

C++ für IT-Berufe

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

int main()
{
    cout << "Informationsteil:" << endl;

    cout << " - Strukturierte Programmierung  
mit C++" << endl;

    cout << " - Objektorientierte Programmierung  
mit C++" << endl;

    cout << " - Windows-Dektop-Programmierung  
mit C++" << endl;

    cout << endl;
    cout << "Aufgabenpool" << endl;
    cout << endl;
    cout << "Lernsituationen" << endl;
    cout << endl;
    cout << endl;
    return 0;
}
```

4. Auflage

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

Europa-Nr.: 85498

Verfasser:

Dirk Hardy, 46049 Oberhausen

Die in diesem Lehr- und Übungsbuch genannten Software-, Hardware- und Handelsnamen sind in der Mehrzahl auch eingetragene Warenzeichen.

Unter Verwendung von Screenshots aus:

Visual Studio Community Edition 2019, ©Microsoft

4. Auflage 2022

Druck 5 4 3 2 1

Alle Drucke derselben Auflage sind parallel einsetzbar, da sie bis auf die Korrektur von Druckfehlern identisch sind.

ISBN 978-3-8085-8730-0

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

© 2022 by Verlag Europa-Lehrmittel, Nourney, Vollmer GmbH & Co. KG, 42781 Haan-Gruiten
www.europa-lehrmittel.de

Satz: Reemers Publishing Services GmbH, Krefeld

Umschlag: braunwerbeagentur, 42477 Radevormwald

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

Druck: Plump Druck & Medien GmbH, 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 Anwendungsentwicklung 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. Die modernen Programmiersprachen sind objektorientiert, die Datenbankmanagementsysteme werden in Zukunft verstärkt objektorientiert sein. Mischformen wie objektrelationale Datenbanken sind schon länger 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 verschiedensten Plattformen
- Grundlage vieler weiterer Sprachen wie Java, Javascript, Perl oder auch PHP und C#

Die letzte ISO-Standardisierung der Sprache C++ war 2017. Damit verfügt C++ über viele neue Konzepte, die in anderen modernen Programmiersprachen (wie C#) bereits umgesetzt worden sind. Die vorliegende vierte Auflage dieses Buches nimmt diese Neuerungen in einem eigenen Kapitel auf und geht dann auch intensiver auf die STL (Standard Template Library) ein, die mit den Standards C++11, C++14 und C++17 weiter ausgebaut wurde. Weitere Standards sind auch schon geplant bzw. in einer Vorabversion verfügbar (C++20 sowie C++23/26). Damit wird C++ auch in Zukunft eine moderne Sprache sein.

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

Frühjahr 2022

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 **fünf Teile** getrennt. Die ersten drei Teile des Buches dienen als **Informationsteil** und bieten eine systematische Einführung in die **strukturierte Programmierung als auch in die objektorientierte Programmierung mit C++**. Abgerundet wird diese Einführung mit einem **Einstieg in die klassische Windows-Programmierung**, die eine hervorragende Grundlage für alle GUI-Programmierungen bietet.

Der vierte 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 **fünfte 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 Entwicklungswerkzeug wird in diesem Buch die **Community- Edition Visual Studio 2019** von Microsoft genutzt. Diese Entwicklungsumgebung ist kostenfrei als Download im Internet verfügbar.

Vorwort	3
Aufbau des Buches.....	4
Teil 1 Strukturierte Programmierung mit 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 anlegen.....	20
2.1.3 Das erste Programm.....	23
2.1.4 Erste Ausgabe auf dem Bildschirm	24
2.2 Grundlegende Konventionen in C++	24
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.....	27
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.....	80
7.1.3	Übergabe von Arrays an Funktionen.....	81
7.2	Zeichenketten in C++	83
7.2.1	Arrays vom Typ char	84
7.2.2	Funktionen zur Zeichenkettenbearbeitung	85
7.3	Sortieren von Arrays	86
7.3.1	Sortieren durch Auswahl	87
7.3.2	Der Bubblesort.....	89
