht direkt verstanden, sondern muss erst „übersetzt“ werden, und zwar in eine für den Computer verständliche Sprache, den **Maschinencode**.

Nach der Übersetzung kann das Programm gestartet werden.

```

Anweisung 1;
Anweisung 2;
Anweisung 3;
Anweisung 4;
Anweisung 5;
:
:
:
Anweisung N;
  
```

Eine endliche Folge von eindeutigen Anweisungen an den Computer nennt man **Algorithmus**.

1.2.2 C++-Quellcode

Die Anweisungen an den Computer in der Sprache C++ werden in Dateien gespeichert. Der Inhalt dieser Dateien wird als **Quellcode** bezeichnet.

Normalerweise haben C++-Quellcode-Dateien die Endungen

- ▶ `cpp` für Cplusplus
- ▶ `h` bzw. `hpp` für Header bzw. Header-plusplus.

2 Das erste C++-Programm

2.1 Ausgabe auf dem Bildschirm

Für die Ausgabe auf dem Bildschirm sind einige Vorbereitungen zu treffen.

Neben dem Kennenlernen der Entwicklungsumgebung, damit überhaupt ein Programm geschrieben werden kann, müssen noch weitere Punkte geklärt werden:

- Wie werden ein C++-Projekt und eine Quellcode-Datei angelegt?
- Welche Bibliothek muss für die Ausgabe auf dem Bildschirm eingebunden sein?
- Wie sieht das „Hauptprogramm“ aus?
- Mit welcher Anweisung wird eine Bildschirmausgabe erreicht?

Die folgenden Kapitel geben Aufschluss über diese Punkte.

2.1.1 Ein C++-Projekt in Visual Studio 2012 anlegen

Die integrierte Entwicklungsumgebung *Visual Studio 2012 für Desktop* ist eine komfortable Umgebung, um C++-Programme zu entwickeln. Besonders erfreulich ist der Umstand, dass die Umgebung kostenfrei als *Express Edition* im Internet bereit steht. Ein C++-Programm besteht aus einer oder mehreren Quellcodedateien. Diese Dateien werden in einem Projekt organisiert.

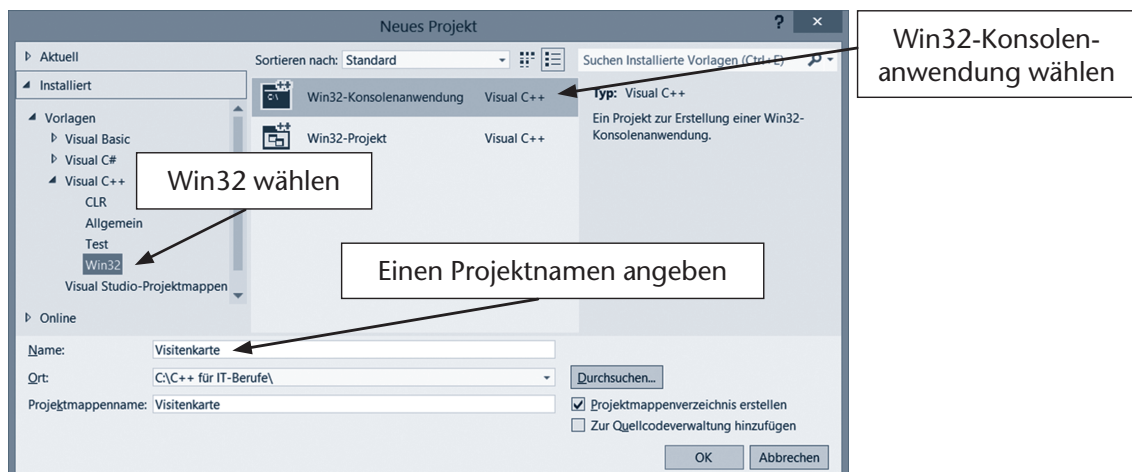
Visual C++ unterscheidet im Prinzip vier verschiedene Projektarten:

- Win32-Projekt (eine Windows-Anwendung)
- **Win32-Konsolenanwendung (ähnlich einem DOS-Programm)**
- CLR-Konsolenanwendung (die Konsolenanwendung unter .NET)
- Klassenbibliothek (Sammlung von Klassen unter .NET)

In diesem Buch ist hauptsächlich eine Projektform von Bedeutung: **die Win32-Konsolenanwendung**. Diese Form ist ausreichend, um die Programme in C++ auf dem Bildschirm darzustellen. In den späteren Kapiteln wird dann ein Umstieg auf eine weitere Entwicklungsumgebung nötig sein, die allerdings nur unter Windows 8 lauffähig ist. Die Konsolenanwendung ist natürlich nicht so ansprechend wie ein Windows-Programm oder eine App, aber um die Sprache C++ zu lernen völlig ausreichend.

