

# Inhaltsverzeichnis

<b>Einführung</b>	<b>19</b>
Über dieses Buch	19
Konventionen in diesem Buch	20
Was Sie nicht lesen müssen	20
Voraussetzungen	20
Wie dieses Buch aufgebaut ist	20
Symbole in diesem Buch	21
Wie geht es weiter?	21
<b>Teil II</b>	
<b>Visual C++ .NET in sechs leichten Kapiteln</b>	<b>23</b>
<b>Kapitel 1</b>	
<b>Was befindet sich in dem Visual-C++-.NET-Paket?</b>	<b>25</b>
Visual C++-Funktionen	25
Mit einem Compiler Programme erstellen	26
Mit einem Debugger Fehler einfangen	27
Mit einer integrierten Entwicklungsumgebung arbeiten	27
Mit Bibliotheken Funktionen und Klassen wiederverwenden	28
Das Arbeiten mit Utilities erleichtern	29
Hilfe anfordern	29
Mit Beispielprogrammen den Einstieg schaffen	29
Verwaltete und nicht verwaltete Programme	30
<b>Kapitel 2</b>	
<b>Der Einstieg in das Programmieren</b>	<b>33</b>
Ein Grundkurs im Programmieren	33
Die main-Funktion	35
Bibliotheken	35
Probleme mit einem Programm lösen	36
Ein ganz anderer Ansatz	39
Der zentrale Begriff des Objekts	40
Einkapselung	41
Vererbung	41
Polymorphismus	42

<b>Kapitel 3</b>	
<b>Das erste Programm schreiben</b>	<b>45</b>
Quelldateien	45
Ihr erstes .NET-Projekt	46
Die main-Funktion	48
Programme mit den Console-Funktionen erstellen	49
Namensräume mit »#using« angeben	50
Programme kommentieren	51
Den Quellcode analysieren	52
Das »Hello World«-Programm verbessern	53
Ohne .NET codieren	54
Daten anzeigen oder ausdrucken	54
Ausgaben mit Sonderzeichen formatieren	55
Daten eingeben	56
Funktionen einer Bibliothek nutzen	57
Die nicht verwaltete Version von »Hello World«	58
<b>Kapitel 4</b>	
<b>Projekte Organisieren</b>	<b>61</b>
Projektmappen und Makefiles	61
Projektdateien	62
Projektmappen und Projekte	62
Programme in einem Schritt erstellen	63
Details für ein neues Projekt festlegen	63
Dateien zu einem Projekt hinzufügen	64
Gebräuchliche Funktionen des Projektmappen-Explorers	66
<b>Kapitel 5</b>	
<b>Dateien mit einem Editor bearbeiten</b>	<b>69</b>
Mehrere Dateien bearbeiten	69
Code editieren	71
Den Code je nach Syntax einfärben	73
Online-Hilfe anfordern	73
Im Code navigieren	74
Code gliedern, verbergen und anzeigen	74
Text im Code suchen und ersetzen	76

<b>Kapitel 6</b>		
<b>Programme kompilieren</b>		<b>81</b>
Kompilieren ist keine Einmalaktion		81
Aus Syntaxfehlern lernen		82
Warnungen beachten		83
Warum korrigiert der Compiler die Fehler nicht?		84
Verschiedene Arten des Kompilierens		85
<b>Teil II</b>		
<b>Alles, was Sie schon immer über C++ wissen wollten, sich aber nicht zu fragen trauten</b>		<b>87</b>
<b>Kapitel 7</b>		
<b>Daten und Datentypen</b>		<b>89</b>
Typisierte Sprachen		89
Den Typ einer Variablen deklarieren		90
Die grundlegenden Datentypen		90
Weniger gebräuchliche Datentypen		91
Typensicherheit		93
String-Umwandlungen		94
Konstanten – Dinge, die sich nie ändern		95
Konstanten zu einem Programm hinzufügen		96
Konstanten und Fehlervermeidung		97
Grundlegende String-Funktionen		97
<b>Kapitel 8</b>		
<b>Mit Variablen arbeiten</b>		<b>103</b>
Variablen benennen		103
Variablen definieren		105
Variablen initialisieren		106
Konventionen bei der Namensvergabe		106
<b>Kapitel 9</b>		
<b>Strukturen</b>		<b>109</b>
Strukturen deklarieren		109
Mit Strukturen arbeiten		111
Große Strukturen aus kleinen Strukturen zusammensetzen		111
Ein Programm mit Strukturen		111

