lichst großes Spektrum von Fragestellungen und damit verbundenen Problemlösungen aufzuzeigen.

Die Kapitel werden im Folgenden kurz skizziert:

Kapitel 1
Im ersten Kapitel stehen Gesichtspunkte, die in einer hochvernetzten und globalisierten Arbeitswelt den Umgang mit zu lösenden Problemen diktieren, im Fokus der Betrachtung. Längst sind die Zeiten vorbei, in denen sich ein Softwareentwickler so intensiv mit seinem Problem auseinandersetzen konnte, bis er sicher war, eine optimale und anwendungsbreite Lösung gefunden zu haben. Die Maxime heute lautet: in kürzester Zeit ein brauchbares Ergebnis zu liefern.

Kapitel 2
Dieses Kapitel beschäftigt sich mit einigen grundlegenden Aspekten der Programmiersprache C/C++. Neben der Frage, warum es sinnvoll ist, gerade mit C/C++ zu arbeiten, werden Funktionsweisen der Komponenten der Entwicklungsumgebung betrachtet und erläutert.

Kapitel 3
Im dritten Kapitel wird die Arbeitsweise von VC++ und die Arbeit mit dieser Entwicklungsumgebung beschrieben. Der Leser erfährt, wie ein komplexeres Projekt organisiert wird.


Auf Installationhinweise auf zur jeweils aktuellen Version von VC++ haben wir absichtlich verzichtet, da Microsoft in sehr kurzen Zyklen neue Versionen zur Verfügung stellt, deren Installationen keinerlei Probleme darstellen. Die Kompatibilität von eingesetztem Betriebssystem und verwendeter Version der Entwicklungsumgebung von VC++ wird auf den Internet-Seiten von Microsoft dargestellt (s. a. https://msdn.microsoft.com/de-de).

Kapitel 4
In diesem Kapitel werden die grundlegenden Sprach- und Steuerungselemente der Syntax der Programmiersprache C/C++ an einfachen Beispielen dargestellt.

Kapitel 5
Dieses Kapitel enthält eine Einführung in die strukturierte Programmierung und ihre Darstellungsformen. Der Leser lernt rechnergestützte Systeme zur Erstellung von Struktogrammen kennen, die an Beispielen zunehmender Komplexität beschrieben werden. Außerdem werden die Bestandteile einer Software-Dokumentation beschrieben und die Frage beantwortet, was Software mit Qualität zu tun haben sollte.

Kapitel 6
Im sechsten Kapitel werden die Kenntnisse der strukturierten Programmierung zunächst an einfachen Problemen angewendet. Von der Problemanalyse bis zur Ergebnisausgabe der Programme sind die Beispiele durchgängig dokumentiert.

Kapitel 7
Dieses Kapitel gibt eine Einführung in die objektorientierte Programmierung. Der Leser lernt das erweiterte Vokabular und die Techniken der OOP kennen.

Kapitel 8
Mit den Grundkenntnissen der OOP können in diesem Kapitel fortgeschrittene Probleme unter Zuhilfenahme von grafischen Oberflächen gelöst werden. Die Darstellung der Ergebnisse geschieht hier teilweise auch in zeichnerischer Form.

Kapitel 9
Im neunten Kapitel werden komplexe Fragestellungen durch konsequente Anwendung aller erlernten Techniken bearbeitet. Die erzielten Lösungen sind dabei nach entsprechend gründlicher Problemanalyse verblüffend einfach.

Kapitel 10
In Kapitel 10 finden Sie Tabellen und Übersichten, es stellt eine hilfreiche Zusammenfassung der für die Problemlösung erforderlichen Teilgebiete dar. Neben Datentypen, Symbolen für die Erstellung von Struktogrammen und Programmablaufplänen enthält es ein Glossar für den Sprachumfang von C und C++ und eine Tabelle oft genutzter Standardfunktionen.

Außerdem finden Sie im Internet alle Code-Beispiele der Kapitel 6 bis 9 unter: *http://www.hanser-fachbuch.de/buch/Technische+Probleme+loesen+mit+C+C/9783446463080* auf der Homepage des Hanser Verlags. Die Beispiele sind hier sowohl als PDF-Dateien mit den durchnummerierten Zeilen, auf die in den Erläuterungen verwiesen ist, als auch als Quelltexte, die Sie ohne mühsames Abtippen sofort in eigene Programme integrieren können, vorhanden. So lassen sich alle Beispiele unmittelbar erzeugen und ausführen. Einige Beispiele der Kapitel 8 und 9 finden Sie aufgrund ihrer Länge nur im Internet.

