EUROPA-FACHBUCHREIHE
Programmierung für die
IT-Ausbildung

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
Düsselberger Str. 23 42781 Haan-Gruiten

Europa-Nr.: 85498

Verfasser:
Dirk Hardy, 46049 Oberhausen

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

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.

Dirk Hardy Frühjahr 2022

1 Teil
Strukturierte Programmierung mit C++

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 C[2] , 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 Pseudocode[5]. 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 (Geschäftprozesse, 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:

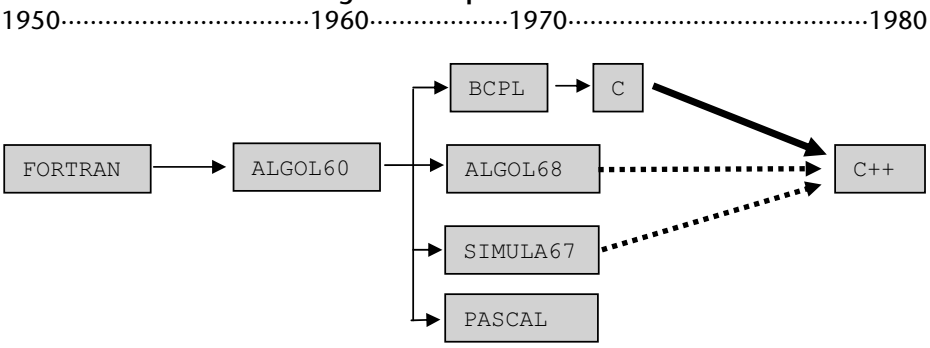
```