Anlegen eines neuen Projektes:

- Starten Sie *Visual Studio Express 2012 für Desktop*
- Wählen Sie den Menüpunkt *Datei → Neues Projekt*



5 Selektion und Iteration

Die Selektion (Auswahl) und die Iteration (Wiederholung) sind zwei sehr wichtige Konstrukte in einer Programmiersprache. In den vorherigen Kapiteln sind schon einige Probleme ohne diese Konstrukte gelöst worden, aber große und immer komplexer werdende Programme können ohne Selektion und Iteration nicht auskommen.

5.1 Die Selektion

5.1.1 Darstellung der Selektion mit einem Programmablaufplan

Problemstellung:

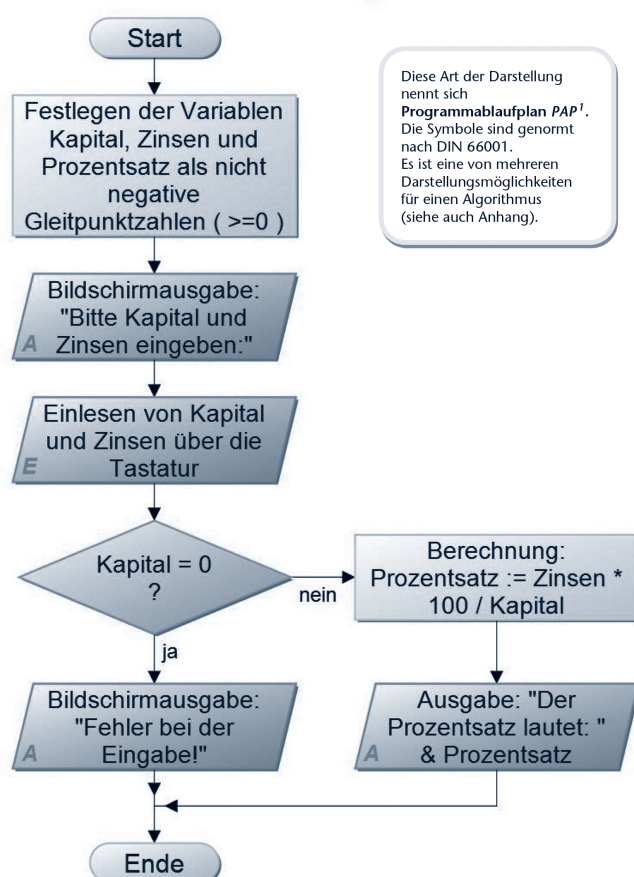
Es soll eine Berechnung des Prozentsatzes bei gegebenem Kapital und Zinsen durchgeführt werden.

$$\text{Formel: } p = \frac{Z * 100}{K}$$

Das Programm kann einen Fehler verursachen, wenn für das Kapital der Wert Null eingegeben wird (die Division durch Null ist verboten).

Lösungsmöglichkeit:

Das Programm erkennt, ob eine Null eingegeben wurde, und führt dann keine Berechnung durch. Das Problem wird durch eine Dokumentationstechnik, den Programmablaufplan, zuerst schematisch erfasst. Anschließend wird dann auf die Umsetzung in C++ eingegangen.



¹ Der Programmablaufplan wurde mit dem Programm *PapDesigner* erstellt. Diese Software wurde speziell für die Ausbildung im IT-Bereich entwickelt und steht kostenfrei im Internet zum Download bereit.

Die Vorstellung eines zweidimensionalen Arrays entspricht einer Tabelle. Tabellen sind allgemein bekannt: Sie bestehen aus Zeilen und Spalten, die den entsprechenden Index des Arrays widerspiegeln.

Beispiel:

Es wird ein zweidimensionales Array angelegt.

```
float Tab [ 3 ] [ 4 ];
```

entspricht der Anzahl der Spalten

entspricht der Anzahl der Zeilen

	Spalte 0	Spalte 1	Spalte 2	Spalte 3
Zeile 0	1.5	7	12.33	77.5
Zeile 1	124	99.99	453	67.89
Zeile 2	12	90.2	2727.5	22

```
Tab [ 1 ] [ 2 ] == 453
```

Spaltenangabe

Zeilenangabe

Dreidimensionale Arrays kann man sich als eine Sammlung von Tabellenblättern vorstellen, die hintereinander angeordnet sind.

Beispiel:

Es wird ein dreidimensionales Array angelegt.

```
float Tab [ 3 ] [ 3 ] [ 4 ];
```

		Blatt 2	Spalte 0	Spalte 1	Spalte 2	Spalte 3
		Blatt 1	Spalte 0	Spalte 1	Spalte 2	Spalte 3
		Blatt 0	Spalte 0	Spalte 1	Spalte 2	Spalte 3
Zeile 0		1.5	7	12.33	77.5	45.55
Zeile 1		124	99.99	453	67.89	78.6
Zeile 2		12	90.2	2727.5	22	

```
Tab [ 1 ] [ 1 ] [ 3 ] == 45.55
```

Spaltenangabe

Zeilenangabe

Tabellenblatt

Nach der dritten Dimension hört die menschliche Vorstellungskraft auf. Die mehrdimensionalen Arrays mit mehr als drei Dimensionen sind dann auch nicht mehr so konkret vorstellbar wie beispielsweise die Tabellen, aber dennoch sind sinnvolle Einsätze dieser Arrays denkbar.