8	Zeiger	92
8.1	Zeigervariablen	92
8.1.1	Deklaration eines Zeigers.....	92
8.1.2	Der Adressoperator	92
8.1.3	Der Dereferenzierungsoperator	93
8.2	Anwendungen von Zeigervariablen.....	94
8.2.1	Der call by reference	94
8.2.2	Zeiger und Arrays.....	95
8.2.3	Zeigerarithmetik.....	96
8.2.4	Zeiger auf Funktionen	98
8.2.5	Arrays von Zeigern	100
8.2.6	Dynamische Speicherreservierung	100
8.3	Die Referenz.....	104
8.3.1	Der Referenzoperator	104
8.3.2	Anwendung des Referenzoperators	105

9	Strukturen	106
9.1	Die Struktur in C++	106
9.1.1	Deklaration einer Struktur	106
9.1.2	Zugriff mit Operatoren.....	107
9.1.3	Strukturen in Strukturen.....	108
9.1.4	Arrays von Strukturen	109
9.2	Höhere Datenstrukturen.....	110
9.2.1	Die doppelt verkettete Liste	111
9.2.2	Der Binärbaum.....	113
Teil 2 Objektorientierte Programmierung mit C++		116
10	Das Klassenkonzept in C++	117
10.1	Die Klasse in C++	119
10.1.1	Aufbau einer Klasse in C++	119
10.1.2	Die Konstruktoren einer Klasse	121
10.1.3	Der Destruktor einer Klasse.....	124
10.1.4	Get- und Set-Methoden	125
10.2	Dynamische Speicherreservierung in Klassen	127
10.2.1	Die Klasse CKette	127
10.2.2	Call by value und der Copy-Konstruktor	129
10.3	Weitere Elemente einer Klasse	131
10.3.1	Der this-Zeiger	131
10.3.2	Statische Klassenelemente	132
10.4	Deklaration und Implementation bei Klassen	133
10.4.1	Header- und cpp-Datei.....	133
11	Dateioperationen	134
11.1	Ein- und Ausgabeströme	135
11.1.1	Eine Datei im Textmodus öffnen.....	135
11.1.2	Fehler bei Dateioperationen	137
11.1.3	Methoden der FileStream-Klassen.....	138
11.1.4	Eine Datei im Binärmodus öffnen	141
11.2	Wahlfreier Zugriff in Dateien	142
11.2.1	Positionieren des Dateizeigers	142
11.2.2	Lesen der Dateizeiger-Position.....	144
12	Das Überladen von Operatoren	145
12.1	Globale überladene Operator-Funktion	145
12.1.1	Die globale Operator-Funktion	145
12.1.2	Addition von Zeichenketten	146
12.1.3	Weitere Beispiele für globale Operator-Funktionen	148
12.2	Überladene Operatorfunktion als Methode	149
12.2.1	Überladener Operator als Methode	149
12.2.2	Addition von Zeichenketten	150
12.2.3	Weitere Beispiele für überladene Operatoren als Methoden.....	150
12.3	Überladen der Ein- und Ausgabeoperatoren	151
12.3.1	Überladene Operatoren der iostream-Klassen	151
12.3.2	Überladen des Eingabeoperators für eigene Klassen	152
12.3.3	Überladen des Ausgabeoperators für eigene Klassen.....	153
13	Vererbungskonzept in C++	154
13.1	Die einfache Vererbung	154
13.1.1	Umsetzung einer einfachen Vererbung in C++.....	155
13.1.2	Attribute als protected deklarieren.....	156
13.1.3	Aufruf der Basisklassenkonstruktoren	156
13.1.4	Weitere Formen der Vererbung	157
13.2	Die Mehrfachvererbung	159
13.2.1	Umsetzung der Mehrfachvererbung in C++.....	159
13.2.2	Virtuelle Vererbung	160
13.2.3	Aufruf der Basisklassenkonstruktoren	160

14	Polymorphismus und virtuelle Methoden	161
14.1	Zuweisungen innerhalb einer Vererbungshierarchie.....	161
14.1.1	Zuweisung von Objekten	161
14.1.2	Zeiger auf Basisklassen	162
14.2	Polymorphismus	162
14.2.1	Virtuelle Methoden	163
14.2.2	Regeln im Umgang mit virtuellen Methoden.....	164
14.2.3	Arrays von Basisklassenzeigern.....	164
14.2.4	Abstrakte Basisklassen.....	165
15	Fortgeschrittene Programmierung in C++	166
15.1	Templates	166
15.1.1	Funktionen-Templates	166
15.1.2	Klassen-Templates	167
15.2	Ausnahmen – Exceptions.....	168
15.2.1	Versuchen und Werfen – try und throw	169
15.2.2	Auffangen – catch	169
15.3	Die C++-Standardbibliothek	172
15.3.1	Die Klasse string	172
15.3.2	Intelligente Zeiger.....	175
15.3.3	Containerklassen	177
15.3.4	Algorithmen.....	179
15.3.5	Lambda-Ausdrücke.....	181
15.4	Nützliche Zusatzfunktionalitäten	183
15.4.1	Gelöschte Funktionen / Methoden	183
15.4.2	Methoden mit override und final kennzeichnen	183
15.4.3	Methoden mit const kennzeichnen	184
Teil 3	Windows-Desktop-Programmierung mit C++	186
16	Grundlagen der Windows-Desktop-Programmierung.....	187