KLASSE Kunde
       Name
       Telefon
ENDE

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



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++.

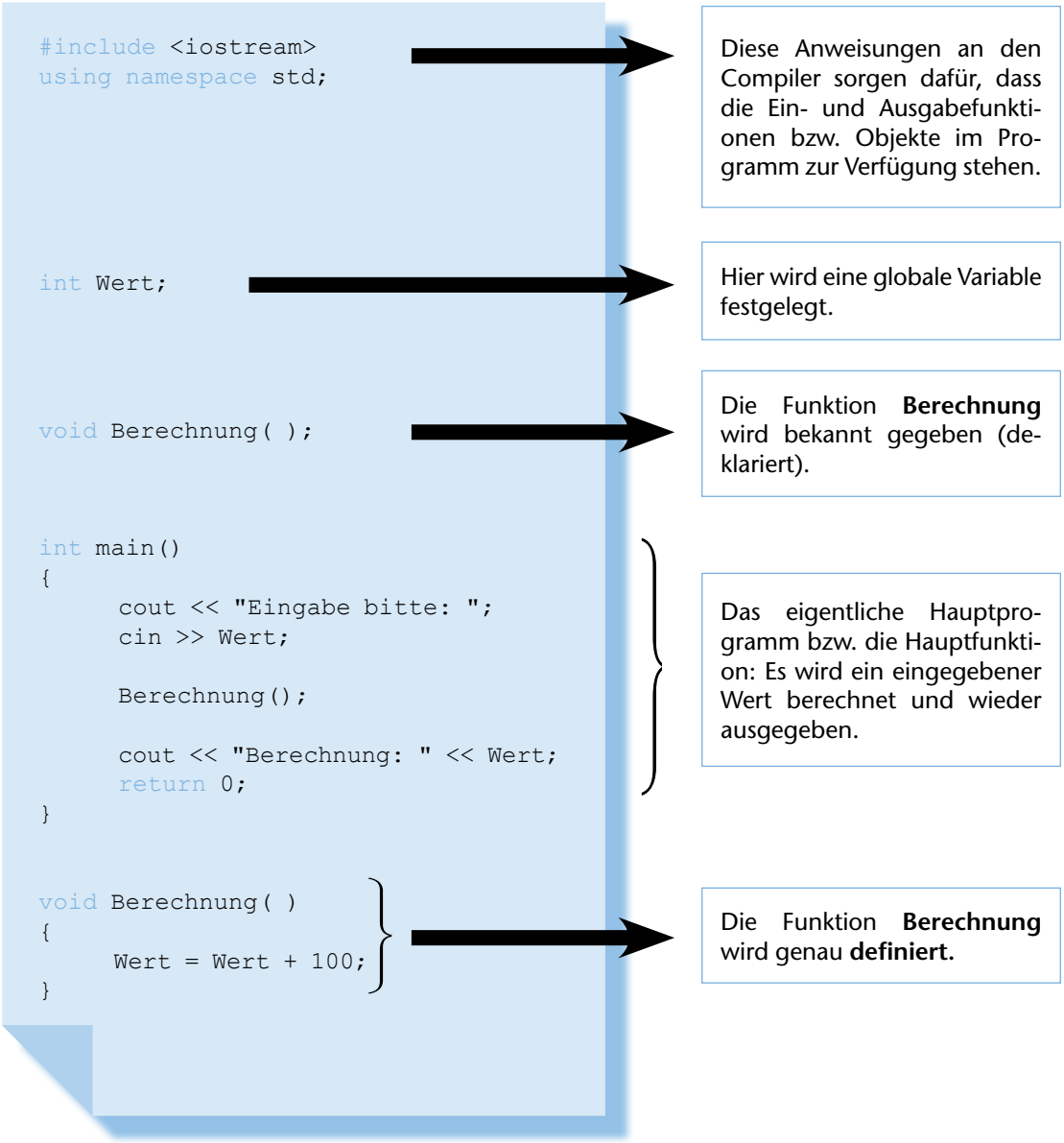
1.2.3 Grundsätzlicher Aufbau eines C++-Programmes

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

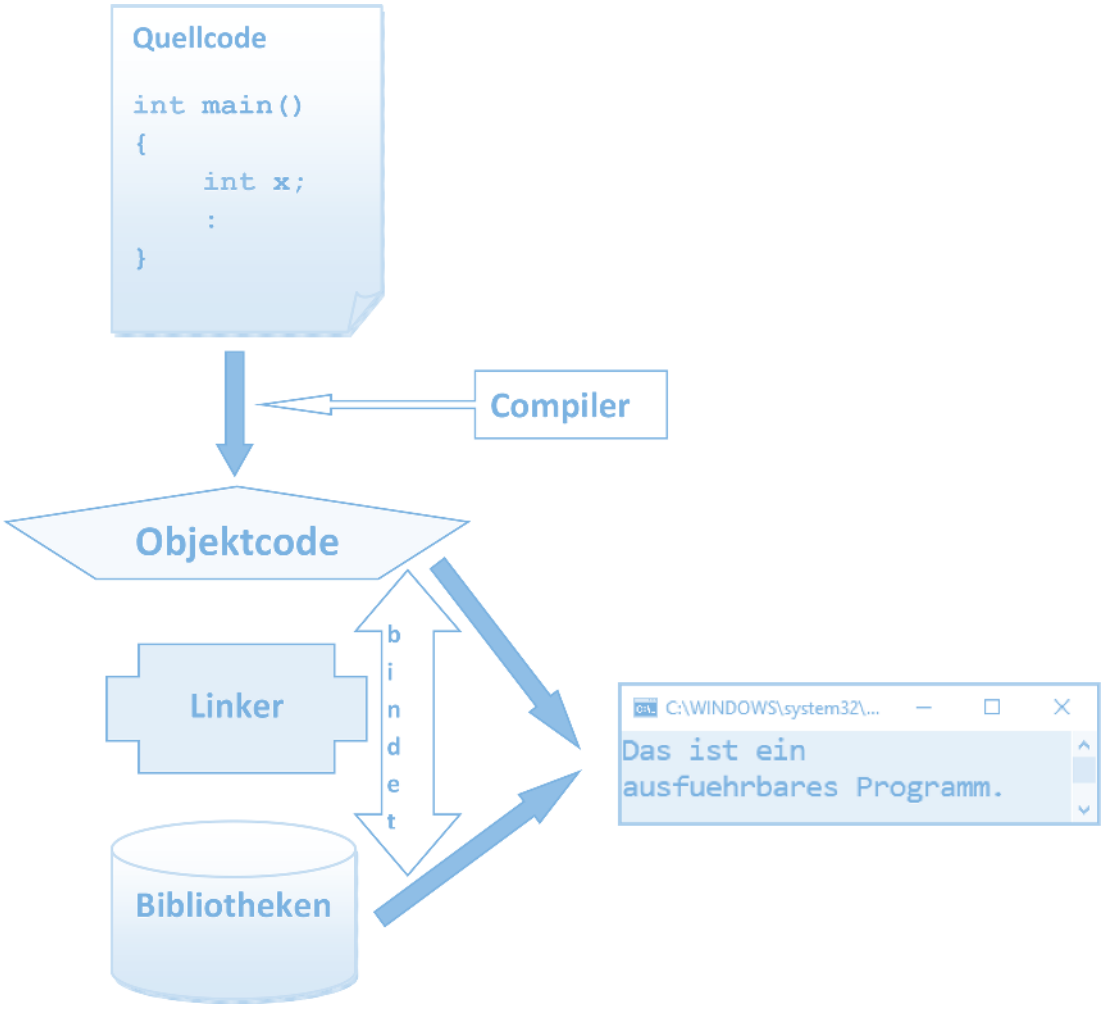
Das Ausführen von Compiler und Linker und das Bereitstellen von Bibliotheken zur Erstellung eines lauffähigen Programms wird dem Entwickler durch die sogenannten **Integrierten Entwicklungsumgebungen (IDE, engl. Integrated Development Environment)** abgenommen. Diese Umgebungen sind selbst Programme. 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