10 Das Klassenkonzept in C++

Die Sprache C++ ist eine Hybridsprache. Sie ist sowohl zur strukturierten Programmierung als auch zur objektorientierten Programmierung geeignet. Die ersten neun Kapitel dieses Buches waren eine Einführung in die strukturierte Programmierung mit C++. Diese Grundlagen sind wichtig, um das Konzept der objektorientierten Programmierung verstehen und umzusetzen zu können.

Unter objektorientierter Programmierung kann eine spezielle Art der Programmierung verstanden werden, die versucht, gewisse Gegebenheiten möglichst realitätsnah umzusetzen.

Im Mittelpunkt der objektorientierten Programmierung steht das Objekt bzw. die Klasse. Eine Klasse kann als eine Art Bauplan betrachtet werden, mit dem Objekte gebildet werden können. Die Begriffe Objekt und Klasse werden nun näher betrachtet.

Was ist ein Objekt?

Ein Objekt ist eine softwaretechnische Repräsentation eines realen oder gedachten, klar abgegrenzten Gegenstandes oder Begriffs.

Das Objekt erfasst alle Aspekte des Gegenstandes durch Attribute (Eigenschaften) und Methoden.

Was sind Attribute und Methoden?

Attribute sind die Eigenschaften des Objektes. Sie beschreiben den Gegenstand vollständig. Attribute sind geschützt gegen Manipulation von außen (das nennt man Kapselung).

Methoden beschreiben die Operationen, die mit dem Objekt (bzw. seinen Attributen) durchgeführt werden können. Von außen erfolgt der Zugriff auf Attribute durch die Methoden.

Was ist eine Klasse?

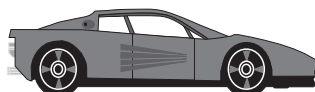
Unter einer Klasse versteht man die softwaretechnische Beschreibung eines Bauplanes für ein Objekt.

Aus einer Klasse können dann Objekte abgeleitet (gebildet) werden.

Diese etwas abstrakten, aber wichtigen Begriffsdefinitionen sollen nun anhand von Beispielen veranschaulicht werden.

Beispiel:

Diese Rennwagen sind konkrete Objekte. Sie haben Attribute wie Farbe, Leistung in KW oder Hubraum.



Name:	Lotus
Farbe:	grau
KW:	250
Hubraum:	4 Liter



Name:	Spider XL
Farbe:	schwarz
KW:	300
Hubraum:	5 Liter

16 C++ und Windows Store Apps

16.1 Applikationen unter Windows 8

16.1.1 Neue Windows-Konzeption

Mit Windows 8 brachte Microsoft 2012 ein neues Betriebssystem heraus, das sich konzeptionell deutlich von den vorherigen Systemen unterscheidet. Auf der einen Seite ist es auf die Anforderungen der mobilen Geräte (Smartphones und Tablet-PCs) vorbereitet, indem es eine völlig veränderte Oberfläche anbietet, auf denen die Anwendungen in Form von *Kacheln* dargestellt werden. Diese Kacheln dienen auf den mobilen Geräten zur manuellen Eingabesteuerung (Touchscreen). Die Anwendungen, die mit solchen Kacheln gestartet werden, heißen *Windows Store-Apps* und zeichnen sich unter anderem dadurch aus, dass sie den ganzen Bildschirm ausfüllen und nicht mehr von einem Fensterrahmen begrenzt werden. Zusätzlich sind diese Apps ausschließlich über den Windows Store zu beziehen und werden in einer völlig anderen Form installiert als die konventionellen Windows-Applikationen. Die Entwicklung solcher Apps¹ wird das Thema der folgenden Kapitel sein.

Auf der anderen Seite bietet Windows 8 aber auch noch eine klassische Desktop-Oberfläche, die ähnlich wie die Oberfläche von Windows 7 (oder auch Windows XP) zu nutzen ist. Es fehlt nur der gewohnte Start-Button, der aber einfach mit kostenfreien Tools nachzuinstallieren ist.



16.1.2 Windows Runtime

Die Entwicklung von Windows Store Apps basiert auf einer völlig neuen Schnittstelle, der **Windows Runtime API** (API = Application Programming Interface). Diese objektorientierte Schnittstelle basiert auf dem COM²-Modell und wurde in C++ geschrieben. Im Gegensatz zu der .NET-Technologie steht die Windows Runtime für eine native Programmentwicklung. Das bedeutet, dass die Programmentwicklung sofort auf eine Schnittstelle zugreift, die Bestandteil des Betriebssystems ist. Damit entfällt eine Zwischenschicht wie unter .NET. Ein großer Vorteil ist die schnelle Geschwindigkeit solcher Anwendungen. Der native Zugriff auf die Windows Runtime geschieht deshalb auch mit C++ (genauer gesagt mit der Erweiterung der Sprache C++/CX, dazu später mehr). Aber auch für andere Sprachen wie C# oder Visual Basic ist der Zugriff über bestimmte Klassen (*Wrapper*³-Klassen) ebenfalls möglich – allerdings müssen dann Übersetzungen von der .NET-Technologie (dem ver-

¹ Dazu müssen das Betriebssystem Windows 8 und das Visual Studio Express für Windows 8 installiert sein.