Zielgruppe des Buchs:
Das Buch wendet sich in erster Linie an Studierende an Fachschulen, Studenten technischer Studiengänge sowie an Auszubildende in den IT-Berufen. Aber auch Schülerinnen und Schüler in technisch orientierten Gymnasien und Fachoberschulen werden an der Vielfalt der Problemstellungen und der Herangehensweise an die Lösung – vom Problem über die Problemanalyse mit Struktogramm bis zum Testing und zur Dokumentation – sicherlich Gefallen finden.

Es liegt ein zeitgemäßes Buch vor. Zeitgemäß deshalb, weil wir den wichtigsten Schritt in der Programmentwicklung, die Problemanalyse, in den Mittelpunkt der Entwicklungsarbeit stellen und erst danach die Umsetzung mit VC++ besprechen. Denn ohne eine gründliche Problemanalyse haben Sie später keine Chance, logische Programmfehler zu finden. Übrigens verdient man in der Wirtschaft in diesem Bereich das meiste Geld.

Zeitgemäß ist das Buch auch dadurch, dass Sie ellenlange Quellcodes nicht mehr abtippen müssen, sondern einfach aus dem Internet herunterladen können. Das vereinfacht den Programmtest und die individuelle Fortentwicklung der Programme ganz gewaltig.

Um Ihnen die Orientierung im Buch zu erleichtern, haben wir Icons verwendet, die folgende Bedeutung haben:

	Bei den **Beispielen** im Buch finden Sie dieses Symbol.
	Wichtige **Hinweise** werden durch dieses Icon kenntlich gemacht.
	Dieses Symbol kennzeichnet **Aufgaben**.
	Die Darstellungen von **Lösungen** sind an diesem Icon zu erkennen.
	Hinweis auf einen Teil im **Internet**.

Zum Abschluss bleibt uns noch all den Personen zu danken, die uns bei der Arbeit an diesem Buch unterstützt haben. Ihnen als Leserin und Leser wünschen wir viel Erfolg bei der Arbeit mit diesem Buch und dem Lernen und Ausprobieren mit und von Visual C++.

Norbert Heiderich und Wolfgang Meyer

1 Systematik der Problemlösung

Einst löste Alexander der Große den Gordischen Knoten sehr unkonventionell mit dem Schlag seines Schwertes. An den kunstvoll geknoteten Stricken, die einen Streitwagen untrennbar mit seinem Zugjoch verbinden sollten, waren zuvor die Gelehrten gescheitert. Sie versuchten, ihn ohne Beschädigung zu entfernen, quasi die Verknotungen umzukehren. Dies zeigt deutlich, dass ein Problem komplex und damit sogar unlösbar werden kann, wenn man nicht fähig ist, es unvoreingenommen zu betrachten, wenn man sich nicht von unvermeidbar erscheinenden Lösungswegen trennen kann. Die Lösung des Problems soll das Ziel sein – aber auch der Weg dorthin!

Zur Lösung eines Problems mit Hilfe eines Rechners geht man üblicherweise in mehreren Einzelschritten vor. Diese Vorgehensweise ist sinnvoll, weil die in jedem Schritt anfallenden Probleme häufig so speziell sind, dass Fachleute des jeweiligen Gebietes sie lösen müssen. So muss z. B. ein Betriebsführer, der eine Problemstellung sehr genau aus der Sicht des Betriebsablaufes beschreiben und sicherlich aus dieser Sicht auch erste Strategien entwickeln kann, nicht notwendigerweise auch derjenige sein, der mögliche Auswirkungen auf die Buchführung und Abrechnung des Unternehmens beurteilen, oder zur Auswahl geeigneter Programmierelemente und einzusetzender Hardware einen Beitrag leisten kann.

1.1 Phasen der Programmentwicklung

In den Anfängen der Datenverarbeitung waren Systemanalyse und methodisches Vorgehen bei der Entwicklung von Software beinahe bedeutungslos und der heute gebräuchliche Begriff **Softwareengineering** war noch nicht geprägt. Die erste Phase des Softwareerstellungsprozesses ist die Systemanalyse. Der Systemanalytiker beschreibt hier die für seine Fragestellung relevanten Elemente und deren Beziehungen zueinander.

Die ersten Rechner waren von den Abmessungen her groß und von der Leistungsfähigkeit aus heutiger Sicht sehr bescheiden. Hardware war so teuer, dass kleinere Unternehmen in der Regel die Verarbeitung ihrer Daten Service-Rechenzentren übergaben. Diese Rechenzentren entwickelten und warteten auch die individuellen Programme ihrer Kunden. Die eigene Datenverarbeitung im Hause bedeutete immense Investitionen, und die Software wurde dann mehr oder weniger individuell um die vorhandene Hardware „gestrickt“.

Die steigende Leistungsfähigkeit und der Preisverfall mit jeder neuen Generation von Rechnern eröffneten nach und nach immer neue Einsatzgebiete. So konnte man zunehmend integrierte Systeme entwickeln. Allerdings wurden mit dem wachsenden Integrationsgrad der Software die Programme und Programmsysteme komplexer.

Betrachtet man zu den Anfängen der Datenverarbeitung in mittleren bis großen Unternehmen das Verhältnis der Kosten von Hard- zur Software, so lag die bei etwa 85:15. Die gleiche Bewertung liefert heute ein Verhältnis von 10:90. Vergleicht man das Kostenverhältnis der Hard- zur Software im PC-Bereich, so ergibt sich für einen normalen Anwender in einem kleinen bis mittleren Betrieb ein ganz anderes Bild. Hier liegt das Verhältnis nahezu bei 50:50.

Der Einsatz von Datenverarbeitung in neuen Anwendungsgebieten ist primär ein Problem der Qualität, Funktionalität und Verfügbarkeit der Software zum richtigen Zeitpunkt und zu einem vertretbaren Preis. Damit wird deutlich, dass die Entwicklung von Software ein hochkomplexes Unterfangen ist und ein abgestimmtes, methodisches Verfahren und organisatorisches Vorgehen verlangt. Zusammengefasst wird dies unter dem Begriff Softwareengineering.

Softwareengineering wurde als Vorgehensweise zur Verbesserung der bis dahin unbefriedigenden Situation bei der Softwareentwicklung und -wartung betrachtet. Software sollte produziert werden können wie Produkte aus der industriellen Fertigung: solide, zuverlässig und kontrollierbar. Aus diesen Anfängen entwickelte sich die heutige Definition:

Unter **Softwareengineering** versteht man die Anwendung von Strategien, Methoden, Werkzeugen und Kontrollinstrumenten im gesamten Prozess der Softwareentwicklung und -wartung einschließlich des Managements.

Die Beschäftigung mit Softwareengineering setzt nun einen gewissen Erfahrungsschatz in der Softwareentwicklung voraus. Bei der **Softwareentwicklung im Kleinen** geht es um die Umsetzung überschaubarer Problemstellungen in rechnergestützte Lösungen. Dem Anwender der fertigen Software sollen möglichst viele, von ihm bisher evtl. mit anderen Hilfsmitteln erledigte Arbeitsschritte durch einen Rechner abgenommen werden. Dabei stehen die Auswahl und das Design einzelner Konstrukte im Vordergrund, was für die korrekte Funktionsweise und das spätere Verständnis eines Bausteins absolut wesentlich ist. Bei der **Softwareentwicklung im Großen** geht es um die zweckmäßige, fast generalstabsmäßige Organisation eines Arbeitsvolumens von vielen Mann-Jahren. (In der Informatik wird der Begriff Mann-Tage, Mann-Monate oder Mann-Jahre als Aufwandsmaß eines abstrakten Wesens verwendet, das während seiner Arbeitszeit weder männlich noch weiblich ist.)

In manchem ist das Softwareengineering mit der Arbeitsorganisation in herkömmlichen Produktions- und Konstruktionsprozessen vergleichbar. Softwareengineering beschäftigt sich mit Arbeitsabläufen in und um die Softwareentwicklung herum. Neben dem eigentlichen Entwicklungsprozess sind dies:

- Projektmanagement,
- Qualitätssicherung und
- Projektverwaltung.