

Ulla Kirch | Peter Prinz

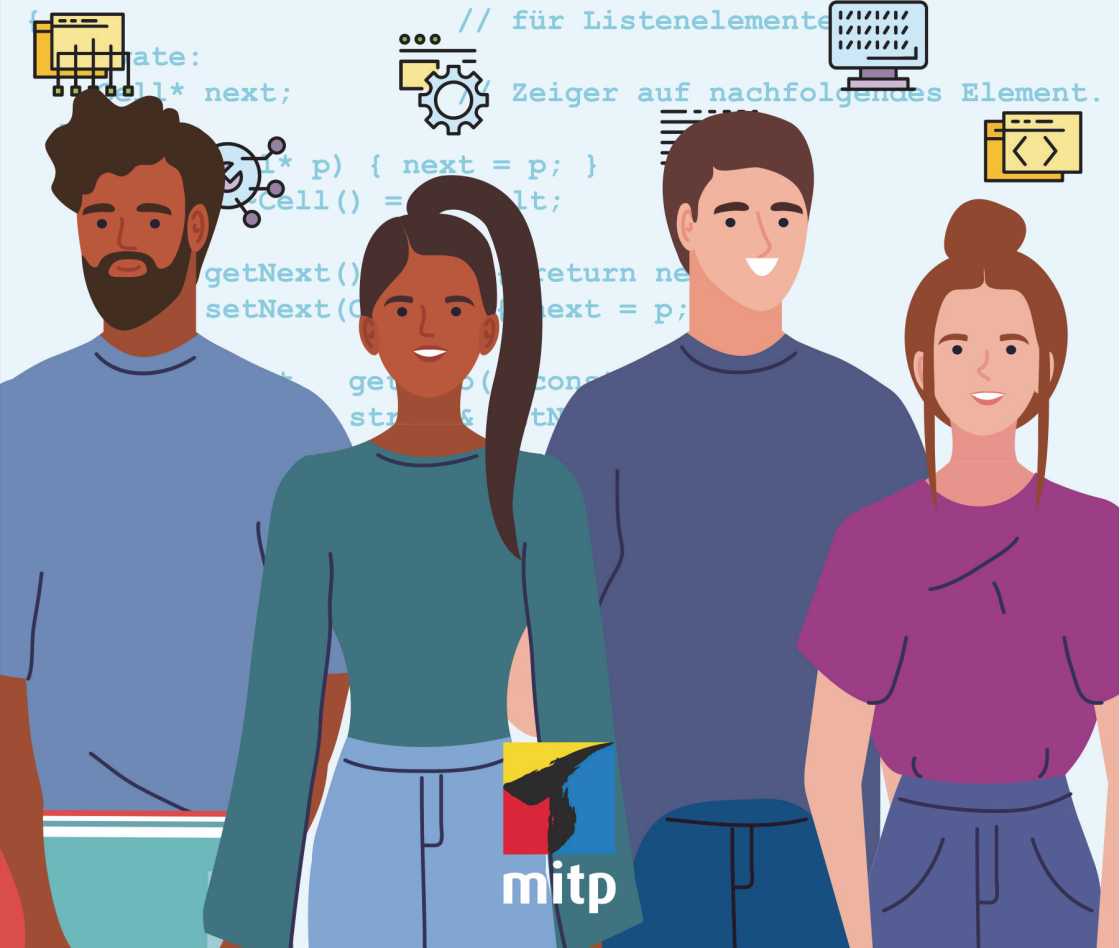
C++

DAS ÜBUNGSBUCH

Testfragen und Aufgaben mit Lösungen

6. AUFLAGE

```
class Cell // Abstrakte Basisklasse
{ // für Listenelemente
public:
    Cell* next; // Zeiger auf nachfolgendes Element.
    Cell(const T& p) { next = p; }
    Cell() = default;
    T* getNext() const { return next; }
    void setNext(Cell* p) { next = p; }
};
```



Inhaltsverzeichnis

	Einleitung	11
1	Grundlagen	13
	Verständnisfragen	14
	Aufgaben	16
	Lösungen zu den Verständnisfragen	19
	Lösungen zu den Aufgaben	20
2	Elementare Datentypen, Konstanten und Variablen	23
	Verständnisfragen	24
	Aufgaben	26
	Lösungen zu den Verständnisfragen	28
	Lösungen zu den Aufgaben	29
3	Verwenden von Funktionen und Klassen	33
	Verständnisfragen	34
	Aufgaben	36
	Lösungen zu den Verständnisfragen	40
	Lösungen zu den Aufgaben	41
4	Ein- und Ausgaben mit Streams	45
	Verständnisfragen	46
	Aufgaben	48
	Lösungen zu den Verständnisfragen	51
	Lösungen zu den Aufgaben	52
5	Operatoren für elementare Datentypen	57
	Verständnisfragen	58
	Aufgaben	60
	Lösungen zu den Verständnisfragen	63
	Lösungen zu den Aufgaben	64
6	Kontrollstrukturen	69
	Verständnisfragen	70
	Aufgaben	74

	Lösungen zu den Verständnisfragen	77
	Lösungen zu den Aufgaben	78
7	Symbolische Konstanten und Makros	85
	Verständnisfragen	86
	Aufgaben	89
	Lösungen zu den Verständnisfragen	92
	Lösungen zu den Aufgaben	92
8	Umwandlung arithmetischer Datentypen	99
	Verständnisfragen	100
	Aufgaben	103
	Lösungen zu den Verständnisfragen	106
	Lösungen zu den Aufgaben	107
9	Die Standardklasse string	109
	Verständnisfragen	110
	Aufgaben	113
	Lösungen zu den Verständnisfragen	115
	Lösungen zu den Aufgaben	116
10	Funktionen	123
	Verständnisfragen	124
	Aufgaben	126
	Lösungen zu den Verständnisfragen	130
	Lösungen zu den Aufgaben	131
11	Speicherklassen und Namensbereiche	139
	Verständnisfragen	140
	Aufgaben	143
	Lösungen zu den Verständnisfragen	147
	Lösungen zu den Aufgaben	148
12	Referenzen und Zeiger	157
	Verständnisfragen	158
	Aufgaben	161
	Lösungen zu den Verständnisfragen	165
	Lösungen zu den Aufgaben	166
13	Definition von Klassen	171
	Verständnisfragen	172
	Aufgaben	175

	Lösungen zu den Verständnisfragen	179
	Lösungen zu den Aufgaben	179
14	Methoden	189
	Verständnisfragen	190
	Aufgaben	192
	Lösungen zu den Verständnisfragen	196
	Lösungen zu den Aufgaben	197
15	Teilobjekte und statische Elemente	207
	Verständnisfragen	208
	Aufgaben	210
	Lösungen zu den Verständnisfragen	215
	Lösungen zu den Aufgaben	216
16	Vektoren	223
	Verständnisfragen	224
	Aufgaben	226
	Lösungen zu den Verständnisfragen	232
	Lösungen zu den Aufgaben	233
17	Zeiger und Vektoren	243
	Verständnisfragen	244
	Aufgaben	247
	Lösungen zu den Verständnisfragen	251
	Lösungen zu den Aufgaben	252
18	Grundlagen der Dateiverarbeitung	259
	Verständnisfragen	260
	Aufgaben	262
	Lösungen zu den Verständnisfragen	266
	Lösungen zu den Aufgaben	267
19	Operatoren überladen	283
	Verständnisfragen	284
	Aufgaben	286
	Lösungen zu den Verständnisfragen	294
	Lösungen zu den Aufgaben	294
20	Typumwandlung für Klassen	311
	Verständnisfragen	312
	Aufgaben	315

	Lösungen zu den Verständnisfragen	320
	Lösungen zu den Aufgaben	320
21	Speicherreservierung zur Laufzeit	329
	Verständnisfragen	330
	Aufgaben	333
	Lösungen zu den Verständnisfragen	340
	Lösungen zu den Aufgaben	341
22	Dynamische Elemente	355
	Verständnisfragen	356
	Aufgaben	359
	Lösungen zu den Verständnisfragen	371
	Lösungen zu den Aufgaben	371
23	Vererbung	395
	Verständnisfragen	396
	Aufgaben	398
	Lösungen zu den Verständnisfragen	406
	Lösungen zu den Aufgaben	407
24	Typumwandlungen in Klassenhierarchien	431
	Verständnisfragen	432
	Aufgaben	436
	Lösungen zu den Verständnisfragen	440
	Lösungen zu den Aufgaben	440
25	Polymorphe Klassen	455
	Verständnisfragen	456
	Aufgaben	459
	Lösungen zu den Verständnisfragen	470
	Lösungen zu den Aufgaben	470
26	Abstrakte Klassen	495
	Verständnisfragen	496
	Aufgaben	498
	Lösungen zu den Verständnisfragen	504
	Lösungen zu den Aufgaben	505

27	Mehrfachvererbung	517
	Verständnisfragen	518
	Aufgaben	521
	Lösungen zu den Verständnisfragen	526
	Lösungen zu den Aufgaben	527
28	Ausnahmebehandlung	543
	Verständnisfragen	544
	Aufgaben	546
	Lösungen zu den Verständnisfragen	551
	Lösungen zu den Aufgaben	552
29	Mehr über Dateien	563
	Verständnisfragen	564
	Aufgaben	566
	Lösungen zu den Verständnisfragen	576
	Lösungen zu den Aufgaben	577
	Stichwortverzeichnis	601

Einleitung

Dieses Buch wendet sich an Leser, die ihre C++-Kenntnisse durch »Learning by Doing« vertiefen möchten. Es ist ideal, um sich im Stil eines Workshops auf Prüfungen oder auf die Mitarbeit in einem C++-Projekt vorzubereiten.

Die Gliederung des Stoffs entspricht der des Buches »C++ Lernen und professionell anwenden«, das ab der 9. Auflage den neuesten C++-Standard von 2020 (kurz C++20) berücksichtigt. Neue Sprachelemente sind in aktuellen Compilern noch nicht voll integriert und werden deshalb in diesem Buch mit ^(*) gekennzeichnet.

Aber es ist nicht wesentlich, wie Sie C++ gelernt haben. Jedes Kapitel beginnt mit einer Zusammenfassung des Stoffs, zu dem anschließend Fragen und Aufgaben gestellt werden. Beispielsweise ist das Thema des 9. Kapitels »Die Standardklasse `string`«. Wenn Ihnen dann der Inhalt der Zusammenfassung zur `string`-Klasse vertraut ist, sollten Sie auch ohne größere Probleme die anschließenden Fragen und Aufgaben lösen können.

Jedes Kapitel besteht neben der einführenden Beschreibung des Themas aus drei weiteren Teilen: Verständnisfragen, Programmieraufgaben und den Musterlösungen zu allen Fragen und Aufgaben. Mit jeweils 20 Verständnisfragen können Sie testen, wie gut Sie sich in dem jeweiligen Themenbereich auskennen. Die Art der Fragen sind entweder Ja-Nein-Fragen, Multiple-Choice-Fragen oder es muss eine Aussage vervollständigt werden.

Im Aufgabenteil können Sie dann Ihr Wissen praktisch umsetzen. In jedem Kapitel gibt es mindestens zehn Aufgaben mit steigendem Schwierigkeitsgrad. Die Bearbeitung einfacher Aufgaben ist oft in wenigen Minuten erledigt. Dagegen kann die Lösung umfangreicher Aufgaben auch Tage in Anspruch nehmen. Dies gilt insbesondere bei Aufgaben zu den Themen »Dynamische Elemente«, »Vererbung« und »Polymorphie«. Umfangreichere Problemstellungen sind dabei oft auf mehrere Aufgaben verteilt.

Bei der Auswahl der Problemstellungen für Aufgaben wurde stets darauf geachtet, dass sie typisch und praxisnah sind. Auf diese Weise lernen Sie viele interessante Algorithmen und Datenstrukturen kennen. Auch durch die eigenständige Implementierung von »Iteratoren« und »Intelligenten Zeigern« vertiefen Sie Ihr Verständnis für die Konzepte der Standardbibliothek. In jedem Fall verfügen Sie nach der Durcharbeitung des Buches über fundierte Programmiererfahrungen und einen umfangreichen Fundus von Beispiel-Code.

Trotz ausführlicher Aufgabenstellungen und vieler Hinweise kann es immer mal vorkommen, dass man nicht zum Ziel kommt. Dann hilft ein Blick in die kommentierten Musterlösungen. Außerdem ist es sicher immer interessant, die eigene Lösung mit der im Buch zu vergleichen. Die Musterlösungen finden Sie auch im Internet unter <http://www.mitp.de/0637>.

Dem Leser wünschen wir viele Erfolgserlebnisse beim Lösen der Übungen.

Ulla Kirch
kirch@hm.edu

Peter Prinz
prinz_peter@t-online.de

Grundlagen

Dieses Kapitel umfasst grundlegende Fragen und Aufgaben zur Erstellung von C++-Programmen. Hierzu zählen auch das

- **Inkludieren von Header-Dateien**
Eine Header-Datei beinhaltet Informationen, die von einem C++-Programm verwendet werden. In der Header-Datei `iostream` beispielsweise sind Informationen enthalten, die zur Ein-/Ausgabe von Daten erforderlich sind. Eine Header-Datei wird mit der `#include`-Direktive in ein Programm kopiert.
- **Verwenden der `using`-Direktive**
Vordefinierte Namen, wie z.B. `cout`, gehören zum Namensbereich `std`. Die Direktive `using namespace std`; ermöglicht es, diese Namen ohne den Vorsatz `std::` direkt zu verwenden.
- **Formulieren von Anweisungen**
Eine Anweisung legt fest, was das Programm tun soll, und wird stets mit einem Semikolon abgeschlossen. Zur Ausgabe von Daten auf den Bildschirm wird in C++ der Stream `cout` verwendet, z.B. `cout << "Hallo";`
- **Definieren einer `main`-Funktion**
Die erste Funktion, die in einem C++-Programm ausgeführt wird, ist stets die `main`-Funktion. Die auszuführenden Anweisungen stehen im Funktionsblock, d.h. innerhalb der Klammern `{ }`. Bei Erreichen der `return`-Anweisung wird die Funktion verlassen.
- **Kommentieren von Quelldateien**
Kommentare dienen zur Dokumentation in einem Programm. Sie verbessern die Lesbarkeit und können bei der Fehlersuche nützlich sein. Jede Zeichenfolge, die in `/* ... */` eingeschlossen ist oder mit `//` beginnt, ist ein Kommentar. Der Compiler ignoriert Kommentare.

Verständnisfragen

- 1.1 C++ ist eine rein objektorientierte Sprache.
☐ Richtig ☐ Falsch
- 1.2 Die umfangreiche in C entwickelte Software kann auch in C++-Programmen verwendet werden.
☐ Richtig ☐ Falsch
- 1.3 Eine Quelldatei wird zur Übersetzung an den _____ übergeben.
- 1.4 Der _____ bindet eine Objektdaten mit anderen Modulen zu einer ausführbaren Datei.
- 1.5 Die gebräuchlichsten Endungen im Namen von Quelldateien sind
a) .c b) .cpp c) .cc
- 1.6 Standardisierte Funktionen und Klassen sind in der _____ enthalten.
- 1.7 Bei der Suche nach Fehlern in einem C++-Programm beginnen Sie immer mit
a) dem letzten vom Compiler angezeigten Fehler.
b) irgendeinem angezeigten Fehler.
c) dem ersten angezeigten Fehler.
- 1.8 Eine Warnung kann einen
a) Syntaxfehler anzeigen.
b) logischen Fehler anzeigen.
c) Laufzeitfehler anzeigen.
- 1.9 Jedes C++-Programm enthält die Funktion _____.
- 1.10 In einem C++ Programm bedeutet das Doppelkreuz # am Anfang einer Zeile, dass diese Zeile für
a) den Compiler bestimmt ist.
b) den Präprozessor bestimmt ist.
c) die Header-Datei bestimmt ist.
- 1.11 Vordefinierte Namen der C++-Standardbibliothek befinden sich im Namensbereich _____.

- 1.12 Die Programmausführung beginnt (abgesehen von der Initialisierung globaler Objekte) mit
- a) der ersten `#include`-Direktive.
 - b) der ersten Anweisung in der Funktion `main()`.
 - c) der zuerst definierten Funktion.
- 1.13 Der Name `cout` bezeichnet ein Objekt, das zuständig ist für
- a) Eingaben.
 - b) den Programmstart.
 - c) Ausgaben.
- 1.14 In der Funktion `main()` bewirkt die Anweisung
- ```
return 0;
```
- a) das Verlassen von `main()`.
  - b) die Beendigung des Programms.
  - c) die Rückgabe des Exitcode 0 an das aufrufende Programm.
- 1.15 Die kürzeste Anweisung besteht aus \_\_\_\_\_.
- 1.16 C++-Funktionen müssen in einer bestimmten Reihenfolge definiert werden.
- ☐ Richtig    ☐ Falsch
- 1.17 Die erste Funktion, die in einer Quelldatei definiert wird, ist stets die Funktion `main()`.
- ☐ Richtig    ☐ Falsch
- 1.18 Die Anweisungen, die in der Funktion `main()` ausgeführt werden, stehen im \_\_\_\_\_.
- 1.19 Zeichenfolgen werden als Kommentare interpretiert, wenn sie
- a) mit `/*` beginnen.
  - b) in `/*    */` eingeschlossen sind.
  - c) mit `//` beginnen.
- 1.20 In einer Zeile können mehrere Präprozessor-Direktiven angeführt werden.
- ☐ Richtig    ☐ Falsch

## Aufgaben

- 1.1 Was gibt das folgende Programm auf dem Bildschirm aus?

```
#include <iostream>
using namespace std;

int main()
{
 cout << "Hi Leute, ";
 cout << endl;
 cout << "was habt Ihr heute noch vor";
 cout << "?" << endl;
 return 0;
}
```

- 1.2 Formulieren Sie die entsprechenden Anweisungen, um

```
Mir geht's gut!
```

- a) beginnend bei der aktuellen Cursorposition auszugeben.  
b) am Anfang der nächsten Zeile auszugeben.
- 1.3 Jedes der folgenden Programme enthält einen Fehler. Bestimmen und korrigieren Sie jeden Fehler.
- a)

```
#include <iostream>
int main()
{ // Und jetzt kommt der berühmteste Spruch
 // aus der Welt der Programmiersprachen:
 cout << "Hello, World!" << endl;
 return 0;
}
```

b)

```
#include <iostream>
using namespace std;
int main()
{
 cout >> "Hello, World!" >> endl;
}
```

c)

```
#include <iostream>
using namespace std;
int main()
{
 / Wer zum Teufel hat das gesagt? /
 cout << "Hello, World!" << endl;
 return 0;
}
```

d)

```
#include <iostream>
using namespace std;
int main()
{
 cout << "Hallo, Universum! ";
 << endl;
 return 0;
}
```

- 1.4 Schreiben Sie ein C++-Programm, das Ihren Namen, Ihre Adresse, Telefonnummer und E-Mail-Adresse in je einer Zeile auf dem Bildschirm ausgibt.
- 1.5 Fügen Sie Kommentare in die Lösung zur Aufgabe 1.4 ein, und zwar einen Programmnamen, den Namen des Programmierers sowie eine Beschreibung, was das Programm macht.
- 1.6 Schreiben Sie ein C++-Programm, das folgendes Menü ausgibt:

```
***** Telefonverzeichnis *****

E = Neuen Eintrag einfüegen
L = Eintrag loeschen
S = Telefonnummer suchen
A = Alle Eintraege anzeigen
B = Programm beenden

Ihre Wahl:
```

## 1.7 Sind die folgenden C++-Programme vollständig und fehlerfrei?

a)

```
int main()
{
 return 0;
}
```

b)

```
include <iostream>
using namespace std;
int main()
{
 cout << "Hey, los!" << return 0;
}
```

c)

```
#include <iostream>
using namespace std;
int main(
){
 cout <<
 "Das wär's für heute!" << endl; return 0
;}
```

## 1.8 Angenommen, die folgenden Anweisungen befinden sich in einer main-Funktion. Was ist falsch?

a) cout &gt;&gt; "Weiter mit &lt;return&gt;" &gt;&gt; endl;

b) return "Alles klar!";

c) cout "&lt;&lt; Geben Sie eine Zahl ein: &lt;&lt;" endl;

## 1.9 Verfolgen Sie den Ablauf des folgenden C++-Programms und beschreiben Sie, was auf dem Bildschirm ausgegeben wird.

```
#include <iostream>
using namespace std;

void star1(), star2(), star3();

int main()
{
```

```

 star1();
 star2();
 star3();
 star2();
 star1();
 return 0;
}

void star1() { cout << "****" << endl; }

void star2() { cout << "*****" << endl; }

void star3() { cout << "*****" << endl; }

```

- 1.10 Ändern Sie die `main`-Funktion aus der letzten Aufgabe so, dass folgende Grafik ausgegeben wird:

```



```

Fügen Sie außerdem Kommentare in den Quellcode ein und erklären Sie, was das Programm macht.

## Lösungen zu den Verständnisfragen

- 1.1 Falsch (C++ ist eine Erweiterung der prozeduralen Programmiersprache C.)
- 1.2 Richtig
- 1.3 Compiler
- 1.4 Linker
- 1.5 b) und c)
- 1.6 C++-Standardbibliothek
- 1.7 c)
- 1.8 b)
- 1.9 `main()`
- 1.10 b)

- I.II std
- I.I2 b)
- I.I3 c)
- I.I4 a), b) und c)
- I.I5 einem Semikolon
- I.I6 Falsch
- I.I7 Falsch
- I.I8 Funktionsblock von `main()`
- I.I9 b) und c)
- I.I20 Falsch

## Lösungen zu den Aufgaben

I.I

```
Hi Leute,
was habt Ihr heute noch vor?
```

I.2

```
cout << "Mir geht's gut!";
cout << endl << "Mir geht's gut!";
(oder: cout << "\nMir geht's gut!";)
```

I.3 a) Hinter der Direktive `#include <iostream>` fehlt in einer neuen Zeile:

```
using namespace std;
```

Alternativ kann auch `std::cout` und `std::endl` verwendet werden.

b) Statt `>>` ist der Operator `<<` zu verwenden. Die abschließende Anweisung `return 0;` darf fehlen. Sie wird dann vom Compiler eingefügt.

c) Innerhalb der `main()`-Funktion ist der Kommentar syntaktisch nicht korrekt. Richtig wäre beispielsweise:

```
// Wer zum Teufel hat das gesagt?
/* Wer zum Teufel hat das gesagt? */
```

d) In der ersten Zeile im Rumpf der `main()`-Funktion muss das Semikolon entfernt werden.

I.4

```
#include <iostream>
using namespace std;
int main()
{
 cout << "Sarah Miller" << endl
 << "Karenstr. 123 " << endl
 << "80123 Muenchen" << endl
 << "Tel. (089) 6543210" << endl
 << "sarah.m@yahoo.com" << endl;
 return 0;
}
```

I.5

```
// -----
// Programmname: ex01_05.cpp
// Autor: Sarah Miller
// Das Programm gibt einen Namen, eine Adresse, eine
// Tel.-Nr. und eine E-Mail-Adresse auf dem Bildschirm aus.
// -----
#include <iostream>
using namespace std;
int main()
{
 // Wie in der Lösung zur Aufgabe 1.4.
}
```

I.6

```
// -----
// ex01_06.cpp
// Gibt ein Menü für ein Telefonverzeichnis aus.
// -----
#include <iostream>
using namespace std;

int main()
{
 cout << "***** Telefonverzeichnis *****"
 << endl << endl;
 cout << " E = Neuen Eintrag einfuegen" << endl;
 cout << " L = Eintrag loeschen" << endl;
 cout << " S = Telefonnummer suchen" << endl;
 cout << " A = Alle Eintraege anzeigen" << endl;
 cout << " B = Programm beenden" << endl
 << endl;
}
```

```
cout << "Ihre Wahl: ";

cout << endl;
return 0;
}
```

- 1.7 a) Das Programm tut zwar nichts, der Quellcode ist aber fehlerfrei und vollständig.
- b) Im Quellcode liegen zwei Fehler vor:
1. Das Zeichen `#` fehlt vor `include`.
  2. `return 0;` muss als separate Anweisung angeführt werden, d.h. nicht als Teil der Anweisung `cout << ....;`.
- c) Der Quellcode ist fehlerfrei und vollständig, aber schlecht lesbar.
- 1.8 a) Anstelle von `>>` ist das Symbol `<<` zu verwenden, um den Text in den Ausgabestrom einzufügen.
- b) Bei dem Return-Wert der `main`-Funktion muss es sich um eine Ganzzahl handeln.
- c) Die Symbole `<<` müssen sich außerhalb des Strings "Geben Sie eine Zahl ein: " befinden.

1.9

```



```

1.10

```
// -----
// ex01_10.cpp
// Modifizierung des Programms aus Aufgabe 1.9.
// -----
int main()
{
 star3(); // Gibt 3*4 = 12 Sterne aus.
 star2(); // Gibt 2*4 = 8 Sterne aus.
 star1(); // Gibt 4 Sterne aus.
 star2(); // Gibt 8 Sterne aus.
 star3(); // Gibt 12 Sterne aus.
 return 0;
}
```

# Stichwortverzeichnis

## Symbole

#define 85  
#ifdef 85  
#ifndef 85  
#include 13, 33  
#undef 85  
=default 189  
=delete 189

## A

Ableitung 395  
    mehrfache 517  
Absolutbetrag  
    komplexer Zahl 291  
Abstand  
    Punkte- 463  
Addition  
    Polynom 361  
    von Zeigern 243  
Adresse 157  
    Vektorelement 243  
Adressoperator 157  
Algorithmus  
    Boyer-Moore 337  
    Euklid 130  
    Kongruenzmethode 144  
    Quick-Sort 368  
    Tree-Sort 368  
Aliasname 157  
Analyse  
    objektorientierte 171  
Anker 463  
Anweisung 13  
    break 69  
    catch 543  
    continue 69  
    do-while 69  
    for 69  
    goto 69  
    if-else 69  
    return 13, 124  
    switch 69  
    throw 543  
    while 69

Argument 33  
    Default- 123  
    Objekt 189  
argv 243  
Array 223  
ASCII  
    -Code 27  
assert.h 90  
assert() 90  
assoziativ  
    Vektor 292  
at() 109  
atof() 251  
Aufräumarbeit 189  
Aufruf  
    Funktions- 33, 124  
Aufrufumgebung 543  
Aufzählung 207, 214  
Ausdruck 57  
Ausgabe  
    formatierte 45  
    unformatierte 47  
Ausnahmebehandlung 543  
Auswahloperator 69  
AutoPtr 363

## B

back() 467  
bad\_cast 455  
Bad-Character shift 339  
Basis  
    Zahlensystem 45  
Basisinitialisierer 395, 517  
Basisklasse 395, 517  
    mehrfache indirekte 517  
    virtuelle 517  
Basisklassenzeiger 431,  
433, 455  
Baum  
    binärer 364  
    durchlaufen 367  
    -Ebene 366  
    Höhe 366  
    Lücke 367  
Bedingung 62, 69

bedingungsfrei 69  
begin() 467  
Bereichsoperator 139,  
395, 517  
Bereichsprüfung 335  
Bezeichner 142  
Beziehung  
    Hat- 396  
    Ist- 395  
Bibliothek  
    Standard- 33  
Bild 466  
binär 283  
    Baum 364  
    Operator 57  
    Suchbaum 365  
    Ziffernfolge 75  
Binärmodus 261  
Bindung  
    dynamische 455, 458  
    statisch 458  
Binomialkoeffizient 229  
Blank 25, 114  
Blatt 364  
Block 69  
Boyer, R.S. 337  
break 69  
Brief 403  
Buchstabe  
    Groß/Klein- 126

## C

C  
    Header-Datei 33  
    Kompatibilität 171  
C++  
    Standardbibliothek 33  
C++20 11  
Call  
    by Reference 157, 189  
    by Value 189  
Call by Reference 123  
Call by Value 123  
cassert 90

- Cast
  - Down- 431, 455
  - dynamic- 431
  - Operator 431
  - static- 431
  - Typ- 99
  - Up- 431
- catch-Block 543
- cctype 85
- ceil() 39
- Celsius 61
- char16\_t 23
- char32\_t 23
- cin 45
- Circle 464
- class 171
- clear() 468
- Client 501
- limits 49, 337
- close() 259
- Code 262
  - ASCII- 27
  - Exit- 262
- Compilierung
  - bedingte 85, 88
- Complex 290, 319
- const
  - Datenelement 207
  - Methode 191
  - Parameter 243
- const\_iterator 467
- continue 69
- cos() 37
- Cosinus 37
- cout 13, 15, 25, 35, 45
- cstdlib 177, 251
- C-String 223, 225
- cstring 223
- ctime 145, 524
- ctime() 524
- ctype.h 402
- Customer 264
- CustomerFile 264
- D**
- Darstellung
  - binär 75
  - dezimal 75
  - hexadezimal 23, 45
  - oktal 23
  - Potenz- 115
- Data 437
- Datei 259
  - einlesen 263, 290
- Header- 33
  - mischen 266
  - öffnen 259
  - Position 260
  - schließen 259
  - schreiben 263, 290
- Dateizugriff 260
- Datenabstraktion 395
- Datenelement 33
  - dynamisches 355
  - konstantes 207
  - statisches 207
- Datenkapselung 283
- Datentyp 23, 24
  - const\_iterator 467
  - Hierarchie 99
  - iterator 467
  - list 467
  - time\_t 406
- DayTime 317
- dec 45
- Default
  - Argument 123
  - Destruktor 189
  - Konstruktor 190, 517
  - Methode 358
- Definition
  - Funktions- 123
  - Klassen- 171
  - Methode 172
  - Objekt- 171
  - String- 39
  - Union 171
  - Variable 23, 25
  - Vektor 223, 227
- Deklaration 140
  - extern 139, 140
  - friend 283
  - Funktions- 33
  - using 139, 141
- Dekrementoperator 58
- delete 329, 455
- Dereferenzieren 157
- Destruktor 189
  - virtuell 455, 457
- dezimal 23
- Dezimalzahl 48
- Differenz 57, 126
  - komplexer Zahlen 291
  - Zeiger 243
- Dimension
  - Vektor- 223
- Direktive
  - #define 85
  - #include 13, 33
  - #undef 85
  - using 13, 139
- Division 57
  - Polynom 361
- Divisionsrest 163
- Dokumentation 13
- Dollar 214
- Doppelkreuz 14
- double 23
- do-while 69
- Down-Cast 431, 455
  - sicherer 437, 455
- Dreieck
  - Pascal'sches 229
- Durchlauf
  - sequentieller 466
- Durchschnitt 60, 144, 227, 251
- dynamic\_cast<> 431, 455
- dynamisch
  - Bindung 458
  - Speicher 329
  - Vektor 332
- E**
- Ebene 366
- Eigenschaft 175
- Eingabe
  - formatierte 45
  - unformatierte 47
- Einheit
  - imaginäre 290
- Einheitswürfel 195
- Element
  - Klassen- 171
  - private 171
  - protected 398
  - public 171
  - redefiniertes 395
- Elementinitialisierer 207
- Ellipse 464
- Elternknoten 364
- empty() 467
- end() 467
- Entfernungstabelle 337
- enum 207
- eof 259
- eof() 259
- erase 109
- erase() III, 468
- Eratosthenes
  - Sieb des 570
- Eröffnungsmodus 259, 261

Ersatztext 86  
 Erweiterung  
   Ganzzahl 99  
 Escape-Sequenz 23, 25  
 Euklid 130  
 Exception 109  
   auslösen 543  
   bad\_cast 455  
   Deklaration 543  
   -Handler 543  
 exception  
   Standardfehlerklasse  
     545, 548  
 exception handling 543  
 Exit-Code 262  
 explicit 311  
 explizit  
   Typumwandlung 99  
 exponentiell  
   Notation 23, 45  
 extern 139, 140

**F**

Fahrenheit 60  
 fail() 259  
 false 23  
 Farbe 195  
 Fehler  
   Dateizugriff 259  
   -Flags 47  
 Fehlerbehandlung 543  
 Fehlerklasse 543  
 Fehlermeldung 33  
 Fehlerobjekt 543  
 Fehlersuche 13, 14  
 Feldbreite 45, 47  
 Festpunkt-Notation 46  
 Festpunktzahl 45  
 Fibonaccizahl 130  
 Figur  
   zweidimensionale 462  
 Filterprogramm 85  
 find() 109, 114, 162  
 fixed 45  
 Fläche  
   Kreis- 146  
 Flächeninhalt  
   Quadrat 27  
 Flag  
   Fehler- 47  
   Formatierungs- 45  
   Öffnungsmodus 259  
   Status- 259

float 23  
 FloatULong 178  
 floor() 127  
 fmod() 163  
 for 69, 71  
 Formatierung  
   -Flags 46  
 Freund 365  
 friend  
   Funktion 283  
   Klasse 283  
 front() 467  
 fstream 259  
 Füllzeichen 45  
 Funktion  
   Aufruf 38, 124  
   -Block 123  
   Deklaration 33  
   friend- 283  
   inline 123, 125  
   Konvertierungs- 311  
   -Kopf 123  
   main() 27  
   Operator- 283  
   rekursiv 123, 126  
   Speicherklasse 139  
   static- 139  
   überladen 123, 128  
   Zeigerversion 243  
 fußgesteuert 69

**G**

Ganzzahl 25, 46  
   -Erweiterung 99  
 Garbage Collector 329  
 Geldbetrag  
   runden 214  
 Geldbörse 286  
 Geldstück 176  
 Geldwechsel 214  
 Geltungsbereich 207  
 Genauigkeit 24, 45  
 Generator  
   Zufallszahlen- 145  
 geometrisch 248  
 get() 48, 259  
 getline() 48, 259, 439  
 Gleitpunktdarstellung 49  
 Gleitpunkttyp 23  
 Gleitpunktzahl 23, 37, 45, 49  
   runden 106  
 global 25, 35, 139  
   Operatorfunktion 283

Good-Suffix Shift 339  
 goto 69, 73  
 Grad  
   Polynom 248, 360  
 Grenzwert 26  
 Groß-/Kleinbuchstabe 126  
 Groß-/Kleinschreibung 23

**H**

Halbachse 465  
 Handler  
   Exception- 543  
   New- 331  
 Handy 193  
 Hanoi 230  
 Hat-Beziehung 208, 396  
 Header-Datei 13, 33, 49, 109  
   assert.h 90  
   cassert 90  
   cctype 85  
   climits 49, 337  
   cstdlib 177, 251  
   cstring 223  
   ctime 145  
   ctype.h 402  
   fstream 260  
   iostream 13  
   list 466  
   math.h 163  
   stdlib.h 177  
   string 109  
   time.h 145  
   typeinfo 461  
 hex 45  
 Hexadezimalzahl 47, 48  
 Hierarchie  
   Datentyp 99  
 Hochkomma 23  
 Höhe  
   Baum 366  
   Rechteck 465  
   Zylinder 127  
 Horner-Schema 115, 248, 361

**I**

if-else 69  
 ifstream 259  
 imaginär 290  
 Imaginärteil 290  
 implizit  
   Typumwandlung 99

- Import
  - Namesbereich 142
- Index 109, 112, 223, 224
- Indexoperator 109, 289
  - überladen 293
- inhomogen 466
- Initialisierung
  - Liste 329
  - Teilobjekte Siehe
  - Variablen- 23, 25
  - Vektor 223
- Inkrementoperator 58
- inline 125
  - Methode 189
- insert() 109, 468
- Instanz 35
- IntArr 335, 359
- intelligent 363
- Interface
  - öffentliches 366
- Invertieren
  - String 163
- ios 45, 259
- iostream 35
- isalpha() 85
- iscntrl() 91
- isdigit() 85
- islower() 85
- ISO-Standard 11
- isspace() 402
- Ist-Beziehung 395
- istream 45
- isupper() 85
- iterativ 366
- Iterator 362, 466
- iterator 467
- J**
  - Job 335
  - JobList 337
- K**
  - kartesisch 178, 288
  - Keim 144, 177
  - Kette
    - else-if 69
  - Kindknoten 364
  - Klammer 57
  - Klasse 33, 290, 337
    - abgeleitete 395
    - abstrakte 495
    - Address 403
    - AutoPtr 363
    - Basis- 395
    - Circle 400, 464
    - Complex 319
    - Customer 264
    - Data 437
    - DayTime 317
    - definieren 171
    - Ellipse 464
    - Email 549
    - friend 283
    - IntArr 335, 359
    - Iterator 362
    - Job 335
    - JobList 337
    - konkrete 495
    - KundenService 525
    - Letter 404
    - Line 464
    - LongArr 356
    - Mail 403
    - MailService 404
    - MeasureArr 356
    - mehrfach vererben 517
    - MobilePhone 193, 213, 250
    - MoneyChanger 214
    - MyData 438, 462
    - MyString 402
    - NameValueArr 292
    - Node 365
    - PackedFood 500
    - Pair 315, 318
    - Parcel 404
    - PhoneCall 523
    - PiggyBank 176, 194
    - Player 177
    - Point 463
    - Point3D 288, 315
    - Polygon 464
    - Polyline 463
    - polymorph 455
    - Polynomial 360
    - PrioQueue 503
    - Product 500
    - Random 177, 193
    - Rectangle 401, 464
    - RGB\_Color 195, 215
    - RGB-Color 195
    - SearchTree 365
    - Shape2D 400
    - ShapePtrList 466
    - Square 401
    - StringMatching 338
    - TimeStamp 524
    - TowersOfHanoi 230
    - TraceParcel 405
    - UnpackedFood 500
    - Wallet 286
  - Klasse *siehe auch* Standard-  
klasse
  - Klasse.Point2D 316
  - Klassen-Array 223
  - Klassenhierarchie 398, 405, 431
  - Klassenvariable 207
  - Knoten 364
    - einfügen 366
    - innerer 364
    - löschen 368
  - Kommandozeile 243, 251
  - Kommentar 13, 17
  - komplex 290
  - Komprimieren
    - Vektor 335
  - Kongruenzmethode 144
  - Konstante 24
    - klassenspezifische 207
    - numerische 23
    - String- 23
    - symbolische 85
  - Konstruktor 189, 190
    - Default- 190
    - Konvertierungs- 311
    - Kopier- 355
    - Move- 355
  - Kontrollstruktur 69
  - Konvertierung
    - Datentyp 99
    - Funktion 311
    - in Klassenhierarchien 431
    - Konstruktor 311
  - Konvertierung *siehe* Typum-  
wandlung
  - Koordinaten
    - kartesische 288
    - zweidimensionale 316
  - Koordinatensystem
    - kartesisches 178
  - Kopie
    - flache 468
    - tiefe 355
  - Kopierkonstruktor 355
  - Kredit 61
  - Kreis 127, 146, 465
  - kubisch 128
  - Kugel 90

**L**

Länge  
     String- 36, 110  
 Laufbedingung 69  
 Laufzeit  
     Algorithmen 368  
     Typinformation 455  
 left 45  
 length() 109  
 Letter 404  
 Line 464  
 linear  
     Polynom 128  
 Linie 465  
 Linienzug 465  
 list 466  
 Liste  
     inhomogene 466  
     Initialisierung 223  
     konstante 467  
     sortierte 502  
     verkettete 332, 466  
 list-Klasse 467  
 Literal 23  
     String- 109  
 logic\_error 545  
 lokal 35, 139  
 long double 23  
 long long 23  
 LONG\_MAX 62  
 LONG\_MIN 62  
 LongArr 357  
 LRN-Schema 367

**M**

Mail 403  
 MailService 404  
 main() 27  
 Makro 85, 123  
 Manipulator 45, 46  
 Marke 69  
 Maschinencode 123, 172  
 Matrix 223, 228, 245  
 Maximum 227  
 MeasureArr 356  
 Median 77, 127  
 Mehrdeutigkeit  
     bei Mehrfachvererbung 517  
     bei Typumwandlungen 311  
 Mehrfach  
     -Inkludierung 85

Mehrfachvererbung 517  
 Mehrfachzuweisung 57  
 Memory Leaks 363  
 Menü 17  
 merge() 250  
 Methode 33, 189  
     aktivieren 189  
     const 191  
     deaktivieren 189  
     Default- 358  
     Definition 172  
     inline 189  
     Read-Only 189  
     rein virtuelle 495  
     statische 210  
     virtuell 455  
 Methodentabelle  
     virtuell 458, 495  
 Minimum 69, 227  
 Minute 61, 164  
 Mischen  
     sortierter Vektoren 249  
     von Dateien 266  
 Mittel  
     arithmetisches 60, 144, 227, 251  
     geometrisches 128, 248  
 MobilePhone 193, 213, 250  
 Modul 14  
 Modulodivision 57, 130  
 Modus  
     Binär- 261  
     Eröffnungs- 259  
 Mönch 231  
 MoneyChanger 214  
 Moore, J.S. 337  
 Move  
     -Konstruktor 355  
     -Zuweisung 355  
 Multiplikation 57  
     Polynom 361  
 Münzen 176, 286  
 Murphy 27, 43  
 MyData 438, 462  
 MyString 402

**N**

Nachfolger 364  
 Name 13, 25  
     Alias- 157  
     Header-Datei 33  
     Programm- 243  
     Variablen- 23

Namensbereich 14, 35, 139, 141  
     verschachteln 142  
 namespace 13  
 NameValue 292  
 NameValueArr 292  
 new 329  
 New-Handler 331  
 Newline 25, 114  
 NLR-Schema 368  
 Node 365  
 Norm  
     komplexer Zahl 291  
 noshowpos 45  
 Notation  
     exponentielle 45, 49  
 noupperc case 45  
 npos 162  
 NULL 251

**O**

Oberfläche  
     Kugel 90  
 Objekt 33, 35, 171  
     -Adresse 157  
     aktuelles 189  
     Auf-/Abbau 395  
     definieren 171  
     global 139, 140  
     konstantes 189  
     Konvertierung 311  
     lokal 139  
     persistentes 259  
     Speicherklasse 139  
     statisches 140  
     Teil- 207  
 Objektdatei 14  
 objektorientiert 14  
     Analyse 171  
 oct 45  
 Öffnen  
     Datei 259  
 ofstream 259  
 oktal 23  
 OOP 395  
 open() 259  
 Operand 57  
 Operator 57, 431  
     Adress- 157  
     arithmetischer 57  
     Bereichs- 139  
     binär 283  
     Cast- 99, 431

- delete 329, 455
- dynamic\_cast<> 455
- Funktion 283
- Index 109, 289
- logischer 57
- new 329
- Pfeil- 173
- Präfix- 286
- Punkt- 173
- sizeof() 224
- Symbol 283
- typeid() 455
- überladen 283
- unär 283
- Vergleichs- 57
- Verweis- 157
- virtueller 495
- Zuweisung- 57, 173
- operator
  - Schlüsselwort 283
- Operatorfunktion
  - virtuelle 495
- ostream 45, 46
- P**
  - Paar 113, 292
  - Pair 315, 318
  - Paket 403
  - Parameter 33
    - const 243
    - Makro- 85
    - Zeiger 243
  - Parcel 404
  - Pascal 229
  - permanent 139
  - Persistenz
    - von Objekt 259
  - Pfeiloperator 173
  - Pi 90
  - PiggyBank 176, 194
  - Player 177
  - Point 178, 463
  - Point2D 316
  - Point3D 288, 315
  - Pointer
    - Smart 363, 468
  - Polygon 464, 465
  - Polyline 463
  - Polymorphie 455
  - Polynom 248
    - auswerten 361
    - kubisches 128
    - lineares 128, 129
    - n-ten Grades 248, 360
    - quadratisches 129
  - Polynomial 360
  - pop\_back() 468
  - pop\_front() 468
  - Position 111
    - Datei- 260
  - Potenz 129
  - Potenzdarstellung 115
  - Präfix
    - Operator 286
  - Präprozessor 85
  - Primzahl 127, 570
  - Priorität 57
  - Priority Queue 501
  - private 395
  - Produkt
    - komplexer Zahlen 292
    - Skalar- 289
  - Programm
    - Dokumentation 13
    - Erstellung 13
    - Name 243
  - Programmierung
    - modulare 144
  - protected 395, 398
    - Element 398
  - Prototyp 33, 36, 124
  - Proxy 501
  - Prüfungsergebnis 290
  - Pseudo-Zufallszahlen 144, 177, 193
  - public 171, 395
  - Punkt 33
    - dreidimensionaler 288
    - zweidimensionaler 178, 316
  - Punkttestand 177
  - Punktoperator 173
  - push\_back() 467
  - push\_front() 467
  - put() 259
  - Q**
    - Quadrat 27, 75
    - Quelldatei 14, 124, 144
    - Queue 336
    - Quick-Sort Algorithmus 368
    - Quotient
      - komplexer Zahlen 292
    - quotient 163
  - R**
    - Radius
      - Kreis 127
    - rand() 177
    - Random 177, 193
    - Ratenzahlung 61
    - Raum
      - dreidimensionaler 288
      - zweidimensionaler 316
    - read() 259
    - Read-Only
      - Methode 189
      - Referenz 157
      - Zeiger 243
    - Realteil 290
    - Rechteck 465
    - Rectangle 401, 464
    - Redefinition
      - Element- 395
    - Referenz 157, 158
      - auf Basisklasse 431, 495
      - Read-Only 157
      - Typ 157
    - rein virtuell 495
    - rekursiv 123
      - Funktion 126
    - replace() 109, 114, 162
    - reserve() 163
    - return 13
    - Return-Wert
      - Objekt 189
      - Zeiger 243
    - rfind() 109, 114
    - RGB\_Color 195, 215
    - RGB-Einheitswürfel 195
    - right 45
    - RTTI 461
    - Rückgabe 189
    - Runden
      - Geldbetrag 214
      - Gleitpunktzahl 106
    - runtime\_error 545
    - Run-Time-Type-Informati-  
on 461
  - S**
    - Schachteln
      - Fehlerbehandlung 543
      - if-else 69
    - Schaltjahr 63
    - Schema
      - Horner 115, 361
      - LRN 367
    - Schleife 69
    - Schleifenkopf 69
    - Schließen
      - Datei 259
    - Schlüssel 365

- Schlüsselwort 23
- Schnittstelle 172
  - Klassen 172
  - öffentliche 395
- scientific 45
- SearchTree 365
- Seiteneffekt 123
- Sekunde 61, 164
- Semikolon 13
- sequentiell
  - Durchlauf 466
- Sequenz
  - Escape 23
- setfill() 45
- setprecision() 45
- setw() 45
- Shape 463
- Shape2D 400
- ShapePtrList 466
- Shift
  - Bad-Character 339
  - Good-Suffix- 339
- showpos 45
- Sieb
  - Eratosthenes- 570
- Signatur 123, 125, 456
  - von Konstruktoren 190
- Simulation
  - Würfelspiel 177
- sin() 37
- Sinus 37
- size\_t 26, 109
- size() 467
- sizeof() 24, 224
- Skalarprodukt 289
- Smart Pointer 363, 468
- Software
  - Wiederverwendbarkeit 395
- Sparschwein 176
- Speicherplatz 171
- Speicher
  - dynamischer 329
  - freigeben 330
  - Leck 363, 468
  - reservieren 330
- Speicherklasse 139
- Spezialisierung 395, 431
- Spezifizierer 139
- Sprung
  - goto 69
  - Unterprogramm- 123
- Square 401
- srand() 177
- Stack
  - Unwinding 543
- Standard
  - Bibliothek 33
  - Ein-/Ausgabe 85
  - Fehlerklasse 543
  - Funktion 33
  - Header-Datei 33
  - Kopierkonstruktor 355
  - Makro 85
  - Methode 189
- Standardfehlerklasse
  - exception 545
  - logic\_error 545
  - runtime\_error 545
- Standardklasse
  - fstream 259
  - ifstream 259
  - ios 45
  - istream 45
  - list 467
  - ofstream 259
  - ostream 45
  - string 33, 109
  - type\_info 461
- Standard-Template-Library 466
  - static 139
  - static\_cast<> 431
  - statisch
    - Funktion 141
    - Objekt 140
- Status
  - Flag 259
  - Stream 259
- std 13, 35, 139
- stdlib.h 177
- Steuerzeichen 91
- STL 466
- strcpy() 438
- Stream 45
  - Klassen 33
  - Status 259, 563
- Stream *siehe auch* Datei
- String
  - Definition 113
  - Endezeichen 25, 223
  - ersetzen 162
  - extrahieren 113, 293
  - invertieren 163
  - Konstante 25
  - Länge 109, 110
  - Literal 109
  - Suche 162, 337
  - verketteten 36
  - verschlüsseln 114
  - zerlegen 251
- string 33, 109
- StringMatching 338
- strlen() 438
- struct 171
- Struktur 171
- Stunde 61, 164
- Subtraktion
  - bei Vektoren 243
  - Polynom 361
- Suchbaum 365
- Suchstring 337
- Suchverfahren
  - Boyer-Moore 337
- Suffix 339
- Summe 57, 74, 144
  - komplexer Zahlen 291
  - Vektor 334
- switch 69, 72
- Symbol
  - Operator 283
- system() 293
- T**
  - Tabelle
    - Such- 337
  - Tabulator 25, 114
  - Teilbaum 364
  - Teiler 128
    - größter gemeinsamer 130
  - Teilobjekt 207, 517
  - Teilstring 162
  - Temperatur 61
  - Template 466
  - Textdatei 85
  - Textzeile 36
    - einlesen 439
  - this 189
  - throw 543
  - time\_t 406, 524
  - time.h 145
  - time() 145, 524
  - Token 251
  - tolower() 85, 402
  - Ton 27
  - toupper() 85, 402
  - TowersOfHanoi 230
  - TraceParcel 405
  - Tree-Sort Algorithmus. 368
  - true 23

try-Block 543  
 Türme von Hanoi 230  
 Typ 23  
 type\_info 461  
 typeid() 455  
 typeinfo 461  
 Typinformation  
   zur Laufzeit 455  
 Typumwandlung  
   explizit 99  
   für Klassen 311  
   in Klassenhierarchien 431  
   Mehrdeutigkeit 311  
   übliche arithmetische 99

**U**  
 Überladen  
   Funktionen 123  
 Überladung  
   Operator- 283  
   Zuweisungsoperator 355  
 Überlauf 62  
 Übersetzungseinheit 140  
 UCHAR\_MAX 337  
 UINT\_MAX 49  
 Umfang  
   Ellipse 465  
   Kreis- 146  
   Quadrat 27  
 Umgebungsvariable 293  
 Umlenken  
   Ein-/Ausgabe 89  
 Umrechnung  
   binär->dezimal 75  
   dezimal->binär 75  
 Umwandlung  
   arithmetischer Typen 99  
   dezimal -> hex 115  
   hex -> dezimal 115  
   Sekunden 164  
 unär 283  
   Operator 57  
 Union 171  
 Unterprogrammssprung 123  
 Unterstrich 23  
 Up-Cast 431  
 uppercase 45  
 using 139, 141

**V**  
 Variable 23, 25  
   Definition 25  
 Vektor 223, 243  
   assoziativer 292  
   dynamischer 329, 332, 335  
   komprimieren 335  
   mehrdimensionaler 223  
   mischen 249  
   -Name 243  
   Summme 334  
   Zeiger- 243  
 Vererbung 395  
 Vererbungsart 517  
 Vergleich  
   von Strings 109  
 Verketten  
   Strings 110  
 Verschlüsseln  
   Strings 114  
 Verweisoperator 157  
 Verzweigung 69  
 Vielfaches 128  
 Viereck 465  
 virtual 455  
 virtuell  
   Basisklasse 517  
   Destruktor 455  
   Methode 455  
   Methodentabelle 458  
 void 33  
 Volumen  
   Kugel 90  
   Zylinder 127  
 Vorgänger 364  
 Vorrang 57  
 Vorzeichen 45, 46

**W**  
 Wallet  
   Klasse 286  
 Warnung 14  
 Warteschlange 336  
 Wechselkurs 214  
 Wert  
   Default- 123  
 Wertebereich 26  
 while 69, 74  
 Wiederverwendbarkeit 395  
 write() 259  
 Würfelspiel 177  
 Wurzel 128, 364

**Y**  
 Yen 214

**Z**  
 Zahl  
   Festpunkt- 45  
   komplexe 290  
   konjugiert 291  
   Norm 291  
 Zahlensystem 45  
 Zeichen  
   -Konstante 23  
   Newline 114  
   Steuer- 91  
   Zwischenraum- 25, 92  
 Zeichenkette 33  
 Zeiger 157, 160, 243  
   als Parameter 243, 495  
   Basisklassen- 431  
   intelligenter 363, 468  
   NULL 251  
   Read-Only 243, 246  
   this 189  
   -Vektor 243  
 Zeigerarithmetik 243  
 Zeigerversion 243  
 Zeile  
   Kommando- 243  
 Ziffer 23  
 Zufallszahl 35, 144, 193  
   ganzzahlig 145  
   gebrochen 145  
 Zufallszahlengenerator 145, 177  
 Zugriffsmethode 189, 192  
 Zugriffsproxy 501  
 Zugriffsrecht  
   Datei- 259  
 Zuweisung 58  
   einfache 57  
   Move- 355  
   von Objekten 173  
 Zuweisungsoperator  
   überladen 355, 359  
   virtueller 495  
 Zweig  
   if-/else- 69  
 Zwischenraumzeichen 25, 91  
   entfernen 114  
 Zylinder 127