² COM (Component Object Model) ist eine Technik von Microsoft, mit der Softwarekomponenten erstellt werden können. Dabei ist die Wahl der Programmiersprache unerheblich.

³ Wrapper-Klassen „umhüllen“ andere Klassen und bieten damit die Möglichkeit bestimmte Funktionalitäten auf ein anderes System zu übertragen.

16.2 C++11

16.2.1 Der neue Standard C++11

Der letzte C++-Standard liegt mehr als 10 Jahre zurück (ISO⁵ 1999). Im Jahr 2011 war es dann endlich soweit und der neue Standard (deshalb auch C++11) wurde veröffentlicht. Die Sprache wurde von Grund auf überholt und es entstand eine moderne objektorientierte Sprache. Bislang galt C++ immer als schwer zu erlernen. Mit den neuen Standards sind einige Hürden für den Programmieranfänger gesenkt worden. Außerdem tritt C++ jetzt aus dem Schattendasein hervor, in das es seit den Markteinführungen von Java und C# gedrängt wurde – Microsoft stellt C++11 (bzw. die Erweiterung C++11/CX) mit dem neuen Betriebssystem Windows 8 wieder in den Mittelpunkt der Entwicklung.

Die folgende Tabelle zeigt einige wesentliche Neuerungen von C++11⁶, die im Anschluss auch anhand von Beispielen erklärt werden.

Neuerung	Beschreibung
Der <code>auto</code>-Typ	Eine <code>auto</code> -Variable erhält seinen Datentyp durch die Initialisierung. Wird die <code>auto</code> -Variable beispielsweise mit einem numerischen Wert initialisiert, so ersetzt der Compiler das <code>auto</code> durch ein <code>double</code> .
Intelligente Zeiger (<i>smart-pointer</i>)	Diese Art von Zeigern erleichtert das Handling mit dynamischer Speicherreservierung.
Lambda-Ausdrücke	Diese Ausdrücke verkörpern im Prinzip anonyme Funktionen und helfen beispielsweise auf sehr elegante Weise bei der Suche in STL-Arrays.
Die neue <code>for</code>-Schleife	In vielen modernen Programmiersprachen bereits ein Standard (beispielsweise <code>foreach</code> in C#): die neue <code>for</code> -Schleife erlaubt einfaches Iterieren durch ein Array.

16.2.2 Der `auto`-Typ

Dieser neue Typ vereinfacht die Definition von Variablen. Der Entwickler muss sich keine Gedanken mehr über den Datentyp machen, wenn er beispielsweise den Rückgabewert einer Funktion speichern will. Er setzt die Variable einfach auf den `auto`-Typ, wie das folgende Beispiel zeigt:

Beispiel:

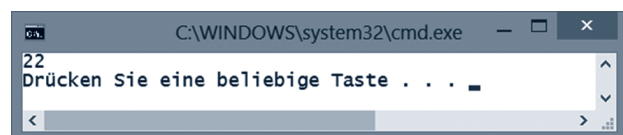
```
#include <iostream>
using namespace std;

double Funktion()
{
    return 10.5;
}

int main()
{
    auto x = 11.5;
    auto y = Funktion();
    auto z = x + y;

    cout << z << endl;
    return 0;
}
```

Durch die Initialisierung erhält die `auto`-Variable ihren Typ.



16.2.3 Intelligente Zeiger

Die dynamische Speicherreservierung liegt auch unter C++11 in den Händen des Entwicklers, da C++11 kein verwalteter Code ist und deshalb auch kein *garbage-collector* wie in Java oder C# für die Aufräumarbeiten sorgt. Dementsprechend ist die Speicherreservierung immer eine Quelle von Fehlern. Die intelligenten Zeiger sorgen dafür, dass dieses Problem entschärft wird. Dazu werden neue Arten von Zeigern zu Verfügung gestellt: `unique_ptr` und `shared_ptr`.

⁵ ISO (International Standardization Organization oder besser International Organization for Standardization) ist eine internationale Vereinigung, die in allen Bereichen (außer Elektronik und Elektrik) Normierungen vornimmt.

⁶ Die Neuerungen von C++ können natürlich nur dann angewendet werden, wenn die Entwicklungsumgebung entsprechend modern ist (beispielsweise das *Visual Studio Express 2012 für Desktop*).