16.1	Grundlagen der Windows-Programmierung	187
16.1.1	Grundkonzept der GUI-Programmierung.....	187
16.1.2	Aufbau einer Windows-Desktop-Anwendung und Nachrichtенbearbeitung.....	188
16.1.3	Eine Windows-Desktopanwendung anlegen.....	188
16.1.4	Die Quellcode-Dateien der Windows-Desktopanwendung.....	189
16.1.5	Die erste Windows-Desktopanwendung	192
16.2	Nachrichtенbearbeitung	197
16.2.1	Die Nachricht WM_PAINT	197
16.2.2	Die Nachrichten WM_CREATE und WM_SIZE	199
16.2.3	Tastatur- und Maus-Nachrichten	200
16.2.4	Die Nachricht WM_COMMAND	202
16.3	Text- und Grafikausgabe	202
16.3.1	Einfache Textausgabe mit DrawText.....	202
16.3.2	Texte positionieren mit TextOut	204
16.3.3	Punkte und Linien zeichnen	205
16.3.4	Rechtecke und Ellipsen zeichnen	206
17	Fortgeschrittene Windows-Desktop-Programmierung	207
17.1	Textausgabe und Bildlaufleisten	207
17.1.1	Schriftarten und Textausgabe.....	207
17.1.2	Bildlaufleisten programmieren.....	208
17.2	Windows-Standarddialoge und Menüs.....	211
17.2.1	Datei-Öffnen- und Datei-Speichern-Dialog	211
17.2.2	Schriftwahl-Dialog.....	213
17.2.3	Menübefehle ergänzen	215
Teil 4	Aufgabenpool	217
1	Aufgaben zum Umfeld der Sprache C++	218
2	Aufgaben zum ersten Programm in C++.....	218
3	Aufgaben zur Ein- und Ausgabe	219

4	Aufgaben zu Operatoren.....	220
5	Aufgaben zur Selektion und Iteration	222
6	Aufgaben zu Funktionen in C++	226
7	Aufgaben zu Arrays in C++	228
8	Aufgaben zu Zeigern.....	231
9	Aufgaben zu Strukturen	234
10	Aufgaben zu Klassen in C++	237
11	Aufgaben zu Dateioperationen mit C++	240
12	Aufgaben zur Überladung von Operatoren.....	242
13	Aufgaben zur Vererbung in C++	246
14	Aufgaben zu virtuellen Methoden	248
15	Aufgaben zur fortgeschrittenen Programmierung.....	249
16	Aufgaben zur Windows-Desktop-Programmierung	252
17	Aufgaben zur fortgeschrittenen Windows-Desktop-Programmierung.....	253

Teil 5 Lernsituationen 255

Lernsituation 1:	Erstellen einer Präsentation mit Hintergrundinformationen zu der Sprache C++ (in Deutsch oder Englisch)	256
Lernsituation 2:	Anfertigen einer Kundendokumentation für den Einsatz einer Entwicklungsumgebung in C++ (in Deutsch oder Englisch)	257
Lernsituation 3:	Entwicklung eines Verschlüsselungsverfahrens für ein internes Memo-System der Support-Abteilung einer Netzwerk-Firma.....	259
Lernsituation 4:	Entwicklung eines Informationstools für die Anzeige von Umgebungsvariablen.....	260
Lernsituation 5:	Planung, Implementierung und Auswertung eines elektronischen Fragebogens.....	262
Lernsituation 6:	Realisierung einer Klasse zur Speicherung von Messwerten	265
Lernsituation 7:	Implementierung einer Klasse zur Simulation der echten Bruchrechnung.....	267
Lernsituation 8:	Entwicklung eines einfachen Readers	269

Anhang A: Strukturierte Dokumentationstechniken..... 271

Der Programmablaufplan (PAP):	271
Beispiel eines Programmablaufplanes:.....	272
Das Struktogramm (auch Nassi-Shneiderman-Diagramm):	273
Beispiel eines Struktogrammes:	274

Anhang B: Index275

Teil Strukturierte Programmierung mit C++

1	Einführung in C++	13
1.1	Historische Entwicklung der Sprache C++.....	13
1.2	Bestandteile eines C++-Programms	15
1.3	Compiler, Linker und Bibliotheken.....	17
2	Das erste C++-Programm	20
2.1	Ausgabe auf dem Bildschirm	20
2.2	Grundlegende Konventionen in C++	24
2.3	Datentypen und Variablen.....	27
3	Ein- und Ausgabe in C++	32
3.1	Ausgabe mit cout.....	32
3.2	Eingabe mit cin	35
4	Operatoren in C++	38
4.1	Arithmetische Operatoren	38
4.2	Relationale und logische Operatoren.....	40
4.3	Bit-Operatoren und weitere Operatoren.....	42
4.4	Prioritäten von Operatoren	45
5	Selektion und Iteration	48
5.1	Die Selektion	48
5.2	Kopf-, fuß- und zählergesteuerte Iterationen	53
6	Funktionen in C++	58
6.1	Entwicklung des Funktionsbegriffs.....	58
6.2	Aufbau der Funktionen in C++	62
6.3	Modularer Programmaufbau	70
7	Arrays	77
7.1	Ein- und mehrdimensionale Arrays	78
7.2	Zeichenketten in C++	83
7.3	Sortieren von Arrays	86
8	Zeiger	92
8.1	Zeigervariablen	92
8.2	Anwendungen von Zeigervariablen.....	94
8.3	Die Referenz.....	104

9	Strukturen	106
9.1	Die Struktur in C++	106
9.2	Höhere Datenstrukturen.....	110

1 Einführung in C++

1.1 Historische Entwicklung der Sprache C++

1.1.1 Von C zu C++

Anfang der 70er-Jahre kam einem Programmierer des amerikanischen Telefonriesen AT&T die Idee, eine Programmiersprache zu entwickeln, mit der das Betriebssystem **UNIX**¹ einerseits programmiert, andererseits möglichst portabel werden sollte. Diese Programmiersprache sollte sowohl systemnah sein als auch die Vorteile einer höheren Programmiersprache enthalten. Damit wurde das Betriebssystem UNIX plattformunabhängig, denn für jede Rechnerumgebung wurde einfach der Quellcode von UNIX mitgeliefert, der dann auf den entsprechenden Rechnern nur noch übersetzt (kompiliert) werden musste.

Dieser Programmierer, **Dennis Ritchie**, nannte seine Neuentwicklung „C“. Er wählte diesen Namen als Fortführung der damals bekannten Programmiersprache „B“ bzw. „BCPL“. In Zusammenarbeit mit **Brian Kernighan** erstellte Ritchie das Handbuch zur Programmiersprache C², das 1978 in einer Revision mit einem **ANSI**³-Standard herausgebracht wurde.

Die Programmiersprache C++ ist nicht nur eine Weiterentwicklung von C, sondern beinhaltet auch eine andere Art der Programmierung – die **objektorientierte** Programmierung. Damit hat C++ einen großen Vorteil gegenüber C hinsichtlich der Reduzierung des Testaufwandes von Programmen und der Gewährleistung einer weitgehenden Fehlerfreiheit. Da C++ eine echte Obermenge von C ist, versteht der C++-Compiler auch jedes C-Programm, umgekehrt natürlich nicht.

1.1.2 Prozedurale, strukturierte und objektorientierte Programmierung

Der Anfang von C++ war eine Erweiterung der Sprache C um das Klassenkonzept. Es entstand die Sprache „**C mit Klassen**“. Als der Erfinder von C++ **Bjarne Stroustrup** 1983/84 dann seinen Neuentwurf der Sprache „C mit Klassen“ vorstellte, wurde von einem Mitarbeiter der Name C++ erfunden, basierend auf dem Inkrementoperator „++“. Damit sollte deutlich gemacht werden, dass dieses C++ eine echte Stufe höher als C ist, denn Inkrementieren bedeutet schließlich eins dazuzählen.

Das Klassenkonzept für C++ übernahm Stroustrup dabei von der Sprache **SIMULA67**. Weitere C++-Eigenheiten wie das Überladen von Operatoren sind an die Sprache **ALGOL68** angelehnt.

C++ wird auch als **Hybridsprache** bezeichnet, da C++ zwei **Programmierparadigmen**⁴ vereinigt – die **prozedurale** und die **objektorientierte** Programmierung.

Es wäre durchaus möglich, mit C++ rein prozedural zu programmieren, das würde aber den Fähigkeiten von C++ nicht gerecht.

Wie die meisten Programmiersprachen ist C++ auch eine strukturierte und prozedurale Programmiersprache.

Zum besseren Verständnis werden diese Begriffe kurz erläutert:

Strukturierte Programmierung

Die strukturierte Programmierung zeichnet sich durch Kontrollstrukturen wie die **Auswahl** (**IF-ELSE**) oder die **Schleifen** (**FOR**, **WHILE** usw.) aus. Damit erhält ein Programm eine nachvollziehbare Struktur. In den Anfängen der Programmierung war es üblich, Sprunganweisungen (**GOTO**) in einem Programm zu benutzen. Dadurch wird ein Programm sehr unübersichtlich und fehleranfällig. Strukturierte Programme sind hingegen übersichtlicher und besser wartbar. Im Informationsteil befassen sich die ersten neun Kapitel ausführlich mit der strukturierten Programmierung in C++.

¹ UNIX ist ein Betriebssystem aus den 60er-Jahren. Es ist ein leistungsfähiges Multiuser- und Multitasking-System. Es ist heute oftmals auf Großrechnern im Einsatz.

² Brian W. Kernighan/Dennis M. Ritchie: The C Programming Language, Prentice-Hall, 1978

³ ANSI: Abkürzung für American National Standard Institute. 1918 gegründet, ist ANSI mit dem deutschen DIN-Institut vergleichbar.

⁴ Paradigma kommt aus dem Griechischen und heißt so viel wie Muster oder Vorbild.

Beispiel:

```
FÜR Var := 1 BIS 5 MIT SCHRITTWEITE 1
    SCHREIBE AUF BILDSCHIRM Var
```

```
1
2
3
4
5
```

Das Beispiel zeigt eine Schleife in sogenanntem Pseudocode⁵. Dieser Code beschreibt den Ablauf des Programmes, ohne allerdings auf eine spezielle Programmiersprache einzugehen.

In dem Beispiel wird eine Variable Var so lange um 1 erhöht, bis der Wert 5 erreicht ist. Jeder Wert der Variablen wird dann auf dem Bildschirm ausgegeben.

Prozedurale Programmierung

Die prozedurale Programmierung teilt Programme in kleine Einheiten (Prozeduren oder Funktionen), die für bestimmte Aufgaben verantwortlich sind. Sind diese Prozeduren einmal geschrieben und getestet, dann können sie immer wieder benutzt werden – das spart Entwicklungszeit und führt auch zu einer besseren Lesbarkeit des Programms.

Der Informationsteil geht speziell in Kapitel 6 (Funktionen) auf die prozedurale Programmierung ein.

Beispiel:

```
PROZEDUR Ausgabe
    SCHREIBE AUF BILDSCHIRM "Hallo"
ENDE

FÜR Var := 1 BIS 5 MIT SCHRITTWEITE 1
    AUFRUF Ausgabe
```

```
Hallo
Hallo
Hallo
Hallo
Hallo
```

Das Beispiel in Pseudocode zeigt eine Prozedur mit dem Namen Ausgabe. Diese Prozedur hat eine Anweisung, die das Wort „Hallo“ auf den Bildschirm schreibt. Die bereits bekannte Schleife aus dem Beispiel vorher läuft dann 5-mal durch und ruft jedes Mal die Prozedur Ausgabe auf. Damit steht 5-mal das Wort „Hallo“ auf dem Bildschirm.

Objektorientierte Programmierung

Die objektorientierte Programmierung möchte Objekte der realen Welt in einem Programm abbilden. Damit sollen Problemstellungen aus beliebigen Bereichen (Geschäftsprozesse, wissenschaftliche Untersuchungen usw.) geeigneter als mit den anderen Programmierparadigmen in Programme umgesetzt werden können.

Im Mittelpunkt der objektorientierten Programmierung steht die **Klasse**, aus der dann konkrete Objekte gebildet werden.

Beispiel:

```
KLASSE Kunde
    Name
    Telefon
ENDE

BILDE OBJEKT K1 VON Kunde
K1.Name := "Maier"
```

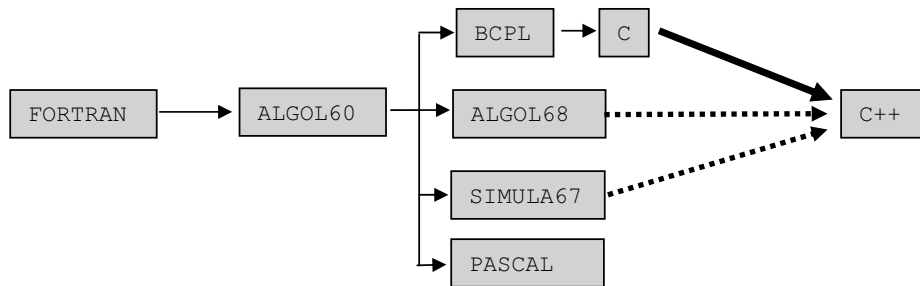
In dem Beispiel wird ein Klasse Kunde angelegt. Von dieser Klasse können dann konkrete Objekte wie K1 (für Kunde 1) gebildet werden. Die Eigenschaften des Objektes (Name, Telefon) können dann mit Werten belegt werden. In diesem Beispiel erhält das Objekt K1 den Namen „Maier“.

Die objektorientierte Programmierung steht ab dem Kapitel 10 im Mittelpunkt des Informationsteils.

⁵ Pseudocode ist eine Art Sprache, mit der der Ablauf eines Programmes beschrieben wird. Pseudocode zeichnet sich dadurch aus, dass er näher an der natürlichen Sprache als eine Programmiersprache ist. Ein Programm, das in Pseudocode geschrieben ist, kann problemlos in jede Programmiersprache übersetzt werden.

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.

Auf den Unterschied zwischen den Dateiendungen wird in Kapitel 6 genauer eingegangen.

Beispiel: Test.cpp

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

class Test
{
private:
    int x;
public:
    Test();

Test::Test()
{
    x = 0;
}

int main ()
{
    Test A;
    Test B;
    return 0;
}
```

Die Datei `Test.cpp` ist eine Quellcode-Datei. Sie enthält Anweisungen in der Sprache C++.

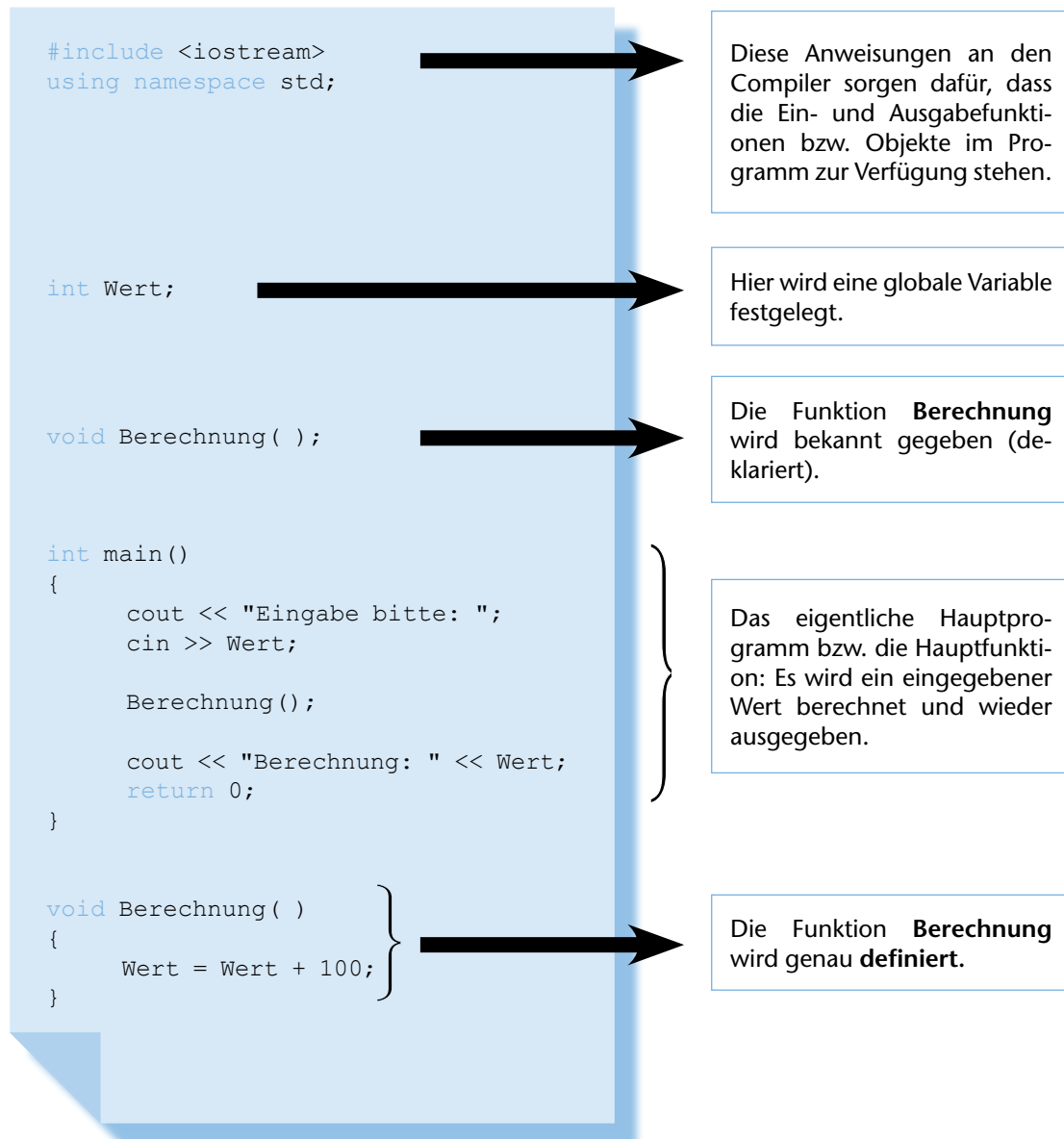
1.2.3 Grundsätzlicher Aufbau eines C++-Programmes

C++-Programme können sehr unterschiedlich aussehen, je nachdem, wer programmiert hat. Die meisten Programmierer halten sich jedoch an gewisse Konventionen, um den Quellcode leserlicher und leichter wartbar zu machen.

Grundsätzlich ist ein C++-Programm nach folgendem Muster aufgebaut:

Compileranweisungen - das sind Anweisungen an das Programm, das den Quellcode übersetzt.
Festlegen der allgemein gültigen (globalen) Variablen
Bekanntgabe der zu verwendenden Funktionen (Prozeduren)
Das eigentliche Programm (Hauptprogramm)
Definieren der verwendeten Funktionen (Prozeduren)

In einem C++-Quellcode könnte die Struktur so aussehen:



Die Bedeutungen der einzelnen C++-Anweisungen werden in den nächsten Kapitel deutlich.

1.3 Compiler, Linker und Bibliotheken

1.3.1 Der Compiler

Der Compiler ist ein Programm, das den Quellcode einer Programmiersprache in **Maschinencode** übersetzt.

Dazu analysiert der Compiler den Quellcode systematisch auf syntaktische Fehler. Die **Syntax** der Programmiersprache ist wie die Grammatik in unserer Sprache. Es gibt klare Regeln, nach denen die Sprache angewendet werden kann. Der Compiler deckt Verstöße gegen diese Regeln auf.

Nach weiteren Überprüfungen wie der korrekten Verwendung von Variablen erstellt der Compiler dann den Maschinencode. Bei einem C++-Compiler wird allerdings eine sogenannte **Objektdatei** erstellt, die dann im Zusammenspiel mit dem Linker (siehe 1.3.2) zu einem ausführbaren Programm führt.

1.3.2 Der Linker

Der Linker ist ein Programm, das verschiedene Module zu einem ausführbaren Programm verbindet.

Beispielsweise verbindet (linkt) das Programm den durch den Compiler erstellten Objektcode mit weiterem Code aus Bibliotheken (siehe 1.3.3). Das ist sinnvoll, um Funktionen oder Prozeduren, die bereits geschrieben worden sind, in das eigene Programm einzubinden, ohne die Funktionalitäten neu schreiben zu müssen.

Beispielsweise sind die Funktionen, die die Ein- und Ausgabe über Tastatur und Bildschirm ermöglichen, bereits vorhanden und können in jedem Programm genutzt werden. Der Linker muss nur den richtigen Objektcode hinzubinden.

Man unterscheidet beim Linken das **statische** und **dynamische** Linken. Das statische Linken ist die feste Verdrahtung der Programmteile zu einer ausführbaren Datei.

Beim dynamischen Linken wird während der Laufzeit des Programms der nötige Objektcode hinzugelinkt. Das ist unter Umständen langsamer als das statische Linken, aber hat den Vorteil, dass die Programme selbst kleiner sind und immer nur das anfordern, was benötigt wird. Für das dynamische Linken stehen DLL-Dateien (Dynamic Link Libraries) zur Verfügung.

1.3.3 Bibliotheken

Bibliotheken sind Sammlungen von Programmteilen. In der Regel sind Funktionen und Prozeduren, die ähnliche bzw. zusammengehörende Aufgaben erfüllen, in einer Bibliothek zusammengefasst. Es gibt unzählige Bibliotheken zur Sprache C++.

Wichtig für das Programmieren in C++ ist zuerst die **C++-Standardbibliothek**.

Diese liefert Funktionalitäten, die für die Erstellung eines C++-Programms unerlässlich sind.

Beispiele:

- ▶ Ein- und Ausgabe-Funktionalität (Tastatur, Bildschirm, Drucker usw.)
- ▶ Zeichenkettenverarbeitung
- ▶ Dateioperationen
- ▶ Mathematische Funktionen

Im Internet sind viele frei verfügbare Bibliotheken zu finden, die für bestimmte Aufgabenstellungen die fertigen Funktionen anbieten.

Beispiele:

- ▶ Kompressionsverfahren
- ▶ Datenbankzugriffe
- ▶ Numerische Berechnungen
- ▶ Grafik-Funktionen

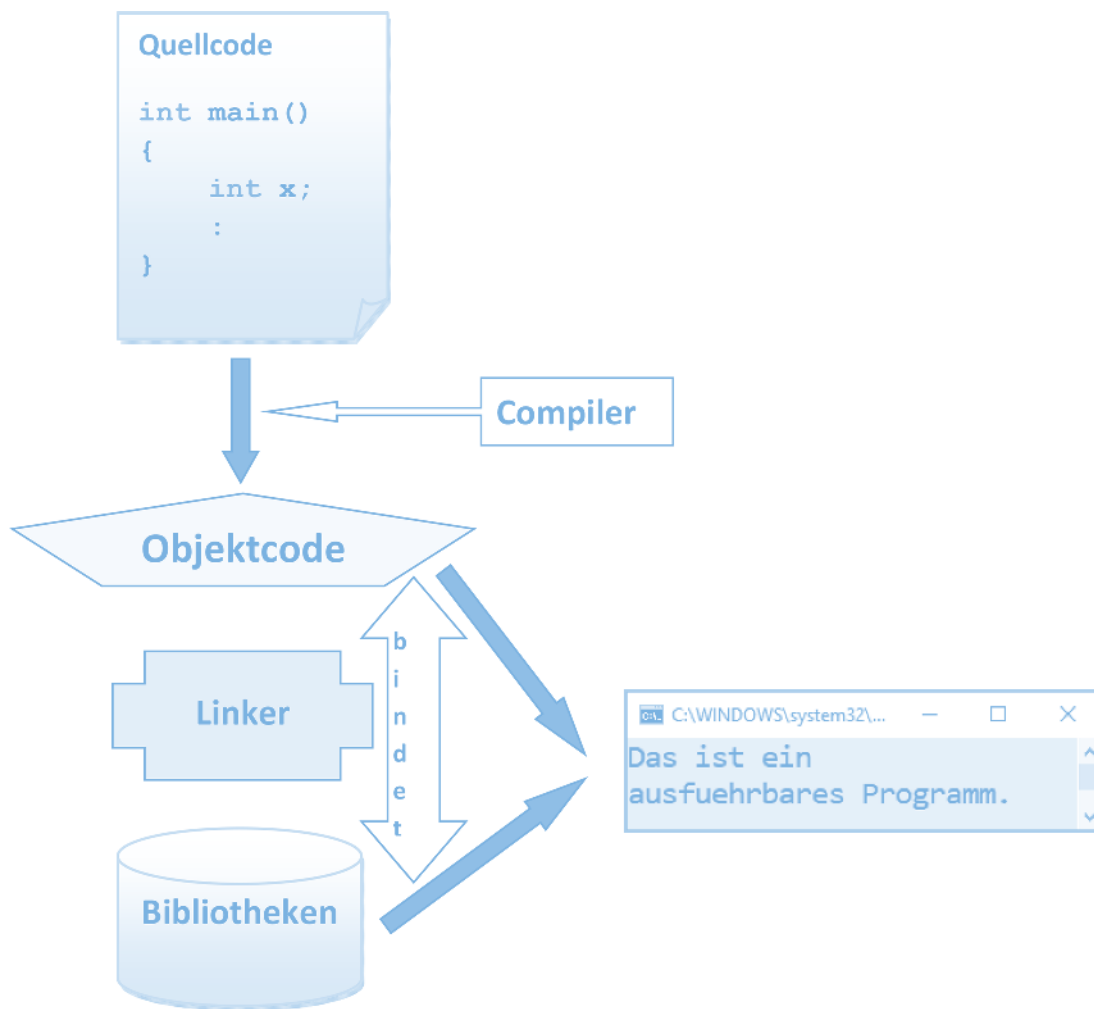
1.3.4 Schematischer Ablauf einer Programmerstellung

Das Ausführen von Compiler und Linker und das Bereitstellen von Bibliotheken zur Erstellung eines lauffähigen Programms wird dem Entwickler durch die sogenannten **Integrierten Entwicklungsumgebungen (IDE, engl. Integrated Development Environment)** abgenommen. Diese Umgebungen sind selbst Programme. Sie integrieren Compiler, Linker und Bibliotheken. Der Entwickler braucht sich nicht mehr darum zu kümmern, dass sein Quellcode kompiliert und dann gelinkt wird. Das geschieht automatisch durch die IDE.

Bekannte Beispiele für Integrierte Entwicklungsumgebungen sind:

- ▶ Microsoft Visual C++
- ▶ Netbeans
- ▶ Eclipse

In diesem Buch wird die kostenfreie Entwicklungsumgebung Visual Studio (Community Edition) genutzt.

Schematischer Ablauf:

Der Quellcode wird vom Compiler in den Objektcode übersetzt. Dieser Code wird dann vom Linker mit den entsprechenden Funktionen (Routinen) aus den Bibliotheken zu einem ausführbaren Programm (in der Regel eine „.exe-Datei“) verbunden (gelinkt).

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 anlegen

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

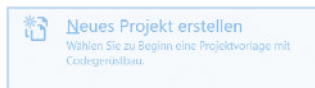
Visual C++ unterscheidet im Prinzip folgende Projektarten:

- Windows-Desktopanwendungen (eine Windows-Anwendung)
- **Windows-Konsolenanwendung / leere Anwendung (ähnlich einem DOS-Programm)**
- CMake-Projekte – (moderne, plattformübergreifende Apps)
- Dynamic Link Library (Bibliotheksanwendung)

In diesem Buch ist hauptsächlich eine Projektform von Bedeutung: **die Windows-Konsolenanwendung bzw. die leere Anwendung**. Diese Form ist ausreichend, um die Programme in C++ auf dem Bildschirm darzustellen. Die Konsolenanwendung ist natürlich nicht so ansprechend wie ein Windows-Programm oder eine App, aber, um die Sprache C++ zu lernen, völlig ausreichend. Zum Abschluss des Buches wird die Windows-Desktopanwendung eingeführt, mit der klassische Windows-Programme realisiert werden können.

Anlegen eines neuen Projektes:

- Starten Sie **Visual Studio 2019**
- Wählen Sie



- Danach öffnet sich ein neues Fenster: Wählen Sie

