

Inhalt

Vorwort	17
Vorwort des Fachgutachters	23

1 Grundlagen in C++	25
1.1 Die Entstehung von C++	25
1.1.1 Aufbau von C++	28
1.2 Erste Schritte der C++-Programmierung	31
1.2.1 Ein Programm erzeugen mit einem Kommandozeilen-Compiler	32
1.2.2 Ausführen des Programms	34
1.2.3 Ein Programm erzeugen mit einer IDE	34
1.3 Symbole von C++	35
1.3.1 Bezeichner	35
1.3.2 Schlüsselwörter	35
1.3.3 Literale	36
1.3.4 Einfache Begrenzer	37
1.4 Basisdatentypen	39
1.4.1 Deklaration und Definition	39
1.4.2 Was ist eine Variable?	40
1.4.3 Der Datentyp »bool«	40
1.4.4 Der Datentyp »char«	41
1.4.5 Die Datentypen »int«	44
1.4.6 Gleitkommazahlen »float«, »double« und »long double«	46
1.4.7 Limits für Ganzzahl- und Gleitpunkt-Datentypen	50
1.4.8 Die Größen der Basistypen	51
1.4.9 void	52
1.5 Konstanten	53
1.6 Standard Ein-/Ausgabe-Streams	54
1.6.1 Die neuen Streams – »cout«, »cin«, »cerr«, »clog«	54
1.6.2 Ausgabe mit »cout«	56
1.6.3 Ausgabe mit »cerr«	56
1.6.4 Eingabe mit »cin«	57
1.7 Operatoren	59
1.7.1 Arithmetische Operatoren	60
1.7.2 Inkrement- und Dekrementoperator	63
1.7.3 Bit-Operatoren	64
1.7.4 Weitere Operatoren	68

1.8	Kommentare	68
1.9	Kontrollstrukturen	69
1.9.1	Verzweigungen (Selektionen)	69
1.9.2	Schleifen (Iterationen)	88
1.9.3	Sprunganweisungen	96
1.10	Funktionen	99
1.10.1	Deklaration und Definition	99
1.10.2	Funktionsaufruf und Parameterübergabe	102
1.10.3	Lokale und globale Variablen	109
1.10.4	Standardparameter	110
1.10.5	Funktionen überladen	113
1.10.6	Inline-Funktionen	117
1.10.7	Rekursionen	120
1.10.8	Die »main«-Funktion	121
1.11	Präprozessor-Direktiven	122
1.11.1	Die »#define«-Direktive	123
1.11.2	Die »#undef«-Direktive	126
1.11.3	Die »#include«-Direktive	127
1.11.4	Die Direktiven »#error« und »#pragma«	128
1.11.5	Bedingte Kompilierung	129

2 Höhere und fortgeschrittene Datentypen 133

2.1	Zeiger	133
2.1.1	Zeiger deklarieren	134
2.1.2	Adresse im Zeiger speichern	135
2.1.3	Zeiger dereferenzieren	137
2.1.4	Zeiger, die auf andere Zeiger verweisen	141
2.1.5	Dynamisch Speicherobjekte anlegen und zerstören – »new« und »delete«	143
2.1.6	void-Zeiger	148
2.1.7	Konstante Zeiger	148
2.2	Referenzen	149
2.3	Arrays	152
2.3.1	Arrays deklarieren	152
2.3.2	Arrays initialisieren	153
2.3.3	Bereichsüberschreitung von Arrays	155
2.3.4	Anzahl der Elemente eines Arrays ermitteln	156
2.3.5	Array-Wert von Tastatur einlesen	157
2.3.6	Mehrdimensionale Arrays	157
2.4	Zeichenketten (C-Strings) – char-Array	159
2.4.1	C-String deklarieren und initialisieren	160