16.4.3 Layout-Container

Das Inhaltskonzept von XAML sieht vor, dass alle Steuerelemente letztendlich einem Container zugewiesen werden. XAML bietet einige Layout-Container an, deren Eigenschaften hier kurz vorgestellt werden:

Layout-Container	Beschreibung
StackPanel	Die Elemente werden einfach untereinander oder nebeneinander angeordnet. Es können beliebig viele Elemente angefügt werden.
WrapPanel	Die Steuerelemente werden zeilenweise angeordnet. Wenn eine Zeile keinen Platz mehr hat, dann wird auf die nächste Zeile verteilt.
DockPanel	Die Elemente werden an verschiedenen Seiten „angedockt“ (Left, Top, Right, Bottom).
Grid	Die Steuerelemente werden in einer Art Tabelle angeordnet. Zusätzlich kann ein Steuerelement über mehrere Tabellenzellen definiert werden.
Canvas	Dieser Container erlaubt eine Anordnung nach festen Koordinaten und entspricht am ehesten der bekannten GUI-Programmierung.

Beispiel:

Das folgende Beispiel zeigt, wie in einem `StackPanel` zwei `Buttons` eingebettet werden:

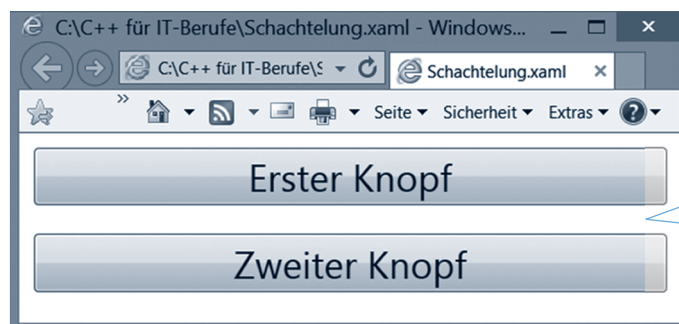
```
<StackPanel xmlns="http://schemas.microsoft.com/winfx/2006/xaml/
presentation">

    <Button Name="knopf1" FontSize = "20pt" Margin = "10,10,10,10">
        Erster Knopf
    </Button>

    <Button Name="knopf2" FontSize = "20pt" Margin = "10,10,10,10">
        Zweiter Knopf
    </Button>

</StackPanel>
```

Nach dem Starten des obigen *Loose-XAML* erscheinen die Buttons einwandfrei „gestapelt“ – entsprechend dem Layout des `StackPanel`-Containers:



Es wird jeweils ein Abstand von 10 eingehalten.

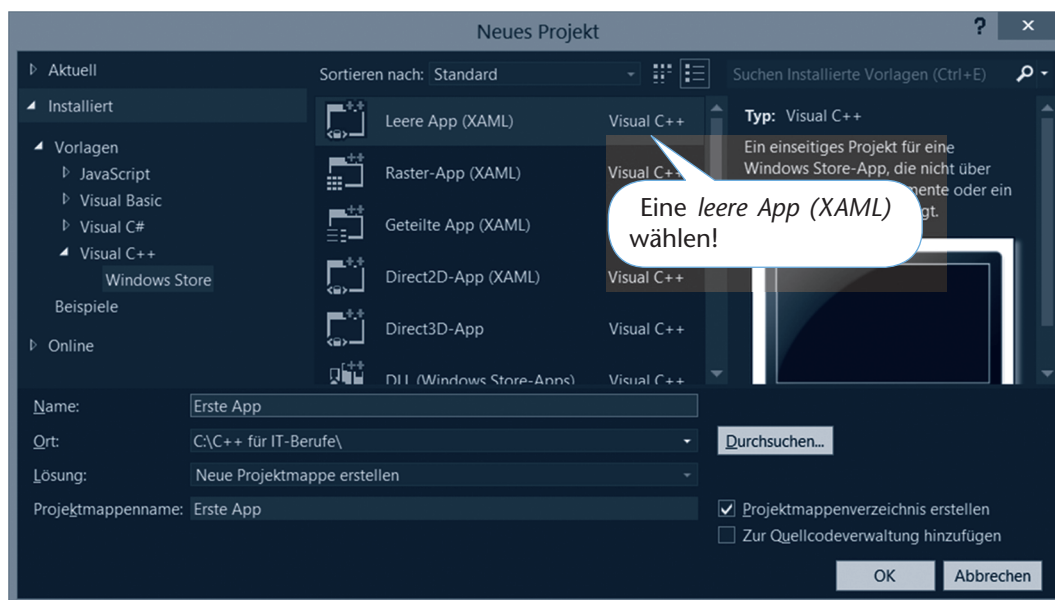
An diesen ersten Beispielen ist erkennbar, dass XAML hervorragend geeignet ist, um die visuelle Gestaltung einer GUI-Anwendung vorzunehmen. Die Logik der Anwendung wird dann in den *Code-behind*-Dateien in einer Sprache wie C++ umgesetzt.

