

Inhalt

Kapitel 1 – Einführung in objektorientierte Sprachen	3
Grundlagen und Konventionen	3
Warum objektorientierte Sprachen?	5
Zur Entstehungsgeschichte objektorientierter Sprachen	5
Experiment und Struktur	7
Der Prozeß der Übersetzung	8
Wie Compiler arbeiten	9
Wozu braucht man Objekte?	12
Die Vorteile der Erbllichkeit	15
Definition einer objektorientierter Sprache	17
C++ versus Smalltalk	19
Zusammenfassung	19
Kapitel 2 – Vordefinierte Klassen anwenden	21
Tools für separates Compilieren	22
Das erste C++-Programm	29
make – ein wesentliches Tool für die separate Compilation	34
Mehr über streams	36
Kontrollstrukturen in C und C++	39
Eine kurze Einführung in C- und C++-Operatoren	46
Standard I/O für Dateizugriffe verwenden	48
Utilities für Streams und Standard I/O	52
Das Makefile für dieses Kapitel	54
Zusammenfassung	56

Kapitel 3 – Klassen in C++ erzeugen	57
Daten in C++	57
Sichtbarkeitsregeln	60
Wie für Variable Speicherplatz reserviert wird	62
Die Operatoren in C und C++	70
Funktionen in C und C++ erzeugen	81
Besonderheiten von C++-Funktionen	85
Die Klasse: Zugriffsbeschränkungen programmieren	90
Das Header-File	97
Member-Funktionen definieren	104
Andere class-ähnliche Konstrukte	112
Hinweise zum Debugging	117
Das Makefile für die Programme dieses Kapitels	122
 Kapitel 4 – Pointer und Referenzen	 125
Adressen sind wie Briefkästen	125
Pointer	126
Mit Pointern und Adressen arbeiten	129
Pointer-Größen (Speichermodelle)	149
Funktions-Adressen	152
Anwendungsbeispiele für Pointer	158
Das Konzept der Referenz	167
Wann sollten Referenzen verwendet werden?	180
Das Makefile für dieses Kapitel	184
 Kapitel 5 – Overloading	 187
Die Syntax des Operator-Overloading	189
Beispiele für Operator-Overloading	189
Operatoren für die Typkonversion definieren	203
Beispiel: das Erzeugen eigener stream-Funktionen	215
Die Wahl zwischen Friend- und Member-Funktionen beim Operator-Overloading	217
Overloading mit Funktionen	221
Das Makefile	224

Kapitel 6 – Die Erzeugung von Objekten zur Laufzeit	225
Tendenzen einiger populärer Programmiersprachen	226
Dynamische Erzeugung von Objekten	227
Objekte veränderbarer Größe	243
Der Mechanismus dynamischer Objekterzeugung	254
Das Verhalten von new und delete verändern	261
Konstruktoren, die an »this« zuweisen	270
Das Makefile für dieses Kapitel	278
Kapitel 7 – C++-Code wiederverwenden	279
Member-Objekte	281
Erblichkeit	289
Objekte auf Diskette speichern	310
Eine Liste, die sich selbst sichert und restauriert	322
Der Zugriff auf die Elemente der Basisklasse	332
Das Makefile für dieses Kapitel	336
Kapitel 8 – Erweiterbare Programme	339
Die Transport-Hierarchie in C++ implementieren	341
Virtuelle Funktionen	347
Ein erweiterbares Menu-System	354
Abstrakte Klassen	371
Simulation mit abstrakten Klassen	379
Das Makefile für dieses Kapitel	386
Kapitel 9 – Argumente und Funktionswerte	389
Namen-, Wert- und Referenzübergaben	390
Verborgene Aktivitäten	392
Der Copy-Initializer X(X&)	395
Objekte zuweisen	399
Ein Beispiel für Deep-Kopie: class matrix	402
Objekte mit *new zurückliefern	414
Das Makefile für dieses Kapitel	418

Kapitel 10 – Komplette Beispiele	421
Kommandozeilen-Argumente, Flags und Dateien	421
Ein Programm für das Listing	432
TAWK: Ein einfacher Datenbank-Interpreter	436
Ein Uhrzeit-orientiertes Kontrollsystem	459
Kapitel 11 – C++ – Version 2.0	487
Fortdauernde Inkompatibilitäten mit ANSI C	488
Mehrfache Erbllichkeit	494
Type-Save Linkage	498
Globale Operatoren new und delete	505
Neue redefinierbare Operatoren	507
Pointer auf Member-Objekte und -Funktionen	509
Neue Qualifizierer für Member-Funktionen	511
Neue Formen der Initialisierung	513
Exception-Handling	520
Verschiedene geringe Veränderungen des Sprachstandards	521
Anhang A – MicroCAD	527
Der MicroCAD-Code	527
Das Makefile für MicroCad	545
Anhang B – Die Matrix-Klasse	547
Standard-Matrix-Dateien	548
Der Code für class matrix	549
Das Makefile für class matrix	565
Anhang C – Windows	567
Der Code für class window	568
Vorschläge für Verbesserungen	576
Das Makefile für Anhang C	576
Index	577