2.4.2	C-String einlesen	161
2.4.3	C-Strings: Bibliotheksfunktionen	161
2.5	Arrays und Zeiger	166
2.5.1	C-Strings und Zeiger	171
2.5.2	Mehrfache Indirektion	171
2.5.3	C-String-Tabellen	173
2.5.4	Arrays im Heap (dynamisches Array)	176
2.6	Parameterübergabe mit Zeigern, Arrays und Referenzen	181
2.6.1	Call by value	181
2.6.2	Call by reference – Zeiger als Funktionsparameter	182
2.6.3	Call by reference mit Referenzen nachbilden	184
2.6.4	Arrays als Funktionsparameter	185
2.6.5	Mehrdimensionale Arrays an Funktionen übergeben ...	187
2.6.6	Argumente an die main-Funktion übergeben	188
2.7	Rückgabewerte von Zeigern, Arrays und Referenzen	190
2.7.1	Zeiger als Rückgabewert	190
2.7.2	Referenz als Rückgabewert	194
2.7.3	const-Zeiger als Rückgabewert	195
2.7.4	Array als Rückgabewert	195
2.7.5	Mehrere Rückgabewerte	196
2.8	Fortgeschrittene Typen	197
2.8.1	Strukturen	198
2.8.2	Unions	218
2.8.3	Aufzählungstypen	221
2.8.4	typedef	223

3 Gültigkeitsbereiche, spezielle Deklarationen und Typumwandlungen 225

3.1	Gültigkeitsbereiche (Scope)	225
3.1.1	Lokaler Gültigkeitsbereich (Local Scope)	226
3.1.2	Gültigkeitsbereich Funktionen	226
3.1.3	Gültigkeitsbereich Namensraum (Namespaces)	228
3.1.4	Gültigkeitsbereich Klassen (Class Scope)	228
3.2	Namensräume (Namespaces)	228
3.2.1	Neuen Namensbereich erzeugen (Definition)	228
3.2.2	Zugriff auf die Bezeichner im Namensraum	231
3.2.3	using – einzelne Bezeichner aus einem Namensraum importieren	233
3.2.4	using – alle Bezeichner aus einem Namensraum importieren	236
3.2.5	Namensauflösung	239

- 3.2.6 Alias-Namen für Namensbereiche 240
- 3.2.7 Anonyme (namenlose) Namensbereiche 241
- 3.2.8 Namensbereich und Header-Dateien 242
- 3.3 C-Funktionen bzw. -Bibliotheken in einem C++-Programm 244
 - 3.3.1 C-Funktionen aus einer C-Bibliothek aufrufen 245
- 3.4 Speicherklassenattribute 249
 - 3.4.1 Speicherklasse »auto« 250
 - 3.4.2 Speicherklasse »register« 250
 - 3.4.3 Speicherklasse »static« 250
 - 3.4.4 Speicherklasse »extern« 251
 - 3.4.5 Speicherklasse »mutable« 253
- 3.5 Typqualifikatoren 253
 - 3.5.1 Qualifizierer »const« 254
 - 3.5.2 Qualifizierer »volatile« 254
- 3.6 Funktionsattribute 255
- 3.7 Typumwandlung 255
 - 3.7.1 Standard-Typumwandlung 256
 - 3.7.2 Explizite Typumwandlung 260

4 Objektorientierte Programmierung 265

- 4.1 OOP-Konzept versus prozedurales Konzept 265
 - 4.1.1 OOP-Paradigmen 266
- 4.2 Klassen (fortgeschrittene Typen) 267
 - 4.2.1 Klassen deklarieren 268
 - 4.2.2 Elementfunktion (Klassenmethode) definieren 269
 - 4.2.3 Objekte deklarieren 271
 - 4.2.4 Kurze Zusammenfassung 271
 - 4.2.5 »private« und »public« – Zugriffsrechte in der Klasse ... 272
 - 4.2.6 Zugriff auf die Elemente (Member) einer Klasse 274
 - 4.2.7 Ein Programm organisieren 281
 - 4.2.8 Konstruktoren 285
 - 4.2.9 Destruktoren 293
- 4.3 Mehr zu den Klassenmethoden (Klassenfunktionen) 295
 - 4.3.1 Inline-Methoden (explizit und implizit) 296
 - 4.3.2 Zugriffsmethoden 299
 - 4.3.3 Read-only-Methoden 303
 - 4.3.4 this-Zeiger 305
- 4.4 Verwenden von Objekten 307
 - 4.4.1 Read-only-Objekte 307
 - 4.4.2 Objekte als Funktionsargumente 307
 - 4.4.3 Objekte als Rückgabewert 314

4.4.4	Klassen-Array (Array von Objekten)	316
4.4.5	Dynamische Objekte	318
4.4.6	Dynamische Klassenelemente	324
4.4.7	Objekte kopieren (Kopierkonstruktor)	328
4.4.8	Dynamisch erzeugte Objekte kopieren (»operator=()«)	330
4.4.9	Standardmethoden (Überblick)	331
4.4.10	Objekte als Elemente (bzw. Eigenschaften) in anderen Klassen	332
4.4.11	Teilobjekte initialisieren	338
4.4.12	Klassen in Klassen verschachteln	340
4.4.13	Konstante Klasseneigenschaften (Datenelemente)	341
4.4.14	Statische Klasseneigenschaften (Datenelemente)	343
4.4.15	Statische Klassenmethoden	348
4.4.16	Friend-Funktionen bzw. friend-Klassen	349
4.4.17	Zeiger auf Eigenschaften einer Klasse	352
4.5	Operatoren überladen	358
4.5.1	Grundlegendes zur Operator-Überladung	358
4.5.2	Überladen von arithmetischen Operatoren	362
4.5.3	Überladen von unären Operatoren	371
4.5.4	Überladen von ++ und --	374
4.5.5	Überladen des Zuweisungsoperators	376
4.5.6	Überladen des Indexoperators »[]« (Arrays überladen)	378
4.5.7	Shift-Operatoren überladen	381
4.5.8	()-Operator überladen	385
4.6	Typumwandlung für Klassen	388
4.6.1	Konvertierungskonstruktor	388
4.6.2	Konvertierungsfunktion	390
4.7	Vererbung (abgeleitete Klassen)	392
4.7.1	Anwendungsbeispiel (Vorbereitung)	394
4.7.2	Die Ableitung einer Klasse	397
4.7.3	Redefinition von Klassenelementen	401
4.7.4	Konstruktoren	404
4.7.5	Destruktoren	407
4.7.6	Zugriffsrecht »protected«	407
4.7.7	Typumwandlung abgeleiteter Klassen	410
4.7.8	Klassenbibliotheken erweitern	413
4.8	Polymorphismus	414
4.8.1	Statische bzw. dynamische Bindung	415
4.8.2	Virtuelle Methoden	415

4.8.3	Virtuelle Methoden redefinieren	420
4.8.4	Arbeitsweise virtueller Methoden	426
4.8.5	Virtuelle Destruktoren bzw. Destruktoren abgeleiteter Klassen	431
4.8.6	Polymorphismus und der Zuweisungsoperator	433
4.8.7	Rein virtuelle Methoden und abstrakte Basisklassen	435
4.8.8	Probleme mit der Vererbung und der dynamic_cast-Operator	439
4.8.9	Fallbeispiel: Verkettete Listen	441
4.9	Mehrfachvererbung	463
4.9.1	Indirekte Basisklassen erben	467
4.9.2	Virtuelle indirekte Basisklassen erben	471

5 Templates und STL 477

5.1	Funktions-Templates	477
5.1.1	Funktions-Templates definieren	479
5.1.2	Typübereinstimmung	482
5.1.3	Funktions-Templates über mehrere Module	483
5.1.4	Spezialisierung von Funktions-Templates	483
5.1.5	Verschiedene Parameter	487
5.1.6	Explizite Template-Argumente	488
5.2	Klassen-Templates	489
5.2.1	Definition	490
5.2.2	Methoden von Klassen-Templates definieren	491
5.2.3	Klassen-Template generieren (Instantiierung)	496
5.2.4	Weitere Template-Parameter	501
5.2.5	Standardargumente von Templates	504
5.2.6	Explizite Instantiierung	506
5.3	STL (Standard Template Library)	507
5.3.1	Konzept von STL	508
5.3.2	Hilfsmittel (Hilfsstrukturen)	512
5.3.3	Allokator	524
5.3.4	Iteratoren	525
5.3.5	Container	530
5.3.6	Algorithmen	581
5.3.7	Allokatoren	643

6 Exception-Handling 661

6.1	Exception-Handling in C++	662
6.2	Eine Exception auslösen	662

6.3	Eine Exception auffangen – Handle einrichten	663
6.3.1	Reihenfolge (Auflösung) der Ausnahmen	666
6.3.2	Alternatives »catch(...)«	666
6.3.3	Stack-Abwicklung (Stack-Unwinding)	668
6.3.4	try-Blöcke verschachteln	670
6.3.5	Exception weitergeben	673
6.4	Ausnahmeklassen (Fehlerklassen)	676
6.4.1	Klassenspezifische Exceptions	678
6.5	Standard-Exceptions	680
6.5.1	Virtuelle Methode »what()«	681
6.5.2	Anwenden der Standard-Exceptions	681
6.6	System-Exceptions	686
6.6.1	bad_alloc	687
6.6.2	bad_cast	687
6.6.3	bad_typeid	687
6.6.4	bad_exception	687
6.7	Exception-Spezifikation	688
6.7.1	Unerlaubte Exceptions	689
6.7.2	terminate-Handle einrichten	691

7 C++-Standardbibliothek 695

7.1	Die String-Bibliothek (string-Klasse)	695
7.1.1	Exception-Handling	697
7.1.2	Datentypen	697
7.1.3	Strings erzeugen (Konstruktoren)	698
7.1.4	Zuweisungen	700
7.1.5	Elementzugriff	702
7.1.6	Länge und Kapazität ermitteln bzw. ändern	703
7.1.7	Konvertieren in einen C-String	706
7.1.8	Manipulation von Strings	707
7.1.9	Suchen in Strings	710
7.1.10	Strings vergleichen	717
7.1.11	Die (überladenen) Operatoren	719
7.1.12	Einlesen einer ganzen Zeile	721
7.2	Ein-/Ausgabe Klassenhierarchie (I/O-Streams)	722
7.2.1	Klassen für Ein- und Ausgabe-Streams	724
7.2.2	Klassen für Datei-Streams (File-Streams)	748
7.2.3	Klassen für String-Streams	763
7.2.4	Die Klasse »streambuf«	770
7.2.5	Die Klasse »filebuf«	774
7.2.6	Die Klasse »stringbuf«	775

7.3	Numerische Bibliothek(en)	776
7.3.1	Komplexe Zahlen (»complex«-Klasse)	776
7.3.2	valarray	779
7.3.3	Globale numerische Funktionen (»cmath« und »cstdlib«)	802
7.3.4	Grenzwerte von Zahlentypen	806
7.3.5	Halbnumerische Algorithmen	811
7.4	Typerkennung zur Laufzeit	814

8 Weiteres zum C++-Guru 821

8.1	Module	821
8.1.1	Aufteilung	822
8.1.2	Die öffentliche Schnittstelle (Header-Datei)	823
8.1.3	Die private Datei	824
8.1.4	Die Client-Datei	826
8.1.5	Speicherklassen »extern« und »static«	827
8.1.6	Werkzeuge	829
8.2	Von C zu C++	830
8.2.1	Notizen	831
8.2.2	Kein C++	831
8.2.3	Kein C	833
8.2.4	»malloc« und »free« oder »new« und »delete«	834
8.2.5	»setjmp« und »longjmp« oder »catch« und »throw«	835
8.3	»Altes« C++	835
8.3.1	Header-Dateien mit und ohne Endung	835
8.3.2	Standardbibliothek nicht komplett oder veraltet	836
8.3.3	Namespaces (Namensbereiche)	836
8.3.4	Schleifenvariable von »for«	836
8.4	UML	837
8.4.1	Wozu UML?	837
8.4.2	UML-Komponenten	839
8.4.3	Diagramme erstellen	840
8.4.4	Klassendiagramme mit UML	840
8.5	Programmierstil	881
8.5.1	Kommentare	881
8.5.2	Code	883
8.5.3	Benennung	884
8.5.4	Codeformatierung	884
8.5.5	Zusammenfassung	885
8.6	Entwicklungsstufen von Software	886
8.6.1	Auftrag bzw. Idee	887

8.6.2	Spezifikation und Anforderung	888
8.6.3	Entwurf (Design)	889
8.6.4	Programmieren (Codieren)	890
8.6.5	Testen und Debuggen	890
8.6.6	Freigabe (Release)	891
8.6.7	Wartung	891
8.6.8	Aktualisierung (Update)	892
8.7	Boost	892
8.7.1	Boost.Regex (reguläre Ausdrücke)	894
8.7.2	Boost.lostreams	909
8.7.3	Boost.Filesystem	911