17 Windows Store App-Entwicklung

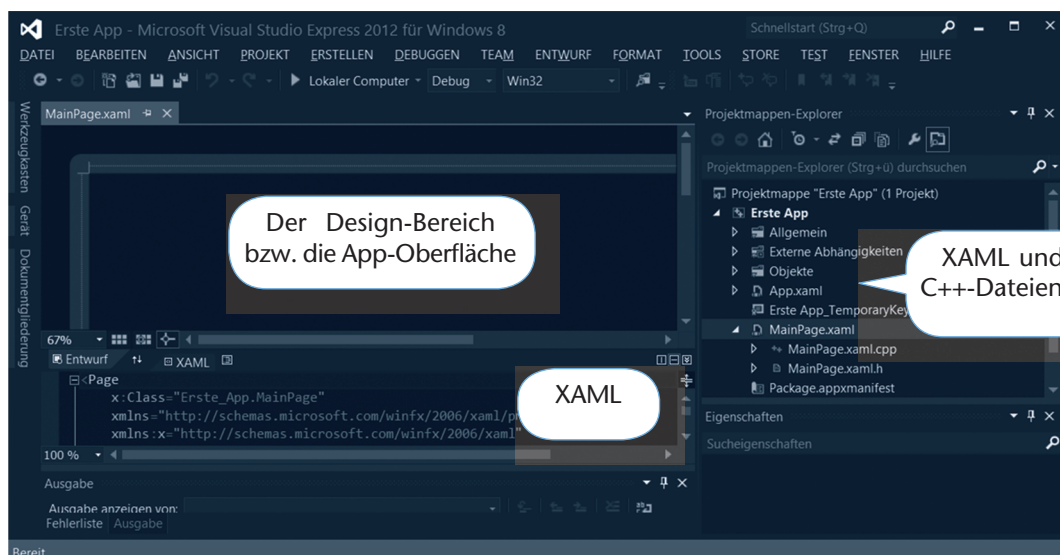
17.1 Die erste Windows Store App

17.1.1 Ein leeres App-Projekt anlegen

Mit der kostenfreien Entwicklungsumgebung *Visual Studio 2012 Express für Windows 8* werden nun Windows Store Apps entwickelt. Nach den Grundlagen zu C++/CX und XAML aus dem letzten Kapitel konzentrieren sich die Ausführungen nun auf die eigentliche App-Entwicklung mit dem Visual Studio. Die Entwicklungsumgebung hilft bei der Erstellung der GUI mit einem automatischen XAML-Generator. Die Logik hinter der GUI wird dann in den *Code-behind*-Dateien implementiert. Der erste Schritt ist das Anlegen eines neuen Projekts unter Visual Studio 2012:



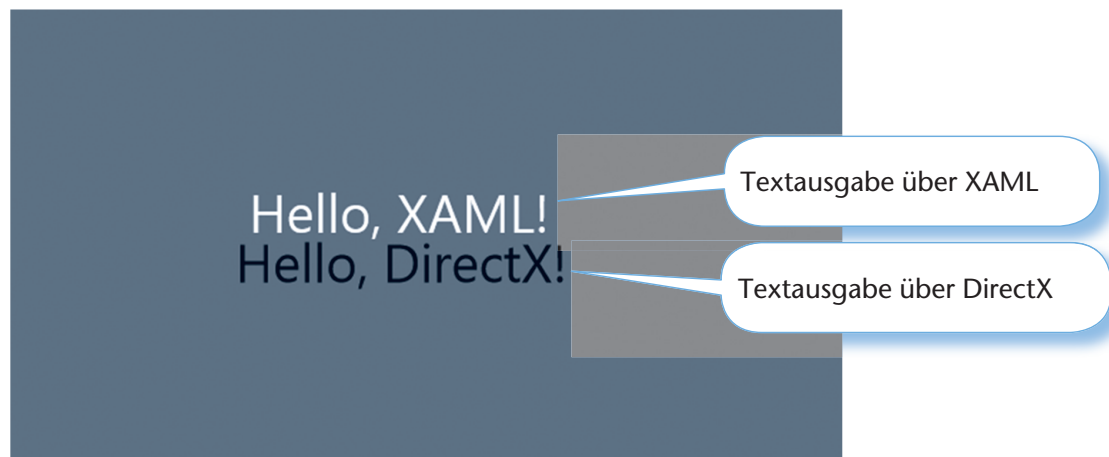
Nach dem Bestätigen mit „OK“ erscheint das neue App-Projekt in der Entwicklungsumgebung:



Die Entwicklungsumgebung erstellt einige Quellcode-Dateien, die als Basis für die Anwendung dienen:

Datei	Beschreibung
BasicTimer.h	Enthält eine Klasse als Basis für die Zeitsteuerung.
DirectXBase.h/cpp	Diese Dateien enthalten die abstrakte Basisklasse <code>DirectXBase</code> . Sie dient als Grundlage zur Vererbung und liefert die wesentlichen Attribute und Methoden.
DirectXPage.xaml DirectXPage.xaml.h/cpp	Die XAML-Datei, die die Oberfläche beschreibt. Sie enthält ein besonderes Element: <code>SwapChainBackgroundPanel</code> . Damit kann direkt auf die Oberfläche gezeichnet (gerendert) werden. Die <i>Code-behind</i> -Dateien sorgen für Logik im Hintergrund. Sie enthalten die Definition der Klasse <code>DirectXPage</code> , die beispielsweise für die Ereignissteuerung zuständig ist.
SimpleTextRenderer.h/cpp	Die Dateien zeigen eine einfache Umsetzung einer Klasse, die von der Basisklasse <code>DirectXBase</code> erbt. Sie enthält die wesentlichen Methoden und kann als Ausgangsbasis der eigenen Entwicklung verwendet werden.

Nach dem Starten erscheint der folgende Bildschirm:



Die App ist in der Lage den DirectX-Text mit der Maus (oder einem Zeigegerät) auf dem Bildschirm zu verschieben:



Die Speicherung der Matrix soll mithilfe der dynamischen Speicherreservierung erfolgen. Dabei sollen folgende Funktionalitäten in der Klasse umgesetzt werden:

Methode Einlesen() :

Der Benutzer kann die Dimension der Matrix angeben und anschließend wird dynamisch Platz reserviert und die Werte eingelesen.

Methode Ausgabe() :

Die Matrix wird (formatiert) auf dem Bildschirm ausgegeben.

Methode Transponieren() :

Die Matrix wird transponiert. Das bedeutet, dass alle Matrixwerte a_{ij} ihre Indizes vertauschen ($a_{ij} = a_{ji}$).

Beispiel einer Transponierung:

$$\begin{bmatrix} 1 & 7 & 2 \\ 3 & 2 & 2 \\ 5 & 6 & 3 \end{bmatrix} \Rightarrow \begin{bmatrix} 1 & 3 & 5 \\ 7 & 2 & 6 \\ 2 & 2 & 3 \end{bmatrix}$$

Zusätzlich sollen folgende Operatoren überladen werden:

- **Gleichheitsoperator =**
- **Additionsoperator +**
- **Multiplikationsoperator ***

Erläuterungen zur Addition und Multiplikation:

Zwei Matrizen werden addiert, indem jedes Element der einen Matrix zu dem Element mit denselben Indizes der anderen Matrix addiert wird.

Beispiel:

$$\begin{bmatrix} 1 & 7 & 2 \\ 3 & 2 & 2 \\ 5 & 6 & 3 \end{bmatrix} + \begin{bmatrix} 2 & 6 & 5 \\ 7 & 1 & 6 \\ 2 & 2 & 7 \end{bmatrix} = \begin{bmatrix} 1+2 & 7+6 & 2+5 \\ 3+7 & 2+1 & 2+6 \\ 5+2 & 6+2 & 3+7 \end{bmatrix} = \begin{bmatrix} 3 & 13 & 7 \\ 10 & 3 & 8 \\ 7 & 8 & 10 \end{bmatrix}$$

Die Multiplikation ist etwas komplizierter (siehe auch Aufgabe 7.4):

Die erste Zeile der ersten Matrix wird mit der ersten Spalte der zweiten Matrix elementweise multipliziert und anschließend werden die Produkte addiert. Das Endergebnis ist das erste Element der Multiplikationsmatrix.

Danach wird die erste Zeile der ersten Matrix mit der zweiten Spalte der zweiten Matrix elementweise multipliziert und anschließend werden die Produkte addiert. Das Endergebnis ist das zweite Element der Multiplikationsmatrix usw.

Am anschaulichsten ist es mit einem einfachen Beispiel:

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \cdot \begin{bmatrix} 3 & 4 \\ 5 & 6 \end{bmatrix} = \begin{bmatrix} 1 \cdot 3 + 2 \cdot 5 & 1 \cdot 4 + 2 \cdot 6 \\ 3 \cdot 3 + 4 \cdot 5 & 3 \cdot 4 + 4 \cdot 6 \end{bmatrix} = \begin{bmatrix} 13 & 16 \\ 29 & 36 \end{bmatrix}$$

Hinweise:

- Denken Sie an den Standardkonstruktor, den Copy-Konstruktor und den Destruktor sowie die korrekte Freigabe des reservierten Speichers.
- Achten Sie auf modulare Programmgestaltung.
- Für die dynamische Reservierung ist ein Zeiger auf einen Zeiger (Typ `int` oder `double`) nötig.

```

void MainPage::TICTACTOEPointerPressed(Platform::Object^ sender,
    Windows::UI::Xaml::Input::PointerRoutedEventArgs^ e)
{
    auto wer = (TextBlock^) sender;
    const wchar_t *wo = wer->Name->Data();
    int x = wo[1]-48;
    int y = wo[3]-48;
    :
}

```

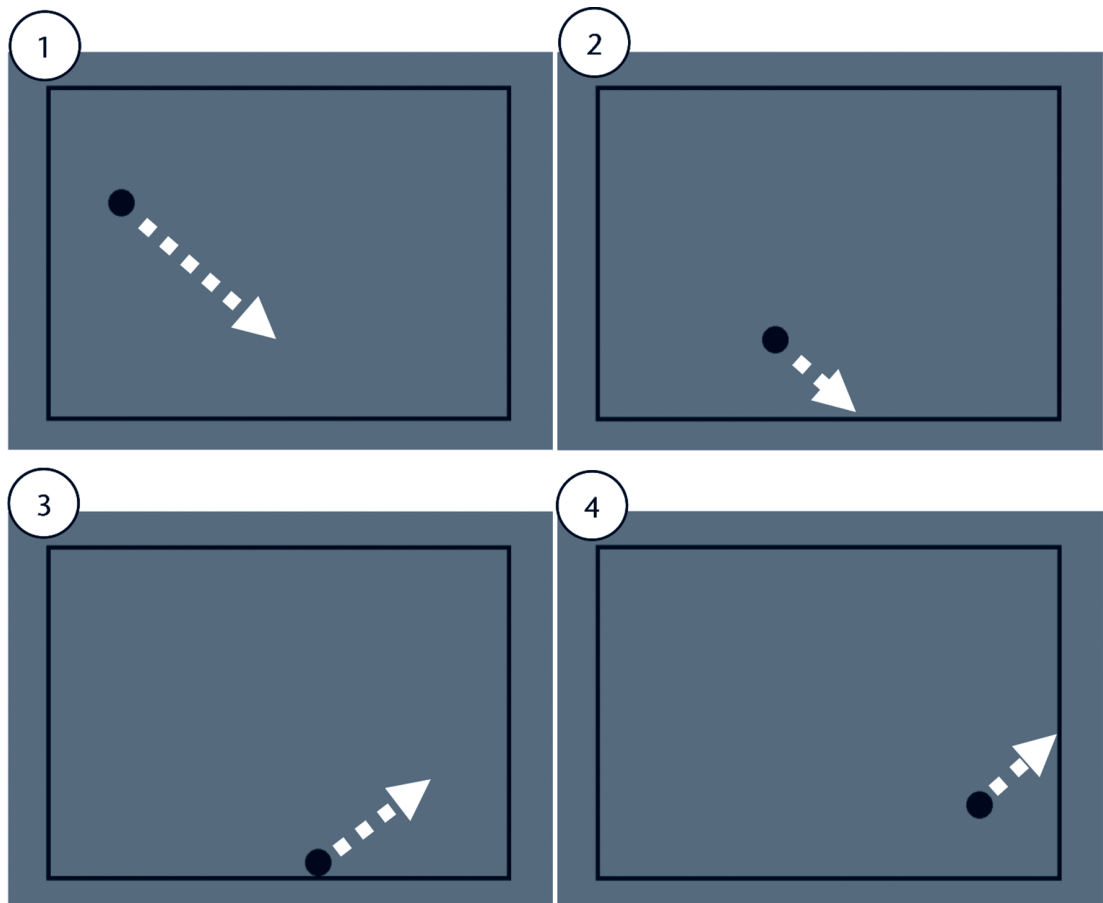
Textblock identifizieren

Position filtern

Aufgabe 17.2

Schreiben Sie eine Direct2D-App (XAML), die einen Ball (bzw. Kreis) in einem vordefinierten Rechteck springen lässt. Stößt der Ball an die Seiten des Rechtecks, so ändert er seine Richtung um 90°. Wenn der Benutzer auf die linke Maustaste (bzw. ein anderes Zeigegerät) drückt, dann soll sich die Ballgeschwindigkeit erhöhen.

Das fertige Programm könnte dann so aussehen:



Lernsituation 8:

Entwicklung einer Grafik-App für einfache Skizzen

Ausgangssituation:

Die Firma **ProSource** hat einige Kunden aus dem handwerklichen Bereich. Um die Kundenbindung zu stärken, möchte **ProSource** ihren Kunden eine kostenfreie Grafik-App zu Verfügung stellen. Damit sollen die Monteure vor Ort einfache Skizzen auf ihren Tablet-PCs anfertigen können, die dann beispielsweise später als Grundlage eines Kostenvoranschlages dienen können. Mit dem Einsatz der neusten App-Technik möchte **ProSource** seinen Kunden auch demonstrieren, dass bei Prosource mit aktueller Technik entwickelt wird.

Als Auszubildender der Firma **ProSource** erhalten Sie nun den Auftrag die Grafik-App zu entwickeln. In einem ersten Prototyp soll die App elementare Grafikobjekte wie Linien, Ellipsen und Rechtecke zeichnen können.

Arbeitsschritte in Einzel- oder Partnerarbeit:

Planung:

Für den ersten Prototyp hat der Teamleiter der Softwareentwicklung bereits einige Vorüberlegungen getroffen:

Die App soll über einige Steuerelemente in der sogenannten `AppBar` verfügen. Das ist der untere Bereich der App, der sich bei einem Klick auf die rechte Maustaste in diesem Bereich öffnet:

