



it
informatik

Bjarne Stroustrup

Einführung in die Programmierung mit C++

Inhaltsverzeichnis

Vorwort zur amerikanischen Ausgabe	27
Hinweis an die Studenten	30
Hinweis an die Lehrkräfte	31
Unterstützung	31
Danksagungen	32
Vorwort zur deutschen Ausgabe	33
Zur Handhabung des Buches	33
Für Dozenten und Studenten	34
Companion Website	35
Kapitel 0 Hinweise für den Leser	37
0.1 Die Struktur dieses Buches	37
0.1.1 Allgemeiner Ansatz	38
0.1.2 Aufgaben, Übungen usw.	40
0.1.3 Was kommt nach diesem Buch?	41
0.2 Eine Philosophie des Lehrens und Lernens	42
0.2.1 Die Reihenfolge der Themen	45
0.2.2 Programmierung und Programmiersprache	46
0.2.3 Portabilität	47
0.3 Programmierung und Informatik	47
0.4 Kreativität und Problemlösung	48
0.5 Feedback	48
0.6 Literaturhinweise	49
0.7 Biografien	50
Kapitel 1 Computer, Menschen und Programmierung	53
1.1 Einführung	54
1.2 Software	55
1.3 Menschen	57
1.4 Informatik	60
1.5 Computer sind allgegenwärtig	61
1.5.1 Mit und ohne Bildschirm	61
1.5.2 Schifffahrt	62
1.5.3 Telekommunikation	63
1.5.4 Medizin	65
1.5.5 EDV	66
1.5.6 Eine vertikale Betrachtung	68
1.5.7 Na und?	69
1.6 Ideale für Programmierer	69

Teil I	Die Grundlagen	77
Kapitel 2	Hello, World!	79
2.1	Programme	80
2.2	Das klassische erste Programm	81
2.3	Kompilierung	84
2.4	Linken	86
2.5	Programmierumgebungen	87
Kapitel 3	Objekte, Typen und Werte	93
3.1	Eingabe	94
3.2	Variablen	96
3.3	Eingabe und Typ	97
3.4	Operationen und Operatoren	99
3.5	Zuweisung und Initialisierung	102
3.5.1	Ein Beispiel: Wortwiederholungen löschen	104
3.6	Zusammengesetzte Zuweisungsoperatoren	105
3.6.1	Ein Beispiel: Wortwiederholungen nummerieren	106
3.7	Namen	107
3.8	Typen und Objekte	109
3.9	Typsicherheit	110
3.9.1	Sichere Typumwandlungen	111
3.9.2	Unsichere Typumwandlungen	112
Kapitel 4	Berechnungen und Anweisungen	119
4.1	Berechnungen	120
4.2	Ziele und Werkzeuge	122
4.3	Ausdrücke	124
4.3.1	Konstante Ausdrücke	125
4.3.2	Operatoren	126
4.3.3	Typumwandlungen	128
4.4	Anweisungen	129
4.4.1	Auswahanweisungen	130
4.4.2	Schleifen	136
4.5	Funktionen	140
4.5.1	Wozu brauchen wir Funktionen?	142
4.5.2	Funktionsdeklarationen	143
4.6	Vektor	144
4.6.1	Einen Vektor vergrößern	146
4.6.2	Ein Zahlenbeispiel	147
4.6.3	Ein Textbeispiel	149
4.7	Sprachkonstrukte	150