<b>Kapitel 10</b>	
<b>Mit Ausdrücken arbeiten</b>	<b>115</b>
Ausdrücke	115
Operatoren	116
Etwas komplexere Operatoren	116
Der <code>++</code> -Operator	117
Der <code>&gt;&gt;</code> -Operator	118
Der <code>&lt;&lt;</code> -Operator	118
Wahrheitswerte mit Booleschen Ausdrücken berechnen	119
Zuweisungsoperatoren	121
Kurzformen für Operatoren	123
Bit-Operatoren	124
Der <code>if</code> -Operator	125
Die Auswertungsreihenfolge von Operatoren	127
Einige Beispiele für Operatoren	128
Mathematische Funktionen in Visual C++ .NET	129
Mathematische Funktionen in C++	131
<b>Kapitel 11</b>	
<b>Den Programmablauf steuern</b>	<b>133</b>
Die drei wichtigsten Kontrollanweisungen: <code>if</code> , <code>for</code> und <code>while</code>	134
Die <code>if</code> -Anweisung	134
Die <code>for</code> -Schleife	138
Die <code>while</code> -Schleife	141
Mehrfachverzweigungen und <code>do</code> -Schleifen	142
Der <code>switch</code> -Befehl	142
Der <code>do</code> -Befehl	144
<b>Kapitel 12</b>	
<b>Mit Funktionen arbeiten</b>	<b>145</b>
Anweisungen mit einem Semikolon abschließen	145
Die Syntax von Funktionsdefinitionen	146
Argumente an Funktionen übergeben	147
Funktionen mit Rückgabewerten	149
Eine überarbeitete Version des Fakultät-Beispiels	151
Rekursion: Funktionen, die sich selbst aufrufen	154
Fortsetzungspunkte (Ellipsen) verwenden	157
Standardargumente	158

<b>Kapitel 13</b>	
<b>Mit Zeigern arbeiten</b>	<b>159</b>
Warum arbeitet man mit Zeigern?	159
Zeiger sind allgegenwärtig	160
Zeiger und Variablen	160
Schwierigkeiten beim Arbeiten mit Zeigern	160
Ein Beispiel für Zeiger: verkettete Listen	163
Zeiger in C++ verwenden	164
Einem Zeiger eine Adresse zuweisen	164
Den Wert ändern, auf den ein Zeiger verweist	167
Den Wert in einer Struktur ändern	167
Eine spezielle Zeigersyntax	168
Mit »new« einen neuen Speicherplatz reservieren	168
Grafische Objekte	169
Farben	171
Zeichenstifte	173
Pinsel	173
Schriftarten	174
Ein einfaches Zeichenprogramm	174
Ein klassisches Programm: eine verkettete Liste	176
Wie das Programm funktioniert	176
Der Code	178
Fallstricke bei Zeigern	182
Speicher freigeben	182
Schutzverletzungen	183
Garbage Collection	184
Mit Strings arbeiten	185
Ein kurzer Rückblick	187
<b>Kapitel 14</b>	
<b>Mit Arrays arbeiten</b>	<b>189</b>
Grundbegriffe von Arrays	189
Auf die Elemente eines Arrays zugreifen	191
Arrays initialisieren	191
Mehrdimensionale Arrays	193
Die ArrayList-Klasse von .NET	195
Stacks (Stapel)	196
Aufzählungstypen	197
Sicherheit durch Aufzählungen	197
Auslassungsfehler	198
Ein Beispiel	198

<b>Kapitel 15</b>		
<b>Der Gültigkeitsbereich von Namen</b>		<b>199</b>
Der Gültigkeitsbereich einer Variablen		199
Warum ist der Gültigkeitsbereich so wichtig?		200
Regeln für Gültigkeitsbereiche		203
<b>Kapitel 16</b>		
<b>Programme debuggen</b>		<b>205</b>
Der Unterschied zwischen Syntaxfehlern und Bugs		206
Ein Überblick über das Debuggen		207
Die Zusammenarbeit von Editor und Debugger		208
Haltepunkte setzen		208
In Funktionen hineinspringen oder Funktionen überspringen		209
Variablen überwachen		210
Variablenwerte während des Überwachens ändern		212
Die Schnellüberwachung		212
Variablenwerte noch schneller anzeigen		212
Eine praktische Debugging-Sitzung		214
Das Programm erstellen		214
Die Stelle des Fehlers finden		215
Ist das Problem damit gelöst?		219
Es gibt noch mehr Bugs		220
Das Programm erneut korrigieren		220
Die Haltepunkte und die Überwachung wieder entfernen		221
<b>Teil III</b>		
<b>Objektorientiert programmieren</b>		<b>223</b>
<b>Kapitel 17</b>		
<b>Grundlagen der objektorientierten Programmierung</b>		<b>225</b>
Eine Einführung in die objektorientierte Programmierung		225
Klassen konstruieren		226
Membervariablen		226
Memberfunktionen (Methoden)		227
Eine Klasse deklarieren		227
Den Zugriff auf eine Klasse einschränken		228
Geschützter Zugriff		228
Memberfunktionen definieren		229
Mit Klassen arbeiten		230
Auf Members (Klassenelemente) zugreifen		230