9 Netzwerkprogrammierung und Cross-Plattform-Entwicklung in C++ 917

9.1	Begriffe zur Netzwerktechnik	918
9.1.1	IP-Nummern	918
9.1.2	Portnummern	919
9.1.3	Host- und Domainname	920
9.1.4	Nameserver	921
9.1.5	IP-Protokoll	921
9.1.6	TCP und UDP	921
9.1.7	Was sind Sockets?	922
9.2	Header-Dateien zur Socketprogrammierung	923
9.2.1	Linux/UNIX	923
9.2.2	Windows	923
9.3	Client-Server-Prinzip	926
9.3.1	Loopback-Interface	926
9.4	Erstellen einer Client-Anwendung	927
9.4.1	»socket()« – Erzeugen eines Kommunikations- endpunkts	927
9.4.2	»connect()« – Client stellt Verbindung zum Server her	929
9.4.3	Senden und Empfangen von Daten	934
9.4.4	»close()« und »closesocket()«	937
9.5	Erstellen einer Server-Anwendung	937
9.5.1	»bind()« – Festlegen einer Adresse aus dem Namensraum	938
9.5.2	»listen()« – Warteschlange für eingehende Verbindungen einrichten	939
9.5.3	»accept()« und die Server-Hauptschleife	940

9.6	Cross-Plattform-Development	943
9.6.1	Abstraction Layer	943
9.6.2	Header-Datei (»socket.h«)	943
9.6.3	Quelldatei (»socket.cpp«)	945
9.6.4	TCP-Echo-Server (Beispiel)	956
9.6.5	Exception-Handling integrieren	958
9.6.6	Server- und Client-Sockets erstellen (TCP)	964
9.6.7	Ein UDP-Beispiel	974
9.7	Mehrere Clients gleichzeitig behandeln	976
9.8	Weitere Anmerkungen zur Netzwerkprogrammierung	986
9.8.1	Das Datenformat	986
9.8.2	Der Puffer	987
9.8.3	Portabilität	988
9.8.4	Von IPv4 nach IPv6	988
9.8.5	RFC-Dokumente (Request for Comments)	990
9.8.6	Sicherheit	990
9.8.7	Fertige Bibliotheken	991

10 GUI- und Multimediaprogrammierung in C++ 993

10.1	GUI-Programmierung – Überblick	993
10.1.1	Low-Level	994
10.1.2	High-Level	994
10.1.3	Überblick über plattformunabhängige Bibliotheken ...	995
10.1.4	Überblick über plattformabhängige Bibliotheken	997
10.2	Multimedia- und Grafikprogrammierung – Überblick	998
10.2.1	Überblick über plattformunabhängige Bibliotheken ...	998
10.2.2	Überblick über plattformabhängige Bibliotheken	1000
10.3	GUI-Programmierung mit »wxWidgets«	1001
10.3.1	Warum »wxWidgets«?	1001
10.3.2	Das erste Programm – Hallo Welt	1002
10.3.3	Die grundlegende Struktur eines »wxWidgets«- Programms	1005
10.3.4	Event-Handle (Ereignisse behandeln)	1012
10.3.5	Die Fenster-Grundlagen	1019
10.3.6	Übersicht über die »wxWidgets«-(Fenster-)Klassen ...	1021
10.3.7	»wxWindow«, »wxControl« und »wxControlWithItems« – die Basisklassen	1023
10.3.8	Top-Level-Fenster	1026
10.3.9	Container-Fenster	1050
10.3.10	Nicht statische Kontrollelemente	1077
10.3.11	Statische Kontrollelemente	1135

10.3.12	Menüs	1140
10.3.13	Ein Beispiel – Text-Editor	1156
10.3.14	Standarddialoge	1171
10.3.15	Weitere Elemente und Techniken im Überblick	1195

A Anhang **1207**

A.1	Operatoren in C++ und deren Bedeutung (Übersicht)	1207
A.2	Vorrangtabelle der Operatoren	1209
A.3	Schlüsselwörter von C++	1210
A.4	Informationsspeicherung	1210
A.4.1	Zahlensysteme	1211
A.5	Zeichensätze	1218
A.5.1	ASCII-Zeichensatz	1219
A.5.2	ASCII-Erweiterungen	1220
A.5.3	Unicode	1222
Index		1225