Kapitel 5	Fehler	157
5.1	Einführung	158
5.2	Fehlerquellen	160
5.3	Kompilierfehler	160
5.3.1	Syntaxfehler	161
5.3.2	Typfehler	162
5.3.3	Fehler, die keine sind (non-errors)	163
5.4	Linkerfehler	163
5.5	Laufzeitfehler	164
5.5.1	Der Aufrufer behandelt die Fehler	166
5.5.2	Die aufgerufene Funktion behandelt die Fehler	167
5.5.3	Fehler melden	168
5.6	Ausnahmen	170
5.6.1	Ungültige Argumente	170
5.6.2	Bereichsfehler	171
5.6.3	Unzulässige Eingaben	173
5.6.4	Fehler durch Einengung	176
5.7	Logische Fehler	176
5.8	Abschätzen	179
5.9	Debuggen	180
5.9.1	Praktische Debug-Hinweise	182
5.10	Vor- und Nachbedingungen	185
5.10.1	Nachbedingungen	187
5.11	Testen	188
Kapitel 6	Ein Programm schreiben	195
6.1	Das Problem	196
6.2	Über das Problem nachdenken	197
6.2.1	Entwicklungsphasen	198
6.2.2	Strategie	198
6.3	Zurück zum Taschenrechner!	200
6.3.1	Erster Versuch	201
6.3.2	Token	203
6.3.3	Token implementieren	204
6.3.4	Token verwenden	206
6.3.5	Zurück ans Reißbrett	208
6.4	Grammatiken	209
6.4.1	Ein Exkurs: deutsche Grammatik	214
6.4.2	Eine Grammatik schreiben	215
6.5	Eine Grammatik in Code umwandeln	216
6.5.1	Grammatikregeln implementieren	216
6.5.2	Ausdrücke	217
6.5.3	Terme	221
6.5.4	Faktoren	222
6.6	Die erste Version ausprobieren	223
6.7	Die zweite Version ausprobieren	227

6.8	Token-Streams	228
6.8.1	<code>Token_stream</code> implementieren	230
6.8.2	Token lesen	232
6.8.3	Zahlen lesen	233
6.9	Programmstruktur	234
Kapitel 7 Ein Programm fertigstellen		241
7.1	Einführung	242
7.2	Eingabe und Ausgabe	242
7.3	Fehlerbehandlung	244
7.4	Negative Zahlen	248
7.5	Rest: %	250
7.6	Aufräumarbeiten	251
7.6.1	Symbolische Konstanten	252
7.6.2	Einsatz von Funktionen	254
7.6.3	Code-Layout	255
7.6.4	Kommentare	256
7.7	Wiederaufnahme der Programmausführung nach Auftreten eines Fehlers	258
7.8	Variablen	261
7.8.1	Variablen und Definitionen	261
7.8.2	Namen einführen	265
7.8.3	Vordefinierte Namen	268
7.8.4	Sind wir fertig?	269
Kapitel 8 Technische Details: Funktionen und mehr		273
8.1	Technische Details	274
8.2	Deklarationen und Definitionen	275
8.2.1	Arten von Deklarationen	279
8.2.2	Variablen- und Konstantendeklarationen	279
8.2.3	Standardinitialisierung	280
8.3	Headerdateien	281
8.4	Gültigkeitsbereich	283
8.5	Funktionsaufrufe und -rückgabewerte	288
8.5.1	Argumente und Rückgabebetyp deklarieren	288
8.5.2	Rückgabewerte	290
8.5.3	Pass-by-value	291
8.5.4	Pass-by-const-reference	292
8.5.5	Pass-by-reference	294
8.5.6	Pass-by-value kontra pass-by-reference	296
8.5.7	Argumentüberprüfung und -umwandlung	299
8.5.8	Implementierung von Funktionsaufrufen	300
8.6	Auswertungsreihenfolge	304
8.6.1	Auswertung von Ausdrücken	305
8.6.2	Globale Initialisierung	305
8.7	Namensbereiche	307
8.7.1	<code>using</code> -Deklarationen und <code>using</code> -Direktiven	308

Kapitel 9 Technische Details: Klassen und mehr 315

9.1	Benutzerdefinierte Typen	316
9.2	Klassen und Klassenmember	317
9.3	Schnittstelle und Implementierung	318
9.4	Eine Klasse entwickeln	319
9.4.1	Strukturen und Funktionen	320
9.4.2	Memberfunktionen und Konstruktoren	321
9.4.3	Halten Sie Details privat	323
9.4.4	Memberfunktionen definieren	324
9.4.5	Objektbezug	327
9.4.6	Fehlerbehandlung	328
9.5	Aufzählungen	329
9.6	Operatorenüberladung	331
9.7	Klassenschnittstellen	332
9.7.1	Argumenttypen	333
9.7.2	Kopieren	335
9.7.3	Standardkonstruktoren	336
9.7.4	Konstante Memberfunktionen	339
9.7.5	Member und „Hilfsfunktionen“	340
9.8	Die Klasse <code>Date</code>	342

Teil II Ein- und Ausgabe 351

Kapitel 10 Ein- und Ausgabestreams 353

10.1	Ein- und Ausgabe	354
10.2	Das E/A-Stream-Modell	355
10.3	Dateien	357
10.4	Dateien öffnen	358
10.5	Dateien lesen und schreiben	360
10.6	E/A-Fehlerbehandlung	362
10.7	Einzelne Werte lesen	366
10.7.1	Das Problem in handliche Teilprobleme zerlegen	368
10.7.2	Trennung von Kommunikation und Funktion	371
10.8	Benutzerdefinierte Ausgabeoperatoren	372
10.9	Benutzerdefinierte Eingabeoperatoren	373
10.10	Standardlösung für eine Einleseschleife	374
10.11	Eine strukturierte Datei lesen	375
10.11.1	Repräsentation im Speicher	376
10.11.2	Strukturierte Werte einlesen	378
10.11.3	Austauschbare Darstellungen	382

Kapitel 11	Die Ein- und Ausgabe anpassen	387
11.1	Regelmäßigkeit und Individualität	388
11.2	Formatierung der Ausgabe	388
11.2.1	Ausgabe ganzer Zahlen	389
11.2.2	Eingabe ganzer Zahlen	391
11.2.3	Ausgabe von Gleitkommazahlen	392
11.2.4	Genauigkeit	393
11.2.5	Felder	395
11.3	Dateien öffnen	396
11.3.1	Öffnungsmodi für Dateien	396
11.3.2	Binärdateien	397
11.3.3	Festlegen der Schreib- und Leseposition in Dateien	400
11.4	Stringstreams	401
11.5	Zeilenorientierte Eingabe	402
11.6	Zeichenklassifizierung	403
11.7	Verwendung eigener Trennzeichen	406
11.8	Und es gibt noch so viel mehr	413
Kapitel 12	Ein Anzeigemodell	419
12.1	Wozu Grafik?	420
12.2	Ein Anzeigemodell	421
12.3	Ein erstes Beispiel	422
12.4	Programmieren mit GUI-Bibliotheken	426
12.5	Koordinaten	427
12.6	Formen	428
12.7	Programmieren mit Grafikprimitiven	428
12.7.1	Die Grafik-Header und <code>main()</code>	429
12.7.2	Ein nahezu leeres Fenster	429
12.7.3	Achsen	431
12.7.4	Grafische Darstellung von Funktionen	433
12.7.5	Polygone	434
12.7.6	Rechtecke	436
12.7.7	Füllen	438
12.7.8	Text	438
12.7.9	Bilder	440
12.7.10	Und vieles mehr	441
12.8	Ausführung des Grafikbeispiels	442
12.8.1	Die Quelldateien	443
Kapitel 13	Grafikklassen	447
13.1	Überblick über die Grafikklassen	448
13.2	<code>Point</code> und <code>Line</code>	451
13.3	<code>Lines</code>	453
13.4	<code>Color</code>	456
13.5	<code>Line_style</code>	458
13.6	<code>Open_polyline</code>	461

13.7	<code>Closed_polyline</code>	462
13.8	<code>Polygon</code>	463
13.9	<code>Rectangle</code>	465
13.10	Arbeiten mit unbenannten Objekten	470
13.11	<code>Text</code>	472
13.12	<code>Circle</code>	474
13.13	<code>Ellipse</code>	476
13.14	<code>Marked_polyline</code>	478
13.15	<code>Marks</code>	479
13.16	<code>Mark</code>	480
13.17	<code>Image</code>	482
Kapitel 14 Grafikklassen-Design		489
14.1	Designprinzipien	490
14.1.1	Typen	490
14.1.2	Operationen	492
14.1.3	Namensgebung	493
14.1.4	Zugriff und Veränderung	495
14.2	<code>Shape</code>	496
14.2.1	Eine abstrakte Klasse	497
14.2.2	Zugriffskontrolle	498
14.2.3	<code>Shape</code> -Objekte zeichnen	501
14.2.4	Kopieren und Zugriffskontrolle	504
14.3	Basisklassen und abgeleitete Klassen	506
14.3.1	Objekt-Layout	507
14.3.2	Klassen ableiten und virtuelle Funktionen definieren	509
14.3.3	Überschreibung	510
14.3.4	Zugriff	511
14.3.5	Rein virtuelle Funktionen	512
14.4	Vorteile der objektorientierten Programmierung	513
Kapitel 15 Grafische Darstellung von Funktionen und Daten		519
15.1	Einführung	520
15.2	Grafische Darstellung einfacher Funktionen	520
15.3	<code>Function</code>	524
15.3.1	Vorgabeargumente	525
15.3.2	Weitere Beispiele	526
15.4	Achsen	528
15.5	Approximation	530
15.6	Darstellung von Daten	536
15.6.1	Einlesen aus Dateien	537
15.6.2	Allgemeines Layout	539
15.6.3	Skalierung	540
15.6.4	Aufbau des Graphen	541

Kapitel 16 Grafische Benutzerschnittstellen	549
16.1 Verschiedene Benutzerschnittstellen	550
16.2 Die Schaltfläche Weiter	551
16.3 Ein einfaches Fenster	552
16.3.1 Eine Callback-Funktion	554
16.3.2 Eine Warteschleife	557
16.4 Schaltflächen und andere Widgets	558
16.4.1 Widgets	558
16.4.2 Schaltflächen	560
16.4.3 <code>In_box</code> und <code>Out_box</code>	560
16.4.4 Menüs	561
16.5 Ein Beispiel	562
16.6 Umkehrung der Steuerung	565
16.7 Ein Menü hinzufügen	567
16.8 GUI-Code debuggen	571
Teil III Daten und Algorithmen	577
Kapitel 17 Vektoren und Freispeicher	579
17.1 Einführung	580
17.2 Vektor-Grundlagen	582
17.3 Speicher, Adressen und Zeiger	584
17.3.1 Der <code>sizeof</code> -Operator	586
17.4 Freispeicher und Zeiger	587
17.4.1 Freispeicher reservieren (Allokation)	588
17.4.2 Zugriff über Zeiger	589
17.4.3 Bereiche	590
17.4.4 Initialisierung	592
17.4.5 Der Nullzeiger	593
17.4.6 Freispeicher freigeben (Deallokation)	593
17.5 Destruktoren	595
17.5.1 Automatisch generierte Destruktoren	597
17.5.2 Destruktoren und Freispeicher	598
17.6 Zugriff auf Elemente	600
17.7 Zeiger auf Klassenobjekte	601
17.8 Eingriff ins Typensystem: <code>void*</code> und Casts	602
17.9 Zeiger und Referenzen	604
17.9.1 Zeiger- und Referenzparameter	605
17.9.2 Zeiger, Referenzen und Vererbung	607
17.9.3 Ein Beispiel: Listen	607
17.9.4 Operationen für Listen	609
17.9.5 Verwendung von Listen	610
17.10 Der <code>this</code> -Zeiger	611
17.10.1 Weitere Anwendungsbeispiele	613

Kapitel 18	Vektoren und Arrays	619
18.1	Einführung	620
18.2	Kopieren	621
18.2.1	Kopierkonstruktoren	622
18.2.2	Zuweisungsoperatoren	624
18.2.3	Terminologie	626
18.3	Essenzielle Operationen	627
18.3.1	Explizite Konstruktoren	629
18.3.2	Konstruktoren und Destruktoren debuggen	630
18.4	Zugriff auf Vektor-Elemente	632
18.4.1	Überladung für <code>const</code> -Objekte	633
18.5	Arrays	634
18.5.1	Zeiger auf Array-Elemente	636
18.5.2	Zeiger und Arrays	638
18.5.3	Array-Initialisierung	640
18.5.4	Probleme mit Zeigern	641
18.6	Beispiele: Palindrom	644
18.6.1	Palindrome mit <code>string</code>	644
18.6.2	Palindrome mit Arrays	645
18.6.3	Palindrome mit Zeigern	646
Kapitel 19	Vektor, Templates und Ausnahmen	653
19.1	Problematik	654
19.2	Die Größe ändern	657
19.2.1	Darstellung	658
19.2.2	<code>reserve</code> und <code>capacity</code>	659
19.2.3	<code>resize</code>	660
19.2.4	<code>push_back</code>	660
19.2.5	Zuweisung	661
19.2.6	Aktueller Stand unseres Vektors	663
19.3	Templates	664
19.3.1	Typen als Template-Parameter	664
19.3.2	Generische Programmierung	666
19.3.3	Container und Vererbung	669
19.3.4	Integer als Template-Parameter	670
19.3.5	Deduktion von Template-Argumenten	671
19.3.6	Verallgemeinerung von <code>vector</code>	672
19.4	Bereichsüberprüfung und Ausnahmen	675
19.4.1	Eine Nebenbemerkung: Überlegungen zum Design	676
19.4.2	Eine Beichte: Makros	678
19.5	Ressourcen und Ausnahmen	679
19.5.1	Potenzielle Probleme mit der Ressourcenverwaltung	680
19.5.2	Ressourcenbelegung ist Initialisierung (RAII)	682
19.5.3	Garantien	683
19.5.4	<code>auto_ptr</code>	684
19.5.5	RAII für <code>vector</code>	685

Kapitel 20	Container und Iteratoren	693
20.1	Daten speichern und verarbeiten	694
20.1.1	Mit Daten arbeiten	695
20.1.2	Code allgemein halten	696
20.2	STL-Ideale	699
20.3	Sequenzen und Iteratoren	703
20.3.1	Zurück zum Beispiel	705
20.4	Verkettete Listen	706
20.4.1	Listenoperationen	708
20.4.2	Iteration	709
20.5	Weitere Verallgemeinerung des <code>vector</code> -Typs	711
20.6	Ein Beispiel: ein einfacher Texteditor	713
20.6.1	Die Zeilen	715
20.6.2	Iteration	716
20.7	<code>vector</code> , <code>list</code> und <code>string</code>	720
20.7.1	Einfügen und Löschen	721
20.8	Unseren Vektor an die STL anpassen	723
20.9	Annäherung der integrierten Arrays an die STL	725
20.10	Überblick über die Container	727
20.10.1	Iterator-Kategorien	730
Kapitel 21	Algorithmen und Maps	737
21.1	Algorithmen der Standardbibliothek	738
21.2	Der einfachste Algorithmus: <code>find()</code>	739
21.2.1	Einige generische Anwendungsbereiche	741
21.3	Die allgemeine Suche: <code>find_if()</code>	742
21.4	Funktionsobjekte	744
21.4.1	Allgemeine Darstellung des Konzepts der Funktionsobjekte	745
21.4.2	Prädikate für Klassenmember	746
21.5	Numerische Algorithmen	747
21.5.1	Akkumulator	748
21.5.2	<code>accumulate()</code> – allgemeine Version	749
21.5.3	Das innere Produkt	751
21.5.4	<code>inner_product()</code> – allgemeine Version	752
21.6	Assoziative Container	752
21.6.1	Maps	753
21.6.2	Maps – ein Überblick	755
21.6.3	Ein weiteres <code>map</code> -Beispiel	758
21.6.4	<code>unordered_map</code>	761
21.6.5	Sets	763
21.7	Kopieren	765
21.7.1	Kopieren	765
21.7.2	Stream-Iteratoren	766
21.7.3	Mit <code>set</code> Ordnung halten	768
21.7.4	<code>copy_if</code>	769
21.8	Sortieren und suchen	770

Teil IV	Erweiterung des Blickwinkels	777
Kapitel 22	Ideale und Geschichte	779
22.1	Geschichte, Ideale und Professionalität	780
22.1.1	Programmiersprachen – Ziele und Philosophien	780
22.1.2	Programmierideale	782
22.1.3	Stile/Paradigmen	789
22.2	(Kurze) Geschichte der Programmiersprachen	792
22.2.1	Die frühesten Sprachen	793
22.2.2	Die Wurzeln der modernen Sprachen	795
22.2.3	Die Algol-Familie	800
22.2.4	Simula	807
22.2.5	C	809
22.2.6	C++	812
22.2.7	Heute	815
22.2.8	Informationsquellen	816
Kapitel 23	Textmanipulation	821
23.1	Text	822
23.2	Strings	822
23.3	E/A-Streams	826
23.4	Maps	827
23.4.1	Implementierungsdetails	833
23.5	Ein Problem	835
23.6	Die Idee der regulären Ausdrücke	837
23.7	Suchen mithilfe regulärer Ausdrücke	839
23.8	Syntax der regulären Ausdrücke	842
23.8.1	Zeichen und Sonderzeichen	843
23.8.2	Zeichenklassen	844
23.8.3	Quantifizierer	844
23.8.4	Gruppierung	846
23.8.5	Alternativen	846
23.8.6	Zeichensätze und -bereiche	847
23.8.7	Fehler bei regulären Ausdrücken	849
23.9	Abgleich mit regulären Ausdrücken	851
23.10	Literaturhinweise	856
Kapitel 24	Numerik	861
24.1	Einführung	862
24.2	Größe, Genauigkeit und Überlauf	862
24.2.1	Numerische Grenzwerte	866
24.3	Arrays	867
24.4	Mehrdimensionale Arrays im C-Stil	868
24.5	Die <code>Matrix</code> -Bibliothek	869
24.5.1	Dimensionen und Zugriff	870
24.5.2	1D- <code>Matrix</code>	872

24.5.3	2D-Matrix	876
24.5.4	Matrix-E/A	878
24.5.5	3D-Matrix	878
24.6	Ein Beispiel: lineare Gleichungen	879
24.6.1	Klassische Gauß'sche Elimination	881
24.6.2	Pivotisierung	882
24.6.3	Testen	883
24.7	Zufallszahlen	884
24.8	Die mathematischen Standardfunktionen	886
24.9	Komplexe Zahlen	887
24.10	Literaturhinweise	889

Kapitel 25 Programmierung eingebetteter Systeme 893

25.1	Eingebettete Systeme	894
25.2	Grundlegende Konzepte	897
25.2.1	Vorhersagbarkeit	899
25.2.2	Ideale	900
25.2.3	Mit dem Scheitern leben	901
25.3	Speicherverwaltung	903
25.3.1	Probleme mit dem Freispeicher	904
25.3.2	Alternativen zum üblichen Freispeicher	907
25.3.3	Pool-Beispiel	908
25.3.4	Stack-Beispiel	909
25.4	Adressen, Zeiger und Arrays	910
25.4.1	Ungeprüfte Umwandlungen	911
25.4.2	Ein Problem: dysfunktionale Schnittstellen	911
25.4.3	Eine Lösung: eine Schnittstellenklasse	915
25.4.4	Vererbung und Container	918
25.5	Bits, Bytes und Words	921
25.5.1	Bits und Bitoperationen	921
25.5.2	bitset	926
25.5.3	signed und unsigned	927
25.5.4	Bitmanipulation	931
25.5.5	Bitfelder	933
25.5.6	Ein Beispiel: einfache Verschlüsselung	935
25.6	Codierstandards	939
25.6.1	Wie sollte ein Codierstandard aussehen?	940
25.6.2	Beispielregeln	942
25.6.3	Konkrete Codierstandards	947

Kapitel 26 Testen 953

26.1	Worum geht es uns?	954
26.1.1	Warnung	955
26.2	Beweise	956
26.3	Testen	956
26.3.1	Regressionstests	957
26.3.2	Unit-Tests	958

26.3.3	Algorithmen und Nicht-Algorithmen	965
26.3.4	Systemtests	972
26.3.5	Klassen testen	976
26.3.6	Annahmen aufspüren, die nicht standhalten	979
26.4	Testfreundliches Design	981
26.5	Debuggen	982
26.6	Performance	982
26.6.1	Zeitmessungen	984
26.7	Literaturhinweise	985
Kapitel 27 Die Programmiersprache C		989
27.1	C und C++: Geschwister	990
27.1.1	C/C++-Kompatibilität	992
27.1.2	C++-Features, die in C fehlen	993
27.1.3	Die C-Standardbibliothek	995
27.2	Funktionen	996
27.2.1	Keine Überladung von Funktionsnamen	996
27.2.2	Typprüfung von Funktionsargumenten	996
27.2.3	Funktionsdefinitionen	998
27.2.4	C von C++ aus und C++ von C aus aufrufen	1000
27.2.5	Zeiger auf Funktionen	1002
27.3	Kleinere Sprachunterschiede	1003
27.3.1	Namensbereich des <code>struct</code> -Tags	1003
27.3.2	Schlüsselwörter	1004
27.3.3	Definitionen	1005
27.3.4	Typumwandlungen im C-Stil	1006
27.3.5	Umwandlung von <code>void*</code>	1007
27.3.6	<code>enum</code>	1008
27.3.7	Namensbereiche	1009
27.4	Freispeicher	1009
27.5	C-Strings	1011
27.5.1	C-Strings und <code>const</code>	1013
27.5.2	Byte-Operationen	1014
27.5.3	Ein Beispiel: <code>strcpy()</code>	1014
27.5.4	Eine Stilfrage	1015
27.6	Eingabe/Ausgabe: <code>stdio</code>	1015
27.6.1	Ausgabe	1016
27.6.2	Eingabe	1017
27.6.3	Dateien	1018
27.7	Konstanten und Makros	1019
27.8	Makros	1020
27.8.1	Funktionsähnliche Makros	1021
27.8.2	Syntax-Makros	1022
27.8.3	Bedingte Kompilierung	1023
27.9	Ein Beispiel: aufdringliche Container	1024

Teil V	Anhang	1035
Anhang A	Sprachübersicht	1037
A.1	Allgemein	1038
A.1.1	Terminologie	1038
A.1.2	Programmstart und -beendigung	1039
A.1.3	Kommentare	1039
A.2	Literale	1040
A.2.1	Integer-Literale	1040
A.2.2	Gleitkommaliterale	1042
A.2.3	Boolesche Literale	1043
A.2.4	Zeichenliterale	1043
A.2.5	String-Literale	1044
A.2.6	Das Zeigerliteral	1044
A.3	Bezeichner	1044
A.3.1	Schlüsselwörter	1045
A.4	Gültigkeitsbereich, Speicherklasse und Lebensdauer	1045
A.4.1	Gültigkeitsbereich	1046
A.4.2	Speicherklassen	1046
A.4.3	Lebensdauer	1048
A.5	Ausdrücke	1048
A.5.1	Benutzerdefinierte Operatoren	1055
A.5.2	Automatische Typumwandlung	1056
A.5.3	Konstante Ausdrücke	1057
A.5.4	<code>sizeof</code>	1058
A.5.5	Logische Ausdrücke	1058
A.5.6	<code>new</code> und <code>delete</code>	1059
A.5.7	Casts	1059
A.6	Anweisungen	1060
A.7	Deklarationen	1062
A.7.1	Definitionen	1062
A.8	Integrierte Typen	1063
A.8.1	Zeiger	1064
A.8.2	Arrays	1066
A.8.3	Referenzen	1066
A.9	Funktionen	1067
A.9.1	Auflösung von Überladungen	1067
A.9.2	Vorgabeargumente	1069
A.9.3	Unspezifizierte Argumente	1069
A.9.4	Bindespezifikationen	1069
A.10	Benutzerdefinierte Typen	1070
A.10.1	Überladen von Operatoren	1070
A.11	Aufzählungen	1071
A.12	Klassen	1071
A.12.1	Memberzugriff	1071
A.12.2	Klassenmemberdefinitionen	1074
A.12.3	Konstruktion, Destruktion und Kopieren	1075
A.12.4	Abgeleitete Klassen	1078

A.12.5	Bitfelder	1081
A.12.6	Unions	1082
A.13	Templates	1083
A.13.1	Template-Argumente	1083
A.13.2	Template-Instanzierung	1084
A.13.3	Template-Membertypen	1085
A.14	Ausnahmen	1086
A.15	Namensbereiche	1087
A.16	Aliase	1088
A.17	Präprozessor-Direktiven	1089
A.17.1	<code>#include</code>	1089
A.17.2	<code>#define</code>	1089
Anhang B Zusammenfassung der Standardbibliothek		1091
B.1	Überblick	1092
B.1.1	Headerdateien	1093
B.1.2	Namensbereich <code>std</code>	1096
B.1.3	Notation	1096
B.2	Fehlerbehandlung	1097
B.2.1	Ausnahmen	1097
B.3	Iteratoren	1099
B.3.1	Iterator-Modell	1099
B.3.2	Iterator-Kategorien	1101
B.4	Container	1102
B.4.1	Überblick	1105
B.4.2	Membertypen	1106
B.4.3	Konstruktoren, Destruktoren und Zuweisungen	1106
B.4.4	Iteratoren	1107
B.4.5	Elementzugriff	1108
B.4.6	Stack- und Warteschlangenoperationen	1108
B.4.7	Listenoperationen	1109
B.4.8	Größe und Kapazität	1109
B.4.9	Weitere Operationen	1110
B.4.10	Operationen für assoziative Container	1110
B.5	Algorithmen	1111
B.5.1	Nichtmodifizierende Sequenzalgorithmen	1112
B.5.2	Modifizierende Sequenzalgorithmen	1113
B.5.3	Utility-Algorithmen	1115
B.5.4	Sortieren und Suchen	1116
B.5.5	Mengen-Algorithmen	1117
B.5.6	Heap-Algorithmen	1118
B.5.7	Permutationen	1119
B.5.8	<code>min</code> und <code>max</code>	1120
B.6	Hilfskomponenten	1120
B.6.1	Insert-Iteratoren	1120
B.6.2	Funktionsobjekte	1121
B.6.3	<code>pair</code>	1123

B.7	E/A-Streams	1123
	B.7.1 Hierarchie der E/A-Streams	1124
	B.7.2 Fehlerbehandlung	1126
	B.7.3 Eingabeoperationen	1126
	B.7.4 Ausgabeoperationen	1127
	B.7.5 Formatierung	1128
	B.7.6 Standardmanipulatoren	1128
B.8	Stringmanipulation	1130
	B.8.1 Klassifizierung von Zeichen	1130
	B.8.2 Strings	1131
	B.8.3 Reguläre Ausdrücke	1132
B.9	Numerik	1135
	B.9.1 Numerische Grenzwerte	1135
	B.9.2 Mathematische Standardfunktionen	1137
	B.9.3 Komplexe Zahlen	1137
	B.9.4 Valarrays	1139
	B.9.5 Generische Numerik-Algorithmen	1139
B.10	C-Funktionen der Standardbibliothek	1139
	B.10.1 Dateien	1140
	B.10.2 Die <code>printf()</code> -Familie	1141
	B.10.3 C-Strings	1145
	B.10.4 Speicher	1147
	B.10.5 Datum und Uhrzeit	1147
	B.10.6 Weitere Funktionen	1149
B.11	Andere Bibliotheken	1150
Anhang C Erste Schritte mit Visual Studio		1151
C.1	Ein Programm zur Ausführung bringen	1152
C.2	Visual Studio installieren	1152
C.3	Ein Programm erzeugen und ausführen	1152
	C.3.1 Ein neues Projekt anlegen	1153
	C.3.2 Die Headerdatei <code>std_lib_facilities.h</code> verwenden	1153
	C.3.3 Dem Projekt eine C++-Quelldatei hinzufügen	1154
	C.3.4 Quellcode eingeben	1154
	C.3.5 Ein ausführbares Programm erstellen	1154
	C.3.6 Das Programm ausführen	1154
	C.3.7 Das Programm speichern	1155
C.4	Später	1155
Anhang D FLTK-Installation		1157
D.1	Einführung	1158
D.2	Das FLTK herunterladen	1158
D.3	Das FLTK installieren	1159
D.4	Das FLTK in Visual Studio verwenden	1160
D.5	Testen, ob alles funktioniert	1160

Anhang E	GUI-Implementierung	1163
E.1	Callback-Implementierung	1164
E.2	Widget-Implementierung	1165
E.3	Window-Implementierung	1166
E.4	Vector_ref	1167
E.5	Ein Beispiel: Widgets manipulieren	1168
Glossar		1171
Literaturverzeichnis		1181
Bildnachweis		1185
Register		1187
Farbteil		I