Von Memberfunktionen aus auf Membervariablen zugreifen	231
Eine objektorientierte Version des Zeichenprogramms	232
Den Speicher aufräumen	236
Zugriffsfunktionen	241
Einige Ratschläge zur Vorgehensweise	245
<b>Kapitel 18</b>	
<b>Konstruktoren und Destruktoren</b>	<b>249</b>
Initialisieren und aufräumen	249
Mit Konstruktoren arbeiten	250
Mehrere Konstruktoren in einer Klasse	251
Öffentliche und private Konstruktoren	252
Destruktoren	253
Den Speicher aufräumen	253
Den Speicher dynamisch erzeugter Objekte freigeben	256
Klassen innerhalb von Klassen	256
Objektorientierte Programme lesen	258
<b>Kapitel 19</b>	
<b>Vererbung</b>	<b>261</b>
Wie Vererbung funktioniert	261
Vererbung bei öffentlichen, geschützten und privaten Elementen	263
Elemente überschreiben	264
Auf die Basisklasse zugreifen	264
Ein Vererbungsbeispielprogramm	265
Der Einfluss der Vererbung auf Konstruktoren und Destruktoren	267
Zeiger und abgeleitete Klassen	268
Vererbung bei privaten und geschützten Members	269
Virtuelle Funktionen	270
Wann sollten Sie virtuelle Funktionen verwenden?	271
Virtuelle Funktionen deklarieren	271
Ein Beispielprogramm für virtuelle Funktionen	272
Abstrakte Basisklassen	275
Ein Beispielprogramm für virtuelle Funktionen und abstrakte Basisklassen	277
<b>Kapitel 20</b>	
<b>Ausnahmen behandeln</b>	<b>283</b>
Die herkömmliche Form der Ausnahmebehandlung	283
Fehlerbehandlung auf die neue, verbesserte Art	285
Ein Beispiel für eine Ausnahmebehandlung	286
Flexibilität durch Ausnahmebehandlung	287

Fehler abfangen	288
Die Syntax der Ausnahmebehandlung	289
Den passenden catch-Block auswählen	292
Fehlerverarbeitende Klassen vererben	293
Fünf Regeln für eine erfolgreiche Ausnahmebehandlung	294
<b>Kapitel 21</b>	
<b>Mit Dateien arbeiten</b>	<b>297</b>
I/O-Klassen von .NET	297
Zahlen und Wörter schreiben und lesen	298
iostream-Bibliotheksroutine	299
Stream-Variablen wiederverwenden	300
Zahlen und Wörter mit .NET schreiben und lesen	300
Daten bei der Ein- oder Ausgabe manipulieren	302
Füllzeichen und Ausgabebreite setzen	302
Fünf Tipps zum Arbeiten mit Dateien	303
<b>Kapitel 22</b>	
<b>Mit WinForms arbeiten</b>	<b>305</b>
Was sind WinForms-Klassen?	305
Mit Formularen arbeiten	307
Das Formular ableiten	307
Das Formular einrichten	308
Das Formular in ein Programm einbinden	309
Ereignisse verarbeiten	309
Methoden zur Ereignisverarbeitung	310
Delegates	311
<b>Teil IV</b>	
<b>Der Top-Ten-Teil</b>	<b>315</b>
<b>Kapitel 23</b>	
<b>Zehn Syntaxfehler</b>	<b>317</b>
Die falschen Include-Pfade verwenden	317
Ein fehlendes Semikolon	318
Vergessen, eine Header-Datei einzubinden	318
Vergessen, die Klassendeklaration zu aktualisieren	319
Den Klassennamen anstelle des Variablenamens verwenden	320
Das Semikolon am Ende einer Klassendeklaration vergessen	320
Vergessen, »public:« in eine Klassendefinition einzufügen	321

Den falschen Variablenamen verwenden	321
-> anstelle von . verwenden und umgekehrt	322
Eine }-Klammer vergessen	323
<b>Kapitel 24</b>	
<b>Zehn weitere Syntaxfehler</b>	<b>325</b>
Konstruktoren ohne Argumente verwenden	325
Vergessen, einen Kommentar abzuschließen	326
Den falschen Variablentyp verwenden	326
Es funktionierte als C-Programm, wird aber jetzt nicht kompiliert	327
»Nichts« statt »void« verwenden	327
Vergessen, »#using« zu verwenden	328
Keinen öffentlichen Konstruktor verwenden, wenn er benötigt wird	328
Ein Semikolon an das Ende eines #define setzen	329
Die Festplatte ist voll	329
Es gibt wirklich Probleme	330
<b>Kapitel 25</b>	
<b>Die zehn wichtigsten .NET-Funktionen</b>	<b>331</b>
Console::WriteLine	331
Console::ReadLine	331
Int32::Parse	332
Application::Run	332
Graphics->DrawLine	332
Color::FromArgb	332
Graphics->DrawString	333
Image::FromFile	333
Form::OnMouseMove	333
Controls->Add	333
<b>Anhang</b>	
<b>Über die CD</b>	<b>335</b>
Systemanforderungen	335
Mit der CD arbeiten	335
Der Inhalt der CD	337
Wenn Sie Probleme mit der CD haben...	338
<b>Stichwortverzeichnis</b>	<b